



ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Add-ins et Scriptant

Author: Sparx Systems

Date: 7/11/2024

Version: 17.0

CRÉÉ AVEC  **ENTERPRISE
ARCHITECT**

Table des Matières

Add-ins et Scriptant	13
Scriptant	15
Fenêtre Scriptant	19
Propriétés du groupe de scripts	21
JavaScript Math Library	23
Arithmétique et Algébrique	24
sqrt	25
lsqrt	26
cbt	27
polevl , p1evl	28
chbevl	30
round	31
floor	32
ceil	33
frexp	34
ldexp	35
fabs	36
signbit	37
isnan	38
isfinite	39
poladd	40
polsub	41
polmul	42
poldiv	43
polsbt	44
poleva	45
polclr	46
polmov	47
Exponentiel et Trigonométrie	48
acos	49
acosh	50
asinh	51
atanh	52
asin	53
atan	54
atan2	55
cos	56
cosdg	57
exp	58
exp2	59
exp10	60
cosh	61
sinh	62
tanh	63
log	64
log2	65
log10	66

pow	67
powi	68
sin	69
sindg	70
tan	71
tandg	72
Intégrale Exponentielle	73
expn	74
shichi	75
sici	77
Fonctions gamma	79
beta	80
lbeta	81
fac	82
gamma	83
lgam	84
incbet	85
incbi	87
igam	88
igamc	89
igami	90
psi	91
rgamma	93
Fonction d'erreur	94
erf	95
erfc	96
dawsn	97
fresnl	98
Fonctions Bessel	100
airy	101
j0	103
j1	104
jn	105
jv	106
y0	107
y1	108
yn	109
yv	110
i0	111
i0e	112
i1	113
i1e	114
iv	115
k0	116
k0e	117
k1	118
k1e	119
kn	120
Hypergéométrique	121
hyperg	122
hyp2f1	123

hyp2f0	125
onef2	126
threef0	127
Elliptique	128
ellpe	129
ellie	130
ellpk	131
ellik	133
ellpj	134
Probabilité	135
bdtr	136
bdtrc	138
bdtri	140
chdtr	141
chdtrc	142
chdtri	143
fdtr	144
fdtrc	145
fdtri	147
gdtr	149
gdtrc	150
nbdtr	151
nbdtrc	152
ndtr	153
ndtri	154
pdtr	155
pdtrc	156
pdtri	157
stdtr	158
Divers	160
polylog	161
spence	163
zetac	164
zeta	165
struve	167
Matrice	168
fftr	169
simq	170
minv	171
mmpy	173
mvmpy	174
mtransp	175
eigens	176
Intégration Numérique	179
simpsn	180
Arithmétique Complexe	181
cadd	182
csub	184
cmul	186
cdiv	188
cabs	190

csqrt	191
Exponentiel Complexe et Trigonométrie	193
cexp	194
clog	195
ccos	196
cacos	197
csin	198
casin	199
ctan	200
catan	201
ccot	202
erreurs	203
JavaScript Console	204
Fenêtre de la console	207
Interface Solveurs	208
Éditeur de Script	209
Objet de Session	212
Flux de travail	213
Fonctions de script de workflow	215
Fonctions - Valider et contrôler la saisie de l'utilisateur	217
Fonctions - Créer une recherche avec des tâches utilisateur	219
Structures de données de flux de travail remplies	220
Structures de données de flux de travail que vous remplissez	222
Fonctions que vous appelez	223
Débogage de Script	224
Scriptant Hybride	225
Exemple C#	227
Exemple Java	229
Modèle Add-Ins de modèle	231
Créer un Add-In	233
Répondre aux Événements	235
Modifier le code Add-In	236
Gestion des modules Add-In Modèle	237
Bibliothèque de référence des signaux	238
Exemples de Add-Ins	239
Enterprise Architect Modèle Add-In	240
Le gestionnaire Add-In	241
Créer et déployer Add-Ins	242
Créer Add-Ins	243
Définir Items du menu	244
Déployer Add-Ins	246
Conseils et astuces	248
Recherche Add-In	251
EA_SampleSearch	252
Format XML (recherche de données)	253
Événements Add-In	255
EA_OnAddinPropertiesTabChanging	256
EA_Connect	257
EA_Disconnect	258
EA_GetMenuItems	259
EA_GetMenuState	261

EA_GetRibbonCategory	263
EA_MenuClick	264
EA_OnOutputItemClicked	266
EA_OnOutputItemDoubleClicked	267
EA_ShowHelp	268
Événements diffusés	269
Add-In Gestion de Licence Événements	271
EA_AddinLicenseValidate	272
EA_AddinLicenseGetDescription	273
EA_GetSharedAddinName	274
Tableau personnalisé Événements	276
EA_OnCustomTableBeginEdit	277
EA_OnCustomTableEndEdit	278
EA_OnCustomTableSelectionChanged	279
EA_OnCustomTableCellUpdated	280
Compositeur de Schéma Événements	281
EA_GenerateFromSchema	282
EA_GetProfileInfo	283
EA_IsSchemaExporter	284
Compartment Events	285
EA_QueryAvailableCompartments	286
EA_GetCompartmentData	288
Context Item Events	291
EA_OnContextItemChanged	292
EA_OnContextItemDoubleClicked	293
EA_OnNotifyContextItemModified	294
EA_FileClose	295
EA_FileNew	296
EA_FileOpen	297
EA_OnPostCloseDiagram	298
EA_OnPostInitialized	299
EA_OnPostOpenDiagram	300
EA_OnPostTransform	301
EA_OnPreExitInstance	302
EA_OnRetrieveModelTemplate	303
EA_OnTabChanged	305
EA_LoadWindowManager	306
Modèle Validation Événements	308
EA_OnInitializeUserRules	309
EA_OnStartValidation	310
EA_OnEndValidation	311
EA_OnRunElementRule	312
EA_OnRunPackageRule	313
EA_OnRunDiagramRule	314
EA_OnRunConnectorRule	315
EA_OnRunAttributeRule	316
EA_OnRunMethodRule	317
EA_OnRunParameterRule	318
Exemple de validation Modèle	319
Post-New Events	325
EA_OnPostNewElement	326

EA_OnPostNewConnector	327
EA_OnPostNewDiagram	328
EA_OnPostNewDiagramObject	329
EA_OnPostNewAttribute	330
EA_OnPostNewMethod	331
EA_OnPostNewPackage	332
EA_OnPostNewGlossaryTerm	333
Pre-Deletion Events	334
EA_OnPreDeleteElement	335
EA_OnPreDeleteAttribute	336
EA_OnPreDeleteMethod	337
EA_OnPreDeleteConnector	338
EA_OnPreDeleteDiagram	339
EA_OnPreDeleteDiagramObject	340
EA_OnPreDeletePackage	341
EA_OnPreDeleteGlossaryTerm	342
Pre New-Object Events	343
EA_OnPreNewElement	344
EA_OnPreNewConnector	345
EA_OnPreNewDiagram	346
EA_OnPreNewDiagramObject	347
EA_OnPreDropFromTree	348
EA_OnPreNewAttribute	349
EA_OnPreNewMethod	350
EA_OnPreNewPackage	351
EA_OnPreNewGlossaryTerm	352
Valeur Étiquetée Événements	353
EA_OnAttributeTagEdit	354
EA_OnConnectorTagEdit	355
EA_OnElementTagEdit	356
EA_OnMethodTagEdit	357
Technology Events	358
EA_OnInitializeTechnologies	359
EA_OnPreActivateTechnology	360
EA_OnPostActivateTechnology	361
EA_OnPreDeleteTechnology	362
EA_OnDeleteTechnology	364
EA_OnImportTechnology	365
Technology Rules	366
EARules_Initialize	367
Diagram Appearance Rule Events	369
EARules_ClosePartitionName	370
EARules_ElementDisplayName	371
EARules_GetCompartmentItem	372
EARules_GetCompartmentName	374
EARules_GetNameUnderline	375
EARules_GetPropertyString	376
EARules_GetShapeScript	377
EARules_ShowStereotype	378
EARules_StereotypeDisplayName	379
User Interface Rule Events	380

EARules_AllowNesting	381
EARules_AppendChildDiagrams	382
EARules_AppendChildElements	384
EARules_CanOverrideStereotype	386
EARules_CanProxy	387
EARules_CanReparent	388
EARules_CreateModel	389
EARules_EnableElementProperty	390
EARules_ForceLength	392
EARules_GetEquivalentDiagram	393
EARules_IsAdjustable	394
EARules_PropagateStereotype	395
EARules_ShowElementProperty	396
EARules_ShowFrame	398
EARules_ShowParentFrame	399
Vues personnalisées	400
Créer une Vue personnalisée	401
Fenêtre Ancrée Personnalisée	402
Add-Ins MDG	404
Événements MDG	405
MDG_BuildProject	406
MDG_Connect	407
MDG_Disconnect	408
MDG_GetConnectedPackages	409
MDG_GetProperty	410
MDG_Merge	411
MDG_NewClass	414
MDG_PostGenerate	415
MDG_PostMerge	416
MDG_PreGenerate	417
MDG_PreMerge	418
MDG_PreReverse	419
MDG_RunExe	420
MDG_View	421
Add-In Workflow Événements	422
EA_AllowPropertyUpdate	423
EA_AllowTagUpdate	424
EA_CanEditProperty	425
EA_CanEditTag	426
Enterprise Architect Modèle d'Objet	427
Utilisation de l'interface d'automatisation	428
Se connecter à l'interface	429
Définir des références dans Visual Basic	432
Exemples et Conseils	433
Appel de Enterprise Architect	435
Ressources disponibles	437
Référence	438
Interface Overview	439
Object d'application	441
Enumerations	442
ConstLayoutStyles	444

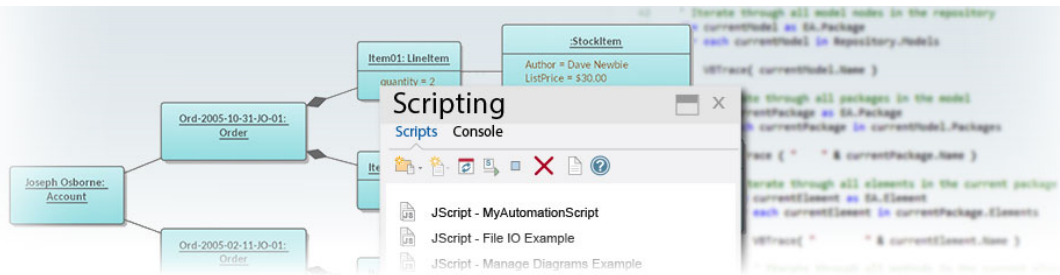
CreateBaselineFlag	445
CreateModelType	446
DocumentBreak	447
DocumentPageOrientation	448
DocumentType	449
EAEditionTypes	450
EnumRelationSetType	451
ExportPackageXMIFlag	452
MDGMenus	453
MessageFlag	454
ObjectType	455
PropType	457
ReloadType	458
ScenarioDiagramType	459
ScenarioStepType	460
ScenarioTestType	461
XMIType	462
Repository Package	463
Author Class	464
Client Class	466
Collection Class	468
The AddNew Function	470
Datatype Class	474
EventProperties Class	477
EventProperty Class	478
ModelWatcher Class	479
Package Class	480
ProjectIssues Class	490
ProjectResource Class	492
ProjectRole Class	494
PropertyType Class	496
Reference Class	498
Classe Référentiel	500
SecurityUser Class	524
Stereotype Class	526
Task Class	528
Term Class	530
Properties Tab Package	532
PropertiesTab Class	533
Element Package	535
Constraint Class	537
Effort Class	539
Classe d'élément	541
Classe ElementGrid	556
File Class	557
Issue (Maintenance) Class	559
Metric Class	561
Requirement Class	563
Resource Class	565
Risk Class	567
Scenario Class	569

ScenarioExtension Class	571
ScenarioStep Classe	572
TaggedValue Class	575
Test Class	577
Element Features Package	579
Attribute Class	580
AttributeConstraint Class	586
AttributeTag Class	588
CustomProperties Collection	590
EmbeddedElements Collection	591
Method Class	592
MethodConstraint Class	597
MethodTag Class	599
Parameter Class	601
ParamTag Class	604
Partitions Collection	606
Properties Class	607
TemplateParameter Class	609
Transitions Collection	611
Connector Package	612
Connector Class	613
ConnectorConstraint Class	621
ConnectorEnd Class	623
ConnectorTag Class	626
RoleTag Class	628
TemplateBinding Class	630
Diagram Package	632
Classe Diagramme	633
DiagramLink Class	641
Classe DiagramObject	644
SwimlaneDef Class	650
Swimlanes Class	652
Swimlane Class	654
Project Interface Package	655
Classe de projet	656
Chart Package	674
Énumérations de graphique	675
ChartAxisCrossType	676
ChartAxisIndex	677
ChartAxisLabelType	678
ChartAxisTickMarkTypeChartAxisTickMarkType	679
Type d'axe de graphique	680
FormeBarreGraphique	681
CatégorieGraphique	682
ChartColorMode	684
ChartCurveType	685
Style de tiret de graphique	686
Style de cadre de graphique	687
ChartGradientTypeChartGradientType	688
GraphiqueMarkerShape	689
GraphiqueStockSeriesType	690

Type de graphique	691
ChartWallOptions	692
Classe de graphique	693
Classe ChartAxisIndexChartAxisIndex Class	696
Classe ChartDataValueChartDataValue Class	698
Classe ChartDiagram3DChartDiagram3D Class	699
Classe ChartFormatSeriesChartFormatSeries Class	700
Classe ChartSeries	701
Document Generator Interface Package	705
Classe DocumentGenerator	706
Paquetage Code Miner	712
Interface CMService	713
Interface CMDDataSet	715
Interface CMDDataNode	717
Interface CMFacet	719
Exemple de script Code Miner	720
Paquetage Data Miner	723
DataMinerManager Class	724
DataMiner Class	726
DataSet Class	727
DMArray Class	728
DMAAction Class	729
DMScript Class	730
DMConnection Class	731
TypeInfoProperties Package	732
TypeInfoProperties Class	733
TypeInfoProperty Class	734
Mail Interface Package	735
MailInterface Class	736
Search Window Package	739
EAContext Class	740
EASelection Class	742
SearchWindow Class	744
Simulation Package	746
Simulation Class	747
Compositeur de Schéma Paquetage	749
SchemaProperty Class	750
SchemaProfile Class	752
SchemaComposer Class	753
ModelTypeEnum Class	755
ModelType Class	756
SchemaTypeEnum Class	758
SchemaType Class	759
SchemaPropEnum Class	760
SearchType Enumeration	761
SchemaNamespace Class	762
SchemaNamespaceEnum Class	763
Exemples de code	764
Open the Repository	765
Iterate Through a .EAP File	766
Add and Manage Packages	767

Add and Manage Elements	769
Add a Connector	770
Add and Manage Diagrams	772
Add and Delete Features	773
Element Extras	774
Repository Extras	778
Stereotypes	781
Work With Attributes	782
Work With Methods	784


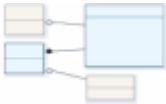


Add-ins et Scriptant



Enterprise Architect dispose d'une incroyable gamme de fonctionnalités intégrées pour travailler avec des modèles, mais il fournit également une gamme d'environnements pour accéder et manipuler le contenu d'un référentiel par programmation. Il s'agit d'une facilité extrêmement utile qui vous donne une capacité illimitée pour interroger et manipuler des modèles, ajouter à l'interface utilisateur Enterprise Architect, générer des rapports et même créer support pour de nouveaux langages modélisation. L'interface d'automatisation vous donne accès au Modèle Object, qui est un ensemble d'objets facile à utiliser et bien défini avec des propriétés et des méthodes qui peuvent être utilisées pour interroger et manipuler le référentiel et son contenu, évitant au programmeur d'avoir à connaître les structures de données sous-jacentes du référentiel.

L'interface d'automatisation est disponible à partir d'un framework de script intégré à l'interface utilisateur Enterprise Architect, via des environnements de script externes ou via Add-Ins qui peuvent être créés dans une large gamme de langages de programmation.

Facilités

Facilité	Description
 <p>Scriptant</p>	Découvrez la capacité de script flexible et facile à utiliser pour inspecter et/ou modifier par programmation des éléments dans votre modèle actuellement ouvert.
 <p>Modèle Object</p>	Découvrez le Enterprise Architect Modèle d'Objet. Écrivez vos propres programmes personnalisés qui accèdent aux informations stockées dans Enterprise Architect.
 <p>Modèle Add-In</p>	Le Enterprise Architect Modèle Add-In vous aide à développer les fonctionnalités fournies par l'interface d'automatisation pour vous permettre d'étendre l'interface utilisateur Enterprise Architect.
 <p>Add-Ins MDG</p>	Add-Ins MDG sont des types de Add-In spécialisés qui ont fonctionnalités et des exigences supplémentaires. Add-Ins MDG sont axés sur la génération, la synchronisation et les processus généraux liés à la conversion de modèles en code et de code en modèles.
<p>Exemples de code et références</p>	Accédez à la richesse des connaissances et des exemples pour vous aider à compléter votre Add-In.



Scriptant



L'environnement de script d' Enterprise Architect est un facilité flexible et facile à utiliser qui supporte à la fois JavaScript et les langages de script Microsoft JScript et VBScript. Lorsqu'un script s'exécute, il a accès à un objet « Référentiel » intégré. A l'aide de cet objet script, vous pouvez inspecter et/ou modifier par programmation des éléments dans votre modèle actuellement ouvert. Enterprise Architect fournit également des éditeurs riches fonctionnalité et des outils pour exécuter , déboguer et gérer vos scripts. Scripts sont modulaires et peuvent inclure d'autres scripts par nom à l'aide de la directive *!include* . Ils peuvent être utilisés à des fins très diverses, de la documentation à la validation et au refactoring, et ils peuvent être d'une aide précieuse pour automatiser des tâches chronophages.

Support du moteur de script

- Mozilla SpiderMonkey [version 1.8]
- Moteur Microsoft Scriptant

Langages d'écriture

- JavaScript
- JScript
- VBScript

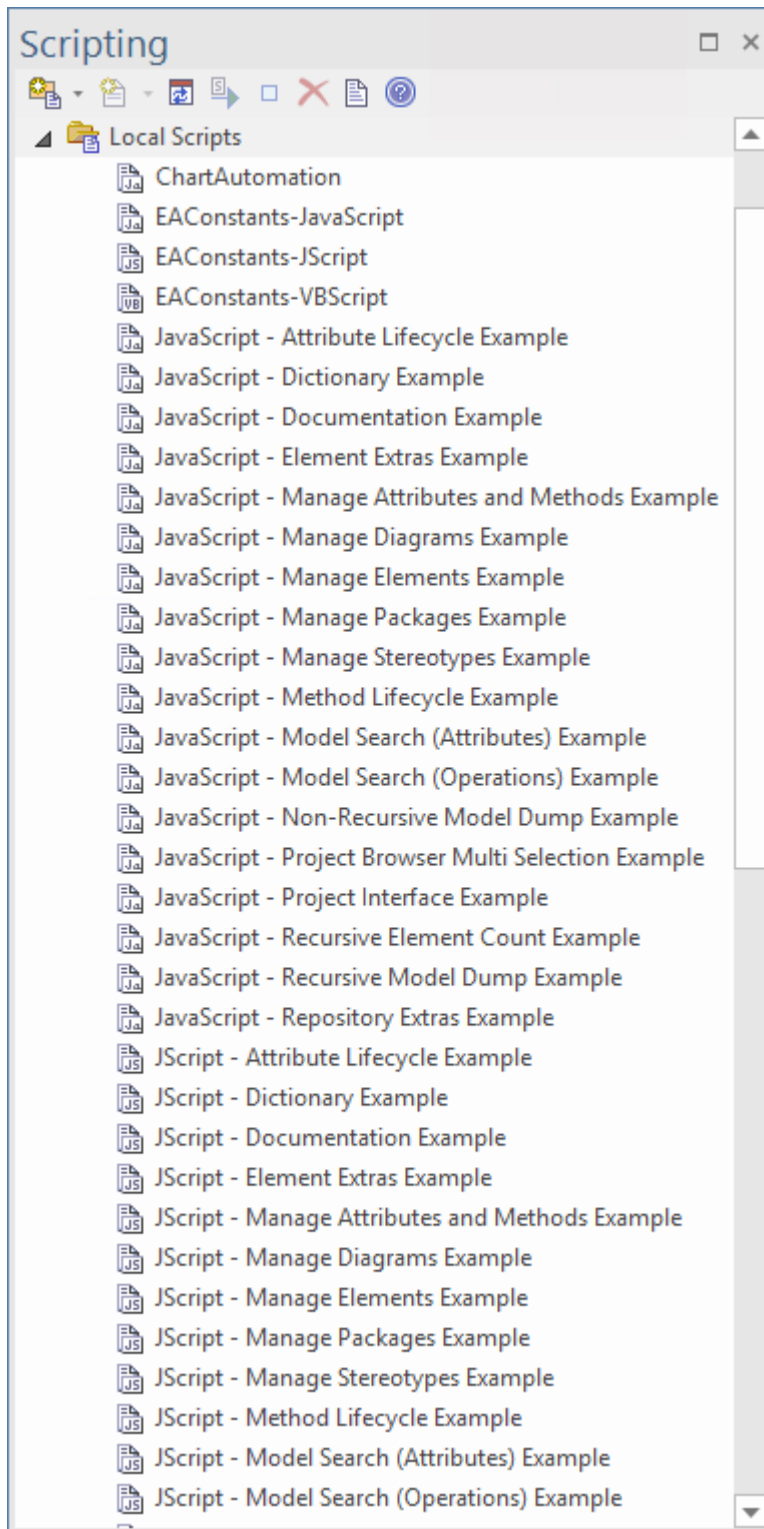
Avantages

- Inspection et rapport sur la composition du modèle et des éléments
- Modification et mise à jour des propriétés des éléments
- Exécution de requêtes pour obtenir des informations étendues sur le modèle
- Modification des schémas diagramme
- Être appelé à partir gabarits de documents de rapport pour remplir les rapports
- Créer et mettre en œuvre des flux de travail de processus
- Être inclus dans MDG Technologies pour augmenter les langages spécifiques au domaine
- Accès étendu à UI des scripts via des menus contextuels
- Rôle du serveur d'automatisation pour les clients COM en cours et hors processus (Scriptant est lui-même un exemple de client en cours ; Add-Ins sont un autre)
- Gouvernance de l'accès aux éléments grâce à la sécurité du workflow
- Intégration de la recherche Modèle

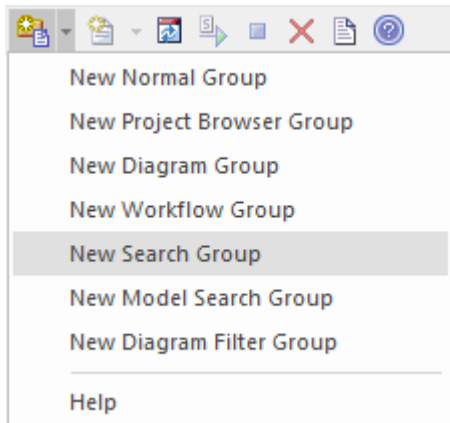
Utilisation de Scripts

L'interface de gestion de Scriptant est la fenêtre Scriptant , affichant l' Vue des scripts, que vous utilisez pour révision , créer et éditer des scripts.

À l'exception des Scripts locaux, qui sont basés sur des fichiers et installés avec Enterprise Architect , tous les autres scripts sont stockés en tant qu'actifs de modèle et peuvent être partagés avec tous les utilisateurs du modèle. Les débogueurs de scripts peuvent vous aider à développer des scripts et les éditeurs de scripts peuvent vous fournir des informations sur les interfaces d'automatisation à votre disposition. Vous pouvez analyser l'exécution, par exemple en enregistrant un diagramme Séquence de l'exécution du script et en interrompant l'exécution pour afficher les variables locales.



Groupes de scripts



Scripts sont gérés et contenus dans des groupes. Chaque groupe possède un attribut appelé ' Type '. Cet attribut est utilisé pour aider Enterprise Architect à décider comment et où le script peut être utilisé et à partir de quelles fonctionnalités il doit être rendu disponible. Les propriétés d'un groupe de scripts peuvent être visualisées à partir de son menu contextuel.

Stockage de scripts

Les scripts intégrés sont basés sur des fichiers et sont installés avec Enterprise Architect . Ils apparaissent sous le groupe *Scripts locaux* .

Vous ne pouvez pas modifier ou supprimer les scripts locaux, mais vous pouvez copier le contenu assez facilement.

Les scripts définis par l'utilisateur sont basés sur des modèles et peuvent donc être partagés par une communauté. Ils sont répertoriés dans le groupe auquel ils appartiennent.

Utilisation Solveurs

Anywhere dans Enterprise Architect qui contient du code JavaScript , comme dans Simulation , vous pouvez maintenant utiliser une construction JavaScript appelée « Solveur » (la classe Solveur) pour intégrer des outils externes et utiliser directement les fonctionnalités de chaque outil pour effectuer de manière simple et intuitive des fonctions mathématiques et graphiques complexes. Les appels vous aident à échanger facilement des variables entre le moteur JavaScript intégré et chaque environnement. Deux bibliothèques mathématiques prises en charge sont MATLAB et Octave.

Pour utiliser la classe Solveur , vous devez avoir une connaissance des fonctions disponibles dans votre Bibliothèque mathématique préférée et des paramètres qu'elles utilisent, comme décrit dans la documentation du produit.

Faisant partie du moteur JavaScript , les classes Solveur sont également immédiatement accessibles aux auteurs de Add-In créant Add-Ins JavaScript basés sur des modèles.

Consultez également les rubriques d'aide *Octave Solveur* , *MATLAB Solveur* et *Solveurs* .

Notes

- Ce facilité est disponible dans les éditions Corporate , Unified et Ultimate
- Si vous avez l'intention d'utiliser le Scriptant facilité sous Crossover/ WINE , vous devez également installer Internet Explorer version 6.0 ou supérieure

Fenêtre Scriptant

La fenêtre Scriptant est composée d'une barre d'outils et d'une vue de tous les scripts par groupe. Les groupes de scripts et leurs scripts disposent également de menus contextuels qui fournissent certaines ou toutes ces options :

- Propriétés du groupe - pour afficher ou modifier les propriétés du groupe de scripts dans la dialogue ' Propriétés du groupe de scripts'
- Exécuter le script - pour exécuter le script sélectionné (ou appuyez sur Ctrl pendant que vous double-cliquez sur le nom du script)
- Script Débogage - pour déboguer le script sélectionné
- Modifier le script - pour mettre à jour le script sélectionné (ou double-cliquez sur le nom du script pour afficher l' Éditeur de Script , qui affiche généralement un gabarit de script, déterminé par le type de groupe d'utilisateurs tel qu'attribué lors de la création ou dans la dialogue « Propriétés du groupe de scripts »)
- Renommer le script - pour modifier le nom du groupe ou du script sélectionné
- Nouveau VBScript/JScript/ JavaScript - ajouter un nouveau script au groupe d'utilisateurs sélectionné
- Importer un script de workflow - pour afficher la dialogue « Navigateur » à travers laquelle vous localisez et sélectionnez un fichier source de script de workflow (.vbs) à importer dans le dossier de script de workflow
- Supprimer le groupe/script - pour supprimer le groupe d'utilisateurs ou le script sélectionné







Vous pouvez également déplacer ou copier un script d'un dossier de scripts utilisateur vers un autre ; pour :










- Déplacez un script, mettez-le en surbrillance dans la fenêtre Scriptant et faites-le glisser dans le dossier des scripts utilisateur auquel il appartient maintenant
- Copiez un script, mettez-le en surbrillance dans la fenêtre Scriptant et appuyez sur Ctrl pendant que vous le faites glisser dans le dossier des scripts utilisateur dans lequel le dupliquer

Accéder

Ruban	Spécialisation > Outils > Bibliothèque de scripts
-------	---------------------------------------------------

Barre d'outils de script

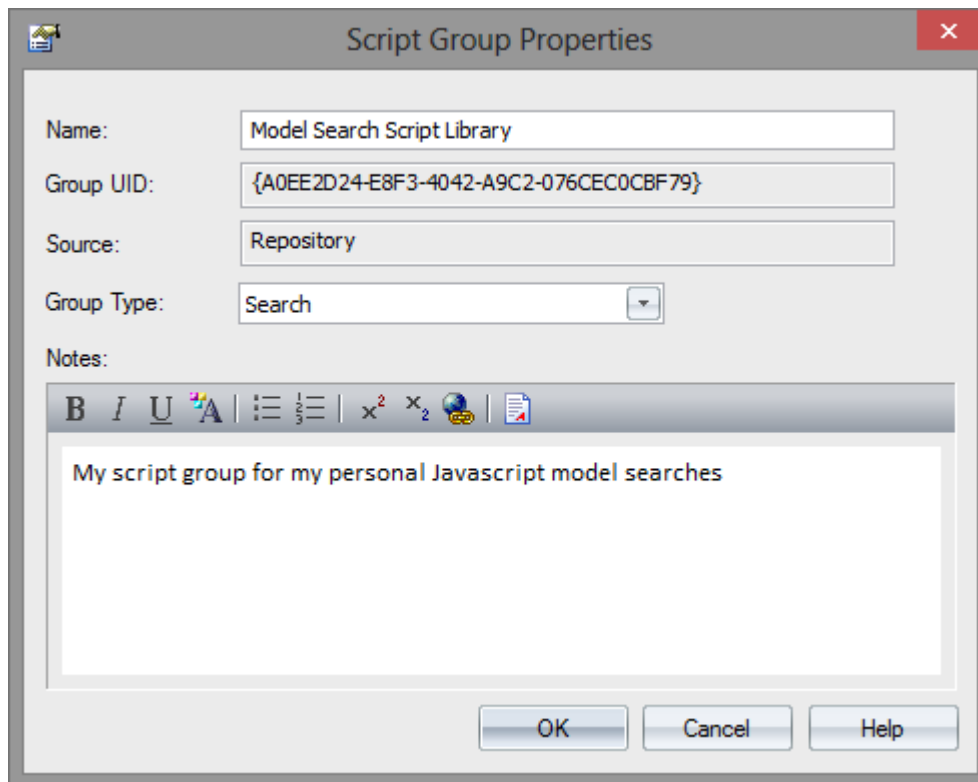
Icône	Action
	<p>Créer un nouveau groupe de scripts ; cette option affiche un court menu des types de groupes de scripts que vous pouvez créer, à savoir :</p> <ul style="list-style-type: none"> • Groupe normal () • Groupe de fenêtres Navigateur () • Groupe Diagramme () • Groupe de flux de travail () • Groupe de recherche () • Groupe de recherche Modèle <p>Le nouveau groupe est ajouté à la fin de la liste dans la fenêtre Scriptant , avec le texte « Nouveau groupe » mis en surbrillance afin que vous puissiez saisir le nom du groupe.</p>

	<p>Créez un nouveau fichier de script dans le groupe de scripts sélectionné ; cela affiche un court menu des types de scripts que vous pouvez créer, à savoir :</p> <ul style="list-style-type: none"> • VBScript () • JScript () • JavaScript () <p>Le nouveau script est ajouté à la fin de la liste dans le groupe sélectionné, avec le texte « Nouveau script » mis en surbrillance afin que vous puissiez saisir le nom du script.</p>
	<p>Actualisez l'arborescence des scripts dans la fenêtre Scriptant ; cette icône recharge également toutes les modifications apportées à un script de workflow.</p>
	<p>Compilez et exécutez le script sélectionné.</p> <p>La sortie du script est écrite dans l'onglet « Script » de la fenêtre Sortie système, que vous affichez à l'aide du bouton Sortie du script Vue .</p>
	<p>Arrêter un script en cours d'exécution ; l'icône est désactivée si aucun script n'est en cours d'exécution.</p>
	<p>Supprimer un script du modèle ; vous ne pouvez pas utiliser cette icône pour supprimer un groupe de scripts (voir l'élément « Menu Contexte » précédent), des scripts dans le groupe « Scripts locaux » ou un script en cours d'exécution.</p> <p>Le système vous prompts à confirmer la suppression uniquement si la case à cocher « Confirmer les suppressions » est sélectionnée dans le panneau « Navigateur de projet » de la page « Général » de la dialogue « Préférences » ; si cette option n'est pas sélectionnée, aucune prompt ne s'affiche.</p> <p>La suppression des scripts est permanente : les scripts ne peuvent pas être récupérés.</p>
	<p>Affichez la fenêtre de sortie système avec les résultats du script le plus récemment exécuté affichés dans l'onglet « Script ».</p>

Notes

- Ce facilité est disponible dans les éditions Corporate , Unified et Ultimate
- Si vous ajoutez, supprimez ou modifiez un script, vous devrez peut-être recharger le modèle pour que les modifications prennent effet.
- Si vous choisissez de supprimer un groupe de scripts contenant des scripts, le système vous prompts toujours à confirmer l'action, quels que soient les paramètres système pour les opérations de suppression. Assurez-vous que vous avez l'intention de supprimer le groupe et ses scripts avant de confirmer la suppression : la suppression des groupes de scripts et des scripts est permanente.

Propriétés du groupe de scripts









Lorsque vous créez un script, vous le développez dans un groupe de scripts dont les propriétés déterminent comment ce script doit être mis à disposition de l'utilisateur : via le menu contextuel de la fenêtre Navigateur pour agir sur des objets d'un type spécifique, ou via un menu contextuel diagramme . Vous créez un groupe de scripts à l'aide de la première icône de la barre d'outils de la fenêtre Scriptant .

Accéder

Ruban	Specialize > Outils > Script Bibliothèque > Scripts > cliquez-droit sur [Nom du groupe] > Propriétés du groupe
-------	----------------------------------------------------------------------------------------------------------------

Définir les Propriétés du groupe de scripts

Champ/Bouton	Action
Nom	Type le nom du groupe de scripts.
UID du groupe	(Lecture seule) Le GUID attribué automatiquement au groupe.
Source	(Lecture seule) L'emplacement du gabarit utilisé pour créer le script.
Type de groupe	Cliquez sur la flèche déroulante et sélectionnez le type de script contenu dans le

	<p>groupe ; cela peut être l'un des suivants :</p> <ul style="list-style-type: none"> • Normal - () Scripts de modèles généraux • Fenêtre Navigateur - () Scripts répertoriés et pouvant être exécutés à partir de l'option de menu contextuel « Scripts » de la fenêtre Navigateur • Workflow - () Scripts exécutés par le moteur de workflow d' Enterprise Architect ; vous ne pouvez créer que des scripts VB de ce type • Recherche - () Scripts pouvant être exécutés comme recherches de modèles ; ces scripts sont répertoriés dans le champ 'Recherche' de la fenêtre Recherche Modèle , dans la dernière catégorie de la liste • Diagramme - () Scripts pouvant être exécutés à partir du sous-menu « Scripts » du menu contextuel diagramme • Rechercher dans Projet - () Scripts pouvant être exécutés à partir du sous-menu « Scripts » d'un menu contextuel dans la vue Recherche Modèle , sur les résultats d'une recherche SQL exécutée avec succès qui inclut CLASSGUID et CLASSTYPE, ou une recherche construite par requête • Élément - Scripts pouvant être exécutés à partir du sous-menu « Scripts » des menus contextuels des éléments ; accessibles à partir de la fenêtre Navigateur , Diagramme , Recherche Modèle , Liste d'éléments, Paquetage Navigateur et des vues Gantt • Paquetage - Scripts exécutables depuis le sous-menu « Scripts » des menus contextuels Paquetage ; accessibles depuis la fenêtre Navigateur • Diagramme - Scripts exécutables depuis l'option de menu contextuel ' Scripts ' pour diagrammes ; accessible depuis la fenêtre Navigateur et diagrammes • Lien - Scripts pouvant être exécutés à partir de l'option de menu contextuel « Scripts » pour les connecteurs ; accessibles à partir diagrammes
Notes	Type tous les commentaires dont vous avez besoin concernant ce groupe de scripts.

JavaScript Math Library

La légendaire Bibliothèque mathématique Cephès est entièrement et étroitement intégrée au moteur JavaScript disponible dans Enterprise Architect . Cette bibliothèque est une collection de plus de 400 routines mathématiques de haute qualité pour les applications scientifiques et d'ingénierie, offrant un large éventail de potentiel mathématique aux modélisateurs souhaitant faire passer leurs modèles d'ingénierie et de systèmes au niveau supérieur.

La bibliothèque de fonctions implémente la norme double précision IEEE Std 754.

- [Arithmetic and Algebraic](#)
- [Exponential and Trigonometric](#)
- [Exponential integral](#)
- [Gamma functions](#)
- [Error function](#)
- [Bessel functions](#)
- [Hypergeometric](#)
- [Elliptic](#)
- [Probability](#)
- [Miscellaneous](#)
- [Matrix](#)
- [Numerical Integration](#)
- [Complex Arithmetic](#)
- [Complex Exponential and Trigonometric](#)
- [errors](#)

Arithmétique et Algébrique

- [sqrt](#) - racine carrée
- [lsqrt](#) - racine carrée integer
- [cbrt](#) - racine cubique
- [polevl](#), [plevl](#) - évaluer un polynôme
- [chbevl](#) - évaluer la série Chebyshev
- [round](#) - round à valeur integer la plus proche
- [ceil](#) - tronquer vers le haut jusqu'à integer
- [floor](#) - tronquer vers le bas jusqu'à integer
- [fexp](#) - extraire l'exposant
- [ldexp](#) - ajouter integer à l'exposant
- [fabs](#) - valeur absolue
- [signbit](#) - renvoie le bit de signe sous forme int
- [isnan](#) - test numérique
- [isfinite](#) - test fini
- [poladd](#) - additionner des polynômes
- [polsub](#) - soustraire des polynômes
- [polmul](#) - multiplier des polynômes
- [poldiv](#) - diviser des polynômes
- [polsbt](#) - variable polynomiale de substitution
- [poleva](#) - évaluer un polynôme
- [polclr](#) - mettre tous les coefficients à zéro
- [polmov](#) - coefficients de copie

sqrt

Square root.

SYNOPSIS:

```
double x, y, sqrt();  
y = sqrt(x);
```

DESCRIPTION:

Returns the square root of x.

Range reduction involves isolating the power of two of the argument and using a polynomial approximation to obtain a rough value for the square root. Then Heron's iteration is used three times to converge to an accurate value.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 10	60000	2.1e-17	7.9e-18
IEEE	0,1.7e308	30000	1.7e-16	6.3e-17

ERROR MESSAGES:

message	condition	value returned
domain	$x < 0$	0.0

Isqrt

Integer square root.

SYNOPSIS:

```
long x, y;  
long Isqrt();  
y = Isqrt(x);
```

DESCRIPTION:

Returns a long integer square root of the long integer argument. The computation is by binary long division. The largest possible result is $\text{Isqrt}(2,147,483,647) = 46341$.

If $x < 0$, the square root of $|x|$ is returned, and an error message is available.

ACCURACY:

An extra, roundoff, bit is computed; hence the result is the nearest integer to the actual square root.

cbrt

Cube root.

SYNOPSIS:

```
double x, y, cbrt();  
y = cbrt(x);
```

DESCRIPTION:

Returns the cube root of the argument, which could be negative. Range reduction involves determining the power of 2 of the argument. A polynomial of degree 2 applied to the mantissa, and multiplication by the cube root of 1, 2, or 4 approximates the root to within about 0.1%. Then Newton's iteration is used three times to converge to an accurate result.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10,10	200000	1.8e-17	6.2e-18
IEEE	0,1e308	30000	1.5e-16	5.0e-17

JavaScript:

```
//  
//Plot of y = 3vx.  
//  
  
function plotYforX(x1, x2)  
{  
    for(var x = x1; x <= x2; x++)  
    {  
        var y = cephes.cbrt(x);  
        Session.Output("plot of x for " + x + " gives y of " + y);  
    }  
}  
  
function main()  
{  
    plotYforX(-1,6);  
}  
main();
```

polevl , p1evl

Evaluate polynomial.

SYNOPSIS:

```
int N;
double x, y, coef[N+1], polevl[];
y = polevl(x, coef, N);
```

DESCRIPTION:

Evaluates polynomial of degree N:

$$y = C_0 + C_1 x + C_2 x^2 + \dots + C_N x^N$$

Coefficients are stored in reverse order:

$$\text{coef}[0] = C_N, \dots, \text{coef}[N] = C_0.$$

The function p1evl() assumes that coef[N] = 1.0 and is omitted from the array. Its calling arguments are otherwise the same as polevl().

SPEED:

In the interest of speed, there are no checks for out of bounds arithmetic. This routine is used by most of the functions in the library. Depending on available equipment features, the user might want to rewrite the program in microcode or assembly language.

JavaScript:

Example:

```
function stirlingFormula(x)
{
  var STIR = [ 7.87311395793093628397E-4, -2.29549961613378126380E-4,
             -2.68132617805781232825E-3, 3.47222221605458667310E-3,
             8.33333333333482257126E-2 ];
  var SQTPi = 2.50662827463100050242E0;
  var MAXSTIR = 143.01608;
  var w = 1.0 / x;
  var y = cephes.exp(x);
```

```
var w = 1.0 + w * cephes.polevl(w, STIR, 4);
if (x > MAXSTIR) {
    var v = cephes.pow(x, 0.5 * x - 0.25);
    y = v * (v / y);
} else {
    y = cephes.pow(x, x - 0.5) / y;
}
y = SQTPI * y * w;
return y;
}
```

chbevl

Evaluate Chebyshev series.

SYNOPSIS:

```
int N;
double x, y, coef[N], chebevl();
```

```
y = chbevl(x, coef, N);
```

DESCRIPTION:

Evaluates the series

$$y = \sum_{i=0}^{N-1} \text{coef}[i] T_i(x/2)$$

of Chebyshev polynomials T_i at argument $x/2$.

Coefficients are stored in reverse order, i.e. the zero order term is last in the array. Note N is the number of coefficients, not the order.

If coefficients are for the interval a to b , x must have been transformed to $x \rightarrow 2(2x - b - a)/(b - a)$ before entering the routine. This maps x from (a, b) to $(-1, 1)$, over which the Chebyshev polynomials are defined.

If the coefficients are for the inverted interval, in which (a, b) is mapped to $(1/b, 1/a)$, the transformation required is $x \rightarrow 2(2ab/x - b - a)/(b - a)$. If b is infinity, this becomes $x \rightarrow 4a/x - 1$.

SPEED:

Taking advantage of the recurrence properties of the Chebyshev polynomials, the routine requires one more addition per loop than evaluating a nested polynomial of the same degree.

JavaScript:

```
var y = cephes.chbevl(x, coef, N);
```

round

Round double to nearest or even integer valued double

SYNOPSIS:

```
double x, y, round();
```

```
y = round(x);
```

DESCRIPTION:

Returns the nearest integer to x as a double precision floating point result. If x ends in 0.5 exactly, the nearest even integer is chosen.

ACCURACY:

If x is greater than $1/(2*\text{MACHEP})$, its closest machine representation is already an integer, so rounding does not change it.

floor

SYNOPSIS:

```
double floor(x);  
double x,y;  
y = floor(x);
```

DESCRIPTION:

floor() returns the largest integer less than or equal to x. It truncates toward minus infinity.

ceil

SYNOPSIS:

```
double ceil(x);  
double x, y;  
y = ceil(x);
```

DESCRIPTION:

ceil() returns the smallest integer greater than or equal to x. It truncates toward plus infinity.

frexp

Extract exponent.

SYNOPSIS:

```
double frexp(x, expnt);  
double x;  
int expnt;  
y = frexp(x, &expnt);
```

DESCRIPTION:

`frexp()` extracts the exponent from `x`. It returns an integer power of two to `expnt` and the significand between 0.5 and 1 to `y`. Thus $x = y * 2^{expnt}$.

ldexp

SYNOPSIS:

```
double ldexp(x,n);  
double x;  
int n;  
y = ldexp(x, n);
```

DESCRIPTION:

ldexp() multiplies x by $2^{*}n$.

fabs

Absolute value.

SYNOPSIS:

```
double x, y;  
y = fabs(x);
```

DESCRIPTION:

Returns the absolute value of the argument.

signbit

SYNOPSIS:

```
int signbit(x);  
double x;  
int n;  
n = signbit(x);
```

DESCRIPTION:

signbit(x) returns 1 if the sign bit of x is 1, else 0.

isnan

SYNOPSIS:

```
int isnan(x);  
double x;  
int n;
```

```
n = isnan(x);
```

DESCRIPTION:

Returns true if x is not a number.

isfinite

SYNOPSIS:

```
int isfinite();  
double x;  
int n;
```

```
n = isfinite(x);
```

DESCRIPTION:

Return true if x is not infinite and is not a NaN

poladd

Polynomial Addition

SYNOPSIS:

```
int maxpol, na, nb, nc;  
double a[na], b[nb], c[nc];  
  
nc = max(na, nb);  
polini( nc );  
poladd( a, na, b, nb, c );
```

DESCRIPTION:

```
poladd( a, na, b, nb, c ); c = b + a, nc = max(na, nb)
```

In this description a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use `or` generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

```
polini( MAXPOL );
```

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

polsub

Polynomial Subtraction

SYNOPSIS:

```
int maxpol, na, nb, nc;
```

```
double a[], b[], c[];
```

```
nc = max(na, nb);
```

```
polini( nc );
```

```
polsub( a, na, b, nb, c );
```

DESCRIPTION:

```
polsub( a, na, b, nb, c ); c = b - a, nc = max(na, nb)
```

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use `or` generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

```
polini( MAXPOL );
```

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is:

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

polmul

Polynomial Multiplication

SYNOPSIS:

```
int maxpol, na, nb, nc;
```

```
double a[], b[], c[];
```

```
nc = na + nb;
```

```
polini( nc );
```

```
polmul( a, na, b, nb, c );
```

DESCRIPTION:

```
polmul( a, na, b, nb, c ); c = b * a, nc = na + nb
```

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use `or` generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

```
polini( MAXPOL );
```

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

poldiv

Polynomial Division

SYNOPSIS:

```
int maxpol, na, nb, nc;
```

```
double a[], b[], c[];
```

```
nc = na + nb
```

```
polini( MAXPOL );
```

```
i = poldiv( a, na, b, nb, c );
```

DESCRIPTION:

```
i = poldiv( a, na, b, nb, c ); c = b / a, nc = MAXPOL
```

returns i = the degree of the first nonzero coefficient of a.

The computed quotient c must be divided by x^i .

An error message is printed if a is identically zero.

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use `or` generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

```
polini( MAXPOL );
```

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

polsbt

Substitute Polynomial Variable

SYNOPSIS:

```
int a, b;  
double a[na], b[nb], c[nc];  
polsbt( a, na, b, nb, c );
```

DESCRIPTION:

If a and b are polynomials, and $t = a(x)$, then

$$c(t) = b(a(x))$$

is a polynomial found by substituting $a(x)$ for t.

The subroutine call for this is:

```
polsbt( a, na, b, nb, c );
```

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

```
polini( MAXPOL );
```

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

poleva

Polynomial Evaluation

SYNOPSIS:

```
int na;  
double sum, x;  
double a[na];
```

```
sum = poleva( a, na, x );
```

DESCRIPTION:

Evaluate polynomial $a(t)$ at $t = x$.

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

polclr

Clear Polynomial

SYNOPSIS:

```
int na;  
double a[na];  
polclr( a, na );
```

DESCRIPTION:

Set all coefficients of polynomial a to zero, up to a[na].

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

polmov

Move Polynomial

SYNOPSIS:

```
int na;  
double a[na], b[na];  
polmov( a, na, b );
```

DESCRIPTION:

Set $b = a$. Copies coefficients of polynomial a , to b .

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \dots + a[na] * x^{na} .$$

Exponentiel et Trigonométrie

- [acos](#) - Arc cosine
- [acosh](#) - Arc cosine hyperbolique
- [asinh](#) - Arc sine hyperbolique
- [atanh](#) - Arc tangente hyperbolique
- [asin](#) - Arc sinus
- [atan](#) - Arctangente
- [atan2](#) - Arctangente correcte du quadrant
- [cos](#) - Cosinus
- [cosdg](#) - Cosinus de arg en degrés
- [exp](#) - Exponentiel , base e
- [exp2](#) - Exponentiel , base 2
- [exp10](#) - Exponentiel , base 10
- [cosh](#) - cosine hyperbolique
- [sinh](#) - sine hyperbolique
- [tanh](#) - Tangente hyperbolique
- [log](#) - Logarithme, base e
- [log2](#) - Logarithme, base 2
- [log10](#) - Logarithme, base 10
- [pow](#) - Puissance
- [powi](#) - Puissance Integer
- [sin](#) - Sine
- [sindg](#) - Sine de arg en degrés
- [tan](#) - Tangente
- [tandg](#) - Tangente de arg en degrés

acos

Inverse circular cosine.

SYNOPSIS:

```
double x, y, acos();  
y = acos(x);
```

DESCRIPTION:

Returns radian angle between 0 and pi whose cosine is x.

Analytically, $\text{acos}(x) = \pi/2 - \text{asin}(x)$. However if $|x|$ is near 1, there is cancellation error in subtracting $\text{asin}(x)$ from $\pi/2$. Hence if $x < -0.5$, $\text{acos}(x) = \pi - 2.0 * \text{asin}(\text{sqrt}((1+x)/2))$; or if $x > +0.5$, $\text{acos}(x) = 2.0 * \text{asin}(\text{sqrt}((1-x)/2))$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-1, 1	50000	3.3e-17	8.2e-18
IEEE	-1, 1	10 ⁶	2.2e-16	6.5e-17

ERROR MESSAGES:

message	condition	value returned
domain	$ x > 1$	NAN

acosh

Inverse hyperbolic cosine.

SYNOPSIS:

```
double x, y, acosh();  
y = acosh(x);
```

DESCRIPTION:

Returns the inverse hyperbolic cosine of an argument.

If $1 \leq x < 1.5$, a rational approximation:

$$\sqrt{z} * P(z)/Q(z)$$

where $z = x-1$, is used. Otherwise:

$$\operatorname{acosh}(x) = \log(x + \sqrt{(x-1)(x+1)}).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	1,3	30000	4.2e-17	1.1e-17
IEEE	1,3	30000	4.6e-16	8.7e-17

ERROR MESSAGES:

message	condition	value returned
domain	$ x < 1$	NAN

asinh

Inverse hyperbolic sine.

SYNOPSIS:

```
double x, y, asinh();  
y = asinh(x);
```

DESCRIPTION:

Returns the inverse hyperbolic sine of an argument.

If $|x| < 0.5$, the function is approximated by a rational form $x + x**3 P(x)/Q(x)$.

Otherwise, $\text{asinh}(x) = \log(x + \text{sqrt}(1 + x*x))$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-3,3	75000	4.6e-17	1.1e-17
IEEE	-1,1	30000	3.7e-16	7.8e-17
IEEE	1,3	30000	2.5e-16	6.7e-17

atanh

Inverse hyperbolic tangent.

SYNOPSIS:

```
double x, y, atanh();  
y = atanh(x);
```

DESCRIPTION:

Returns the inverse hyperbolic tangent of an argument in the range MINLOG to MAXLOG.

If $|x| < 0.5$, the rational form $x + x^3 P(x)/Q(x)$ is employed. Otherwise:

$$\operatorname{atanh}(x) = 0.5 * \log((1+x)/(1-x)).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-1,1	50000	2.4e-17	6.4e-18
IEEE	-1,1	30000	1.9e-16	5.2e-17

asin

Inverse circular sine.

SYNOPSIS:

```
double x, y, asin();  
y = asin(x);
```

DESCRIPTION:

Returns the radian angle between $-\pi/2$ and $+\pi/2$ whose sine is x .

A rational function of the form $x + x^3 P(x^2)/Q(x^2)$ is used for $|x|$ in the interval $[0, 0.5]$. If $|x| > 0.5$ it is transformed by the identity:

$$\text{asin}(x) = \pi/2 - 2 \text{asin}(\sqrt{(1-x)/2}).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-1, 1	40000	2.6e-17	7.1e-18
IEEE	-1, 1	10 ⁶	1.9e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
domain	$ x > 1$	NAN

atan

Inverse circular tangent (arctangent).

SYNOPSIS:

```
double x, y, atan();  
y = atan(x);
```

DESCRIPTION:

Returns the radian angle between $-\pi/2$ and $+\pi/2$ whose tangent is x .

Range reduction is from three intervals into the interval from zero to 0.66. The approximant uses a rational function of degree 4/5 of the form $x + x^3 P(x)/Q(x)$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10, 10	50000	2.4e-17	8.3e-18
IEEE	-10, 10	10 ⁶	1.8e-16	5.0e-17

atan2

Quadrant correct inverse circular tangent.

SYNOPSIS:

```
double x, y, z, atan2();  
z = atan2(y, x);
```

DESCRIPTION:

Returns the radian angle whose tangent is y/x .

Define compile time symbol ANSIC = 1 for ANSI standard, range $-\pi < z \leq +\pi$, args (y,x);

else ANSIC = 0 for range 0 to 2π , args (x,y).

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-10, 10	10^6	$2.5e-16$	$6.9e-17$

COS

Circular cosine.

SYNOPSIS:

```
double x, y, cos();
y = cos(x);
```

DESCRIPTION:

Range reduction is into intervals of $\pi/4$. The reduction error is nearly eliminated by contriving an extended precision modular arithmetic.

Two polynomial approximating functions are employed.

Between 0 and $\pi/4$ the cosine is approximated by:

$$1 - x^2 Q(x^2).$$

Between $\pi/4$ and $\pi/2$ the sine is represented as:

$$x + x^3 P(x^2).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-1.07e9,+1.07e9	130000	2.1e-16	5.4e-17
DEC	0,+1.07e9	17000	3.0e-17	7.2e-18

cosdg

Circular cosine of angle in degrees.

SYNOPSIS:

```
double x, y, cosdg();  
y = cosdg(x);
```

DESCRIPTION:

Range reduction is into intervals of 45 degrees. Two polynomial approximating functions are employed.

Between 0 and $\pi/4$ the cosine is approximated by:

$$1 - x^{**2} P(x^{**2}).$$

Between $\pi/4$ and $\pi/2$ the sine is represented as:

$$x + x^{**3} P(x^{**2}).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+/-1000	3400	3.5e-17	9.1e-18
IEEE	+/-1000	30000	2.1e-16	5.7e-17

exp

Exponential function.

SYNOPSIS:

```
double x, y, exp();
y = exp(x);
```

DESCRIPTION:

Returns e (2.71828...) raised to the x power.

Range reduction is accomplished by separating the argument into an integer k and fraction f such that:

$$x = k + f$$

$$e^x = 2^k e^f$$

A Pade' form

$1 + 2x P(x^2)/(Q(x^2) - P(x^2))$ of degree 2/3 is used to approximate $\exp(f)$ in the basic interval $[-0.5, 0.5]$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+ - 88	50000	2.8e-17	7.0e-18
IEEE	+ - 708	40000	2.0e-16	5.6e-17

Error amplification in the exponential function can be a serious matter. The error propagation involves:

$$\exp(X(1+\delta)) = \exp(X) (1 + X\delta + \dots)$$

This shows that a 1 lsb error in representing X produces a relative error of X times 1 lsb in the function. While the routine gives an accurate result for arguments that are exactly represented by a double precision computer number, the result contains an amplified roundoff error for large arguments not exactly represented.

ERROR MESSAGES:

message	condition	value returned
underflow	x < MINLOG	0.0
overflow	x > MAXLOG	INFINITY

exp2

Base 2 exponential function.

SYNOPSIS:

```
double x, y, exp2();
y = exp2(x);
```

DESCRIPTION:

Returns 2 raised to the x power.

Range reduction is accomplished by separating the argument into an integer k and fraction f, such that:

$$x = k + f$$

$$2^x = 2^k \cdot 2^f$$

A Pade' form:

$$1 + 2x P(x^{**2}) / (Q(x^{**2}) - x P(x^{**2}))$$

approximates 2^{**x} in the basic range [-0.5, 0.5].

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-1022,+1024	30000	1.8e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
underflow	$x < -\text{MAXL2}$	0.0
overflow	$x > \text{MAXL2}$	MAXNUM

For DEC arithmetic, MAXL2 = 127.

For IEEE arithmetic, MAXL2 = 1024.

exp10

Base 10 exponential function. (Common antilogarithm.)

SYNOPSIS:

```
double x, y, exp10();
y = exp10(x);
```

DESCRIPTION:

Returns 10 raised to the x power.

Range reduction is accomplished by expressing the argument as $10^{**x} = 2^{**n} 10^{**f}$, with $|f| < 0.5 \log_{10}(2)$.

The Pade' form:

$$1 + 2x P(x^{**2}) / (Q(x^{**2}) - P(x^{**2}))$$

is used to approximate 10^{**f} .

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-307,+307	30000	2.2e-16	5.5e-17

Test result from an earlier version (2.1):

DEC	-38,+38	70000	3.1e-17	7.0e-18
-----	---------	-------	---------	---------

ERROR MESSAGES:

message	condition	value returned
underflow	$x < -\text{MAXL10}$	0.0
overflow	$x > \text{MAXL10}$	MAXNUM

DEC arithmetic: MAXL10 = 38.230809449325611792.

IEEE arithmetic: MAXL10 = 308.2547155599167.

cosh

Hyperbolic cosine.

SYNOPSIS:

```
double x, y, cosh();  
y = cosh(x);
```

DESCRIPTION:

Returns the hyperbolic cosine of an argument in the range MINLOG to MAXLOG.

$$\cosh(x) = (\exp(x) + \exp(-x))/2.$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+ - 88	50000	4.0e-17	7.7e-18
IEEE	+ - MAXLOG	30000	2.6e-16	5.7e-17

ERROR MESSAGES:

message	condition	value returned
overflow	x > MAXLOG	MAXNUM

sinh

Hyperbolic sine.

SYNOPSIS:

```
double x, y, sinh();  
y = sinh(x);
```

DESCRIPTION:

Returns the hyperbolic sine of an argument in the range MINLOG to MAXLOG.

The range is partitioned into two segments. If $|x| \leq 1$, a rational function of the form $x + x^3 P(x)/Q(x)$ is employed. Otherwise the calculation is $\sinh(x) = (\exp(x) - \exp(-x))/2$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+ - 88	50000	4.0e-17	7.7e-18
IEEE	+ - MAXLOG	30000	2.6e-16	5.7e-17

tanh

Hyperbolic tangent.

SYNOPSIS:

```
double x, y, tanh();  
y = tanh(x);
```

DESCRIPTION:

Returns the hyperbolic tangent of an argument in the range MINLOG to MAXLOG.

A rational function is used for $|x| < 0.625$. The form:

$x + x**3 P(x)/Q(x)$ of Cody_ & Waite

is employed.

Otherwise:

$$\tanh(x) = \sinh(x)/\cosh(x) = 1 - 2/(\exp(2x) + 1).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-2,2	50000	3.3e-17	6.4e-18
IEEE	-2,2	30000	2.5e-16	5.8e-17

log

Un algorithme naturel.

SYNOPSIS:

```
double x, y, log ();  
y = log (x);
```

LA DESCRIPTION:

Renvoie la base e (2,718...) logarithme de x.

L'argument est séparé en ses parties exposant et fractionnaire. Si l'exposant est compris entre -1 et +1, le logarithme de la fraction est approximé par :

$$\log (1+x) = x - 0,5 x^{**2} + x^{**3} P(x)/Q(x).$$

Sinon, en posant $z = 2(x-1)/x+1$,

$$\log (x) = z + z^{**3} P(z)/Q(z).$$

PRÉCISION:

Erreur relative:

domaine arithmétique nombre d'essais crête rms

IEEE 0.5, 2.0 150000 1.44e-16 5.06e-17

IEEE +-MAXNUM 30000 1.20e-16 4.78e-17

DEC 0, 10 170000 1.8e-17 6.3e-18

Dans les tests sur l'intervalle [+MAXNUM], les logarithmes des arguments aléatoires étaient uniformément répartis sur [0,MAXLOG].

MESSAGES D'ERREUR:

singularité : $x = 0$; renvoie -INFINI

domaine : $x < 0$; renvoie NAN

log2

Base 2 logarithm.

SYNOPSIS:

```
double x, y, log2();
y = log2(x);
```

DESCRIPTION:

Returns the base 2 logarithm of x.

The argument is separated into its exponent and fractional parts. If the exponent is between -1 and +1, the base e logarithm of the fraction is approximated by:

$$\log(1+x) = x - 0.5 x^{**2} + x^{**3} P(x)/Q(x).$$

Otherwise, setting $z = 2(x-1)/(x+1)$,

$$\log(x) = z + z^{**3} P(z)/Q(z).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0.5, 2.0	30000	2.0e-16	5.5e-17
IEEE	exp(+/-700)	40000	1.3e-16	4.6e-17

In the tests over the interval $[\exp(+/-700)]$, the logarithms of the random arguments were uniformly distributed.

ERROR MESSAGES:

singularity: $x = 0$; returns -INFINITY

domain: $x < 0$; returns NAN

log10

Common logarithm.

SYNOPSIS:

```
double x, y, log10();  
y = log10(x);
```

DESCRIPTION:

Returns logarithm to the base 10 of x.

The argument is separated into its exponent and fractional parts. The logarithm of the fraction is approximated by:

$$\log(1+x) = x - 0.5 x^{**2} + x^{**3} P(x)/Q(x).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0.5, 2.0	30000	1.5e-16	5.0e-17
IEEE	0, MAXNUM	30000	1.4e-16	4.8e-17
DEC	1, MAXNUM	50000	2.5e-17	6.0e-18

In the tests over the interval [1, MAXNUM], the logarithms of the random arguments were uniformly distributed over [0, MAXLOG].

ERROR MESSAGES:

singularity: x = 0; returns -INFINITY

domain: x < 0; returns NAN

pow

Power function

SYNOPSIS:

```
double x, y, z, pow();  
z = pow(x, y);
```

DESCRIPTION:

Computes x raised to the yth power. Analytically:

$$x^{**}y = \exp(y \log(x)).$$

Following Cody and Waite, this program uses a lookup table of $2^{**}-i/16$ and pseudo extended precision arithmetic to obtain an extra three bits of accuracy in both the logarithm and the exponential.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-26,26	30000	4.2e-16	7.7e-17
DEC	-26,26	60000	4.8e-17	9.1e-18

$1/26 < x < 26$, with $\log(x)$ uniformly distributed.

$-26 < y < 26$, y uniformly distributed.

IEEE	0,8700	30000	1.5e-14	2.1e-15
------	--------	-------	---------	---------

$0.99 < x < 1.01$, $0 < y < 8700$, uniformly distributed.

ERROR MESSAGES:

message	condition	value returned
overflow	$x^{**}y > \text{MAXNUM}$	INFINITY
underflow	$x^{**}y < 1/\text{MAXNUM}$	0.0
domain	$x < 0$ and y noninteger	0.0

powi

Real raised to integer power.

SYNOPSIS:

```
double x, y, powi();
```

```
int n;
```

```
y = powi(x, n);
```

DESCRIPTION:

Returns an argument x raised to the n th power. The routine efficiently decomposes n as a sum of powers of two. The desired power is a product of two-to-the- k th powers of x . Thus to compute the 32767 power of x requires 28 multiplications instead of 32767 multiplications.

ACCURACY:

Relative error:

arithmetic	x domain	n domain	# trials	peak	rms
DEC	.04,26	-26,26	100000	2.7e-16	4.3e-17
IEEE	.04,26	-26,26	50000	2.0e-15	3.8e-16
IEEE	1,2	-1022,1023	50000	8.6e-14	1.6e-14

Returns MAXNUM on overflow, zero on underflow.

sin

Circular sine.

SYNOPSIS:

```
double x, y, sin();
y = sin(x);
```

DESCRIPTION:

Range reduction is into intervals of $\pi/4$. The reduction error is nearly eliminated by contriving an extended precision modular arithmetic.

Two polynomial approximating functions are employed.

Between 0 and $\pi/4$ the sine is approximated by:

$$x + x^{*3} P(x^{*2}).$$

Between $\pi/4$ and $\pi/2$ the cosine is represented as:

$$1 - x^{*2} Q(x^{*2}).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 10	150000	3.0e-17	7.8e-18
IEEE	-1.07e9,+1.07e9	130000	2.1e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
total loss	$x > 1.073741824e9$	0.0

Partial loss of accuracy begins to occur at $x = 2^{*30} = 1.074e9$. The loss is not gradual, but jumps suddenly to about 1 part in $10e7$. Results might be meaningless for $x > 2^{*49} = 5.6e14$. The routine as implemented flags a TLOSS error for $x > 2^{*30}$ and returns 0.0.

sindg

Circular sine of an angle in degrees.

SYNOPSIS:

```
double x, y, sindg();  
y = sindg(x);
```

DESCRIPTION:

Range reduction is into intervals of 45 degrees. Two polynomial approximating functions are employed.

Between 0 and $\pi/4$ the sine is approximated by:

$$x + x^{*3} P(x^{*2}).$$

Between $\pi/4$ and $\pi/2$ the cosine is represented as:

$$1 - x^{*2} P(x^{*2}).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+/-1000	3100	3.3e-17	9.0e-18
IEEE	+/-1000	30000	2.3e-16	5.6e-17

ERROR MESSAGES:

message	condition	value returned
total loss	$x > 8.0e14$ (DEC)	0.0
	$x > 1.0e14$ (IEEE)	

tan

Circular tangent.

SYNOPSIS:

```
double x, y, tan();  
y = tan(x);
```

DESCRIPTION:

Returns the circular tangent of the radian argument x .

Range reduction is modulo $\pi/4$.

A rational function:

$$x + x^{*3} P(x^{*2})/Q(x^{*2})$$

is employed in the basic interval $[0, \pi/4]$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	+/-1.07e9	44000	4.1e-17	1.0e-17
IEEE	+/-1.07e9	30000	2.9e-16	8.1e-17

ERROR MESSAGES:

message	condition	value returned
total loss	$x > 1.073741824e9$	0.0

tandg

Circular tangent of argument in degrees.

SYNOPSIS:

```
double x, y, tandg();
y = tandg(x);
```

DESCRIPTION:

Returns the circular tangent of the argument x in degrees.

Range reduction is modulo $\pi/4$. A rational function:

$$x + x^{**3} P(x^{**2})/Q(x^{**2})$$

is employed in the basic interval $[0, \pi/4]$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,10	8000	3.4e-17	1.2e-17
IEEE	0,10	30000	3.2e-16	8.4e-17

ERROR MESSAGES:

message	condition	value returned
total loss	$x > 8.0e14$ (DEC) $x > 1.0e14$ (IEEE)	0.0
singularity	$x = 180\text{ k} + 90$	MAXNUM

Intégrale Exponentielle

- [expn](#) - Intégrale Exponentielle
- [shichi](#) - Intégrales hyperboliques sine et cosine
- [sici](#) - Intégrales Sine et cosine_

expn

Exponential integral En.

SYNOPSIS:

```
int n;
double x, y, expn();
y = expn(n, x);
```

DESCRIPTION:

Evaluates the exponential integral.

$$E(x) = \int_0^{\infty} \frac{e^{-xt}}{1+nt} dt.$$

Both n and x must be nonnegative.

The routine employs either a power series, a continued fraction, or an asymptotic formula depending on the relative values of n and x.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	0, 30	5000	2.0e-16	4.6e-17
IEEE	0, 30	10000	1.7e-15	3.6e-16

shichi

Hyperbolic sine and cosine integrals.

SYNOPSIS:

```
double x, Chi, Shi, shichi();
shichi(x, &Chi, &Shi);
```

DESCRIPTION:

Approximates the integrals

$$\text{Chi}(x) = \text{eul} + \ln x + \int_0^x \frac{\cosh t - 1}{t} dt,$$

$$\text{Shi}(x) = \int_0^x \frac{\sinh t}{t} dt$$

where $\text{eul} = 0.57721566490153286061$ is Euler's constant. The integrals are evaluated by power series for $x < 8$ and by Chebyshev expansions for x between 8 and 88. For large x , both functions approach $\exp(x)/2x$. Arguments greater than 88 in magnitude return MAXNUM.

ACCURACY:

Test interval 0 to 88.

Relative error:				
arithmetic	function	# trials	peak	rms
DEC	Shi	3000	9.1e-17	
IEEE	Shi	30000	6.9e-16	1.6e-16

Absolute error, except relative when $|\text{Chi}| > 1$:

DEC	Chi	2500	9.3e-17	
IEEE	Chi	30000	8.4e-16	1.4e-16

sici

Sine and cosine integrals.

SYNOPSIS:

```
double x, Ci, Si, sici();
sici(x, &Si, &Ci);
```

DESCRIPTION:

Evaluates the integrals:

$$Ci(x) = \text{eul} + \ln x + \int_0^x \frac{\cos t - 1}{t} dt,$$

$$Si(x) = \int_0^x \frac{\sin t}{t} dt$$

where $\text{eul} = 0.57721566490153286061$ is Euler's constant. The integrals are approximated by rational functions. For $x > 8$ auxiliary functions $f(x)$ and $g(x)$ are employed such that

$$Ci(x) = f(x) \sin(x) - g(x) \cos(x)$$

$$Si(x) = \pi/2 - f(x) \cos(x) - g(x) \sin(x)$$

ACCURACY:

Test interval = [0,50].

Absolute error, except relative when > 1 :

arithmetic	function	# trials	peak	rms
------------	----------	----------	------	-----

IEEE	Si	30000	4.4e-16	7.3e-17
IEEE	Ci	30000	6.9e-16	5.1e-17
DEC	Si	5000	4.4e-17	9.0e-18
DEC	Ci	5300	7.9e-17	5.2e-18

Fonctions gamma

- [beta](#) - beta
- [lbeta](#) - log naturel de beta
- [fac](#) - factorielle
- [gamma](#) - gamma
- [lgam](#) - logarithme de la fonction gamma
- [incbet](#) - intégrale beta incomplète
- [incbi](#) - inverse de l'intégrale beta incomplète
- [igam](#) - intégrale gamma incomplète
- [igamc](#) - intégrale gamma complétée
- [igami](#) - intégrale gamma inverse
- [psi](#) - Fonction psi (digamma)
- [rgamma](#) - Gamma réciproque

beta

Beta function.

SYNOPSIS:

```
double a, b, y, beta();
y = beta(a, b);
```

DESCRIPTION:

$$\text{beta}(a, b) = \frac{\Gamma(a) \Gamma(b)}{\Gamma(a+b)}$$

For large arguments the logarithm of the function is evaluated using `lgam()`, then exponentiated.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	0,30	1700	7.7e-15	1.5e-15
IEEE	0,30	30000	8.1e-14	1.1e-14

ERROR MESSAGES:

message	condition	value returned
overflow	$\log(\text{beta}) > \text{MAXLOG}$	0.0
	$a \text{ or } b < 0 \text{ integer}$	0.0

lbeta

Natural log of |beta|.

Return the sign of beta in sngam.

fac

Factorial function.

SYNOPSIS:

```
double y, fac();
int i;
y = fac(i);
```

DESCRIPTION:

Returns factorial of $i = 1 * 2 * 3 * \dots * i$.

$\text{fac}(0) = 1.0$.

Due to machine arithmetic bounds the largest value of i accepted is 33 in DEC arithmetic or 170 in IEEE arithmetic. Greater values, or negative ones, produce an error message and return MAXNUM.

ACCURACY:

For $i < 34$ the values are simply tabulated, and have full machine accuracy. If $i > 55$, $\text{fac}(i) = \text{gamma}(i+1)$;

Relative error:

arithmetic	domain	peak
IEEE	0, 170	1.4e-15
DEC	0, 33	1.4e-17

gamma

Gamma function.

SYNOPSIS:

```
double x, y, gamma();  
y = gamma(x);
```

DESCRIPTION:

Returns the gamma function of the argument. The result is correctly signed, and the sign (+1 or -1) is also returned in a global (extern) variable named `sgngam`. This variable is also filled in by the logarithmic gamma function `lgam()`.

Arguments $|x| \leq 34$ are reduced by recurrence and the function approximated by a rational function of degree 6/7 in the interval (2,3). Large arguments are handled by Stirling's formula. Large negative arguments are made positive using a reflection formula.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-34, 34	10000	1.3e-16	2.5e-17
IEEE	-170,-33	20000	2.3e-15	3.3e-16
IEEE	-33, 33	20000	9.4e-16	2.2e-16
IEEE	33, 171.6	20000	2.3e-15	3.2e-16

Error for arguments outside the test range will be larger owing to error amplification by the exponential function.

lgam

Natural logarithm of gamma function.

SYNOPSIS:

```
double x, y, lgam();
y = lgam(x);
```

DESCRIPTION:

Returns the base e (2.718...) logarithm of the absolute value of the gamma function of the argument. The sign (+1 or -1) of the gamma function is returned in a global (extern) variable named sgngam.

For arguments greater than 13, the logarithm of the gamma function is approximated by the logarithmic version of Stirling's formula using a polynomial approximation of degree 4. Arguments between -33 and +33 are reduced by recurrence to the interval [2,3] of a rational approximation. The cosecant reflection formula is employed for arguments less than -33.

Arguments greater than MAXLGM return MAXNUM and an error message.

MAXLGM = 2.035093e36 for DEC arithmetic or 2.556348e305 for IEEE arithmetic.

ACCURACY:

arithmetic	domain	# trials	peak	rms
DEC	0, 3	7000	5.2e-17	1.3e-17
DEC	2.718, 2.035e36	5000	3.9e-17	9.9e-18
IEEE	0, 3	28000	5.4e-16	1.1e-16
IEEE	2.718, 2.556e305	40000	3.5e-16	8.3e-17

The error criterion was relative when the function magnitude was greater than one but absolute when it was less than one.

This test used the relative error criterion, though at certain points the relative error could be much higher than indicated.

IEEE	-200, -4	10000	4.8e-16	1.3e-16
------	----------	-------	---------	---------

incbet

Incomplete beta integral.

SYNOPSIS:

```
double a, b, x, y, incbet();
y = incbet(a, b, x);
```

DESCRIPTION:

Returns the incomplete beta integral of the arguments, evaluated from zero to x. The function is defined as:

$$\frac{\int_0^x t^{a-1} (1-t)^{b-1} dt}{\int_0^1 t^{a-1} (1-t)^{b-1} dt}$$

The domain of definition is $0 \leq x \leq 1$. In this implementation a and b are restricted to positive values. The integral from x to 1 can be obtained by the symmetry relation:

$$1 - \text{incbet}(a, b, x) = \text{incbet}(b, a, 1-x).$$

The integral is evaluated by a continued fraction expansion or, when $b*x$ is small, by a power series.

ACCURACY:

Tested at uniformly distributed random points (a,b,x) with a and b in "domain" and x between 0 and 1.

arithmetic	domain	# trials	Relative error	
			peak	rms
IEEE	0,5	10000	6.9e-15	4.5e-16
IEEE	0,85	250000	2.2e-13	1.7e-14
IEEE	0,1000	30000	5.3e-12	6.3e-13
IEEE	0,10000	250000	9.3e-11	7.1e-12
IEEE	0,100000	10000	8.7e-10	4.8e-11

Outputs smaller than the IEEE gradual underflow threshold were excluded from these statistics.

ERROR MESSAGES:

message	condition	value returned
domain	$x < 0, x > 1$	0.0

underflow

0.0

incbi

Inverse of incomplete beta integral.

SYNOPSIS:

```
double a, b, x, y, incbi();
x = incbi(a, b, y);
```

DESCRIPTION:

Given y , the function finds x such that:

$$\text{incbet}(a, b, x) = y .$$

The routine performs interval halving or Newton iterations to find the root of $\text{incbet}(a,b,x) - y = 0$.

ACCURACY:

Relative error:

x a,b

arithmetic	domain	domain	# trials	peak	rms
IEEE	0,1	.5,10000	50000	5.8e-12	1.3e-13
IEEE	0,1	.25,100	100000	1.8e-13	3.9e-15
IEEE	0,1	0,5	50000	1.1e-12	5.5e-15
VAX	0,1	.5,100	25000	3.5e-14	1.1e-15

With a and b constrained to half-integer or integer values:

IEEE	0,1	.5,10000	50000	5.8e-12	1.1e-13
IEEE	0,1	.5,100	100000	1.7e-14	7.9e-16

With a = .5, b constrained to half-integer or integer values:

IEEE	0,1	.5,10000	10000	8.3e-11	1.0e-11
------	-----	----------	-------	---------	---------

igam

Incomplete gamma integral.

SYNOPSIS:

```
double a, x, y, igam();
y = igam(a, x);
```

DESCRIPTION:

The function is defined by

$$\text{igam}(a,x) = \frac{\int_0^x t^{a-1} e^{-t} dt}{\int_0^\infty t^{a-1} e^{-t} dt}$$

In this implementation both arguments must be positive. The integral is evaluated by either a power series or continued fraction expansion, depending on the relative values of a and x.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0,30	200000	3.6e-14	2.9e-15
IEEE	0,100	300000	9.9e-14	1.5e-14

igamc

Complemented incomplete gamma integral.

SYNOPSIS:

```
double a, x, y, igamc();
y = igamc(a, x);
```

DESCRIPTION:

The function is defined by

$$\text{igamc}(a,x) = 1 - \text{igam}(a,x)$$

$$\text{igam}(a,x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt$$

In this implementation both arguments must be positive. The integral is evaluated by either a power series or continued fraction expansion, depending on the relative values of a and x .

ACCURACY:

Tested at random a, x .

	a	x	Relative error:		
			domain	# trials	peak
IEEE arithmetic	0.5,100	0,100	200000	1.9e-14	1.7e-15
IEEE	0.01,0.5	0,100	200000	1.4e-13	1.6e-15

igami

Inverse of complemented incomplete gamma integral.

SYNOPSIS:

```
double a, x, p, igami();
x = igami(a, p);
```

DESCRIPTION:

Given p , the function finds x such that

$$\text{igamc}(a, x) = p.$$

Starting with the approximate value

$$x = a t^3$$

where

$$t = 1 - d - \text{ndtri}(p) \sqrt{d}$$

and

$$d = 1/9a,$$

the routine performs up to 10 Newton iterations to find the root of $\text{igamc}(a,x) - p = 0$.

ACCURACY:

Tested at random a, p in the intervals indicated.

	a p		Relative error:		
	arithmetic domain	domain	# trials	peak	rms
IEEE	0.5,100	0,0.5	100000	1.0e-14	1.7e-15
IEEE	0.01,0.5	0,0.5	100000	9.0e-14	3.4e-15
IEEE	0.5,10000	0,0.5	20000	2.3e-13	3.8e-14

psi

Psi (digamma) function.

SYNOPSIS:

```
double x, y, psi();
y = psi(x);
```

DESCRIPTION:

$$\psi(x) = \frac{d}{dx} \ln \Gamma(x)$$

is the logarithmic derivative of the gamma function.

For integer x:

$$\psi(n) = -\text{EUL} + \sum_{k=1}^{n-1} \frac{1}{k}$$

This formula is used for $0 < n \leq 10$. If x is negative, it is transformed to a positive argument by the reflection formula $\psi(1-x) = \psi(x) + \pi \cot(\pi x)$. For general positive x, the argument is made greater than 10 using the recurrence $\psi(x+1) = \psi(x) + 1/x$. Then this asymptotic expansion is applied:

$$\psi(x) = \log(x) - \frac{1}{2x} - \sum_{k=1}^{\infty} \frac{B_{2k}}{2k x^{2k}}$$

where the B_{2k} are Bernoulli numbers.

ACCURACY:

Relative error (except absolute when $|\psi| < 1$):

arithmetic	domain	# trials	peak	rms
DEC	0,30	2500	1.7e-16	2.0e-17
IEEE	0,30	30000	1.3e-15	1.4e-16

IEEE -30,0 40000 1.5e-15 2.2e-16

ERROR MESSAGES:

message	condition	value returned
singularity	x integer <=0	MAXNUM

rgamma

Reciprocal gamma function.

SYNOPSIS:

```
double x, y, rgamma();  
y = rgamma(x);
```

DESCRIPTION:

Returns one divided by the gamma function of the argument.

The function is approximated by a Chebyshev expansion in the interval [0,1]. Range reduction is by recurrence for arguments between -34.034 and +34.84425627277176174. 1/MAXNUM is returned for positive arguments outside this range. For arguments less than -34.034 the cosecant reflection formula is applied; logarithms are employed to avoid unnecessary overflow.

The reciprocal gamma function has no singularities, but overflow and underflow could occur for large arguments. These conditions return either MAXNUM or 1/MAXNUM with the appropriate sign.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-30,+30	4000	1.2e-16	1.8e-17
IEEE	-30,+30	30000	1.1e-15	2.0e-16

For arguments less than -34.034 the peak error is in the order of 5e-15 (DEC), excepting overflow or underflow.

Fonction d'erreur

- [erf](#) - Fonction d'erreur
- [erfc](#) - Fonction d'erreur complétée
- [dawsn](#) - Intégrale de Dawson
- [fresnl](#) - Intégrale de Fresnel

erf

Error function.

SYNOPSIS:

```
double x, y, erf();
y = erf(x);
```

DESCRIPTION:

The integral is

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt.$$

The magnitude of x is limited to 9.231948545 for DEC arithmetic; 1 or -1 is returned outside this range.

For $0 \leq |x| < 1$, $\text{erf}(x) = x * P4(x**2)/Q5(x**2)$; otherwise $\text{erf}(x) = 1 - \text{erfc}(x)$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,1	14000	4.7e-17	1.5e-17
IEEE	0,1	30000	3.7e-16	1.0e-16

erfc

Complementary error function.

SYNOPSIS:

```
double x, y, erfc();
y = erfc(x);
```

DESCRIPTION:

$$1 - \operatorname{erf}(x) = \operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp(-t^2) dt$$

For small x , $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$; otherwise rational approximations are computed.

A special function `exp2.c` is used to suppress error amplification in computing $\exp(-x^2)$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0,26.6417	30000	1.3e-15	2.2e-16

ERROR MESSAGES:

message	condition	value returned
underflow	$x > 9.231948545$ (DEC)	0.0

dawson

Dawson's Integral.

SYNOPSIS:

```
double x, y, dawson();
y = dawson(x);
```

DESCRIPTION:

Approximates the integral

$$\text{dawson}(x) = \frac{\exp(-x^2)}{\sqrt{\pi}} \int_0^x \exp(t^2) dt$$

Three different rational approximations are employed, for the intervals 0 to 3.25; 3.25 to 6.25; and 6.25 up.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	0,10	10000	6.9e-16	1.0e-16
DEC	0,10	6000	7.4e-17	1.4e-17

fresnl

Fresnel integral.

SYNOPSIS:

```
double x, S, C;
void fresnl();
fresnl(x, _&S, _&C);
```

DESCRIPTION:

Evaluates the Fresnel integrals

$$C(x) = \int_0^x \cos(\pi/2 t^2) dt,$$

$$S(x) = \int_0^x \sin(\pi/2 t^2) dt.$$

The integrals are evaluated by a power series for $x < 1$. For $x \geq 1$ auxiliary functions $f(x)$ and $g(x)$ are employed such that:

$$C(x) = 0.5 + f(x) \sin(\pi/2 x^2) - g(x) \cos(\pi/2 x^2)$$

$$S(x) = 0.5 - f(x) \cos(\pi/2 x^2) - g(x) \sin(\pi/2 x^2)$$

ACCURACY:

Relative error.

Arithmetic	function	domain	# trials	peak	rms
IEEE	S(x)	0, 10	10000	2.0e-15	3.2e-16
IEEE	C(x)	0, 10	10000	1.8e-15	3.3e-16
DEC	S(x)	0, 10	6000	2.2e-16	3.9e-17
DEC	C(x)	0, 10	5000	2.3e-16	3.9e-17

JavaScript:

```
var x= 2.5625;  
var r = cephes.fresnl(x);  
Session,Output(r.result);  
Session,Output(r.ssa);  
Session,Output(r.csa);
```

Return value: Object

Format: JSON

```
{  
  "result" : int,  
  "ssa" : double,  
  "cca" : double  
}
```

Fonctions Bessel

- [airy](#) - Fonction aérée
- [j0](#) - Bessel , ordre 0
- [j1](#) - Bessel , ordre 1
- [jn](#) - Bessel , ordre n
- [jv](#) - Bessel , ordre non entier
- [y0](#) - Bessel , deuxième espèce, ordre 0
- [y1](#) - Bessel , deuxième espèce, ordre 1
- [yn](#) - Bessel , deuxième espèce, ordre n
- [yv](#) - Bessel , ordre non entier
- [i0](#) - Bessel modifié, ordre 0
- [i0e](#) - i0 mis à l'échelle de manière exponentielle
- [i1](#) - Bessel modifié, ordre 1
- [i1e](#) - i1 mis à l'échelle exponentiellement
- [iv](#) - Bessel modifié, ordre non int.
- [k0](#) - Bessel modifié, 3e type, ordre 0
- [k0e](#) - k0 mis à l'échelle exponentiellement
- [k1](#) - Bessel modifié, 3e espèce, ordre 1
- [k1e](#) - k1 mis à l'échelle exponentiellement
- [kn](#) - Bessel modifié, 3e espèce, ordre n_

airy

Airy function.

SYNOPSIS:

```
double x, ai, aip, bi, bip;  
int airy();  
airy(x, _&ai, _&aip, _&bi, _&bip);
```

DESCRIPTION:

Solution of the differential equation:

$$y''(x) = xy.$$

The function returns the two independent solutions Ai, Bi and their first derivatives Ai'(x), Bi'(x).

Evaluation is by power series summation for small x, by rational minimax approximations for large x.

ACCURACY:

Error criterion is absolute when function ≤ 1 , relative when function > 1 , except * denotes relative error criterion.

For large negative x, the absolute error increases as $x^{1.5}$.

For large positive x, the relative error increases as $x^{1.5}$.

Arithmetic	domain	function	# trials	peak	rms
IEEE	-10, 0	Ai	10000	1.6e-15	2.7e-16
IEEE	0, 10	Ai	10000	2.3e-14*	1.8e-15*
IEEE	-10, 0	Ai'	10000	4.6e-15	7.6e-16
IEEE	0, 10	Ai'	10000	1.8e-14*	1.5e-15*
IEEE	-10, 10	Bi	30000	4.2e-15	5.3e-16
IEEE	-10, 10	Bi'	30000	4.9e-15	7.3e-16
DEC	-10, 0	Ai	5000	1.7e-16	2.8e-17
DEC	0, 10	Ai	5000	2.1e-15*	1.7e-16*
DEC	-10, 0	Ai'	5000	4.7e-16	7.8e-17
DEC	0, 10	Ai'	12000	1.8e-15*	1.5e-16*
DEC	-10, 10	Bi	10000	5.5e-16	6.8e-17
DEC	-10, 10	Bi'	7000	5.3e-16	8.7e-17

JavaScript:

```
var x = 9.50313909;  
var a = cephes.airy(x);
```

Return value: Object

Format: JSON

```
{  
  "result" : integer,  
  "ai" : double,  
  "aip" : double.  
  "bi" : double,  
  "bip" : double  
}
```

j0

Bessel function of order zero.

SYNOPSIS:

```
double x, y, j0();
y = j0(x);
```

DESCRIPTION:

Returns a Bessel function of order zero of the argument. The domain is divided into the intervals [0, 5] and (5, infinity). In the first interval this rational approximation is used:

$$(w - r_1)^2 (w - r_2)^2 P(w) / Q(w)$$

1 2 3 8

2

where $w = x$ and each r is a zero of the function.

In the second interval, the Hankel asymptotic expansion is employed with two rational functions of degree 6/6 and 7/7.

ACCURACY:

Absolute error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	4.4e-17	6.3e-18
IEEE	0, 30	60000	4.2e-16	1.1e-16

j1

Bessel function of order one.

SYNOPSIS:

```
double x, y, j1();  
y = j1(x);
```

DESCRIPTION:

Returns a Bessel function of order one of the argument.

The domain is divided into the intervals $[0, 8]$ and $(8, \text{infinity})$. In the first interval a 24 term Chebyshev expansion is used. In the second, the asymptotic trigonometric representation is employed, using two rational functions of degree 5/5.

ACCURACY:

Absolute error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	4.0e-17	1.1e-17
IEEE	0, 30	30000	2.6e-16	1.1e-16

jn

Bessel function of integer order.

SYNOPSIS:

```
int n;  
double x, y, jn();  
y = jn(n, x);
```

DESCRIPTION:

Returns a Bessel function of order n , where n is a (possibly negative) integer.

The ratio of $j_n(x)$ to $j_0(x)$ is computed by backward recurrence. First the ratio j_n/j_{n-1} is found by a continued fraction expansion. Then the recurrence relating successive orders is applied until j_0 or j_1 is reached.

If $n = 0$ or 1 the routine for j_0 or j_1 is called directly.

ACCURACY:

Absolute error:

arithmetic	range	# trials	peak	rms
DEC	0, 30	5500	6.9e-17	9.3e-18
IEEE	0, 30	5000	4.4e-16	7.9e-17

Not suitable for large n or x . Use $jv()$ instead.

jv

Bessel function of non-integer order.

SYNOPSIS:

```
double v, x, y, jv();
y = jv(v, x);
```

DESCRIPTION:

Returns a Bessel function of order v of the argument, where v is real. Negative x is allowed if v is an integer.

Several expansions are included: the ascending power series, the Hankel expansion, and two transitional expansions for large v . If v is not too large, it is reduced by recurrence to a region of best accuracy. The transitional expansions give 12D accuracy for $v > 500$.

ACCURACY:

Results for integer v are indicated by *, where x and v both vary from -125 to +125. Otherwise, x ranges from 0 to 125, v ranges as indicated by "domain." Error criterion is absolute, except relative when $|jv()| > 1$.

arithmetic	v domain	x domain	# trials	peak	rms
IEEE	0,125	0,125	100000	4.6e-15	2.2e-16
IEEE	-125,0	0,125	40000	5.4e-11	3.7e-13
IEEE	0,500	0,500	20000	4.4e-15	4.0e-16

Integer v:

IEEE	-125,125	-125,125	50000	3.5e-15*	1.9e-16*
------	----------	----------	-------	----------	----------

y0

Bessel function of the second kind, order zero, of the argument.

SYNOPSIS:

```
double x, y, y0();  
y = y0(x);
```

DESCRIPTION:

Returns a Bessel function of the second kind, of order zero, of the argument.

The domain is divided into the intervals [0, 5] and (5, infinity). In the first interval a rational approximation R(x) is employed to compute:

$$y_0(x) = R(x) + 2 * \log(x) * j_0(x) / \text{PI}.$$

Thus a call to j0() is required.

In the second interval, the Hankel asymptotic expansion is employed with two rational functions of degree 6/6 and 7/7.

ACCURACY:

Absolute error, when $y_0(x) < 1$; else relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	9400	7.0e-17	7.9e-18
IEEE	0, 30	30000	1.3e-15	1.6e-16

y1

Bessel function of second kind of order one.

SYNOPSIS:

```
double x, y, y1();  
y = y1(x);
```

DESCRIPTION:

Returns a Bessel function of the second kind of order one of the argument.

The domain is divided into the intervals $[0, 8]$ and $(8, \text{infinity})$. In the first interval a 25 term Chebyshev expansion is used, and a call to `j1()` is required. In the second, the asymptotic trigonometric representation is employed using two rational functions of degree 5/5.

ACCURACY:

Absolute error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	8.6e-17	1.3e-17
IEEE	0, 30	30000	1.0e-15	1.3e-16

(error criterion relative when $|y1| > 1$).

yn

Bessel function of second kind of integer order.

SYNOPSIS:

```
double x, y, yn();  
int n;  
y = yn(n, x);
```

DESCRIPTION:

Returns a Bessel function of order n, where n is a (possibly negative) integer.

The function is evaluated by forward recurrence on n, starting with values computed by the routines y0() and y1().

If n = 0 or 1 the routine for y0 or y1 is called directly.

ACCURACY:

Absolute error, except relative

when $y > 1$:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	2200	2.9e-16	5.3e-17
IEEE	0, 30	30000	3.4e-15	4.3e-16

ERROR MESSAGES:

message	condition	value returned
singularity	$x = 0$	MAXNUM
overflow		MAXNUM

Spot checked against tables for x, n between 0 and 100.

yv

Bessel function of the second kind, of non-integer order.

SYNOPSIS:

```
double v, x, y, yv();  
y = yv(v, x);
```

DESCRIPTION:

Returns a Bessel function of the second kind, of order v of the argument, where v is a non-integer.

ACCURACY:

Not accurately characterized, but spot checked against tables.

i0

Modified Bessel function of order zero.

SYNOPSIS:

```
double x, y, i0();  
y = i0(x);
```

DESCRIPTION:

Returns a modified Bessel function of order zero of the argument.

The function is defined as $i_0(x) = j_0(ix)$.

The range is partitioned into the two intervals $[0,8]$ and $(8, \text{infinity})$. Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,30	6000	8.2e-17	1.9e-17
IEEE	0,30	30000	5.8e-16	1.4e-16

i0e

Modified Bessel function of order zero, exponentially scaled.

SYNOPSIS:

```
double x, y, i0e();  
y = i0e(x);
```

DESCRIPTION:

Returns exponentially scaled modified Bessel function of order zero of the argument.
The function is defined as $i0e(x) = \exp(-|x|) j0(ix)$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	0,30	30000	5.4e-16	1.2e-16

i1

Modified Bessel function of order one.

SYNOPSIS:

```
double x, y, i1();  
y = i1(x);
```

DESCRIPTION:

Returns the modified Bessel function of order one of the argument.

The function is defined as $i1(x) = -i j1(ix)$.

The range is partitioned into the two intervals $[0,8]$ and $(8, \text{infinity})$. Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	3400	1.2e-16	2.3e-17
IEEE	0, 30	30000	1.9e-15	2.1e-16

i1e

Modified Bessel function of order one, exponentially scaled.

SYNOPSIS:

```
double x, y, i1e();  
y = i1e(x);
```

DESCRIPTION:

Returns the exponentially scaled modified Bessel function of order one of the argument.
The function is defined as $i1(x) = -i \exp(-|x|) j1(ix)$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	2.0e-15	2.0e-16

iv

Modified Bessel function of noninteger order.

SYNOPSIS:

```
double v, x, y, iv();  
y = iv(v, x);
```

DESCRIPTION:

Returns the modified Bessel function of order v of the argument. If x is negative, v must be integer-valued.

The function is defined as $I_v(x) = J_v(ix)$. Here, it is computed in terms of the confluent hypergeometric function, according to the formula:

$$I_v(x) = (x/2)^{-v} e^{-x} \text{hyperg}(v+0.5, 2v+1, 2x) / \text{gamma}(v+1)$$

If v is a negative integer, then v is replaced by $-v$.

ACCURACY:

Tested at random points (v, x) , with v between 0 and 30, x between 0 and 28.

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,30	2000	3.1e-15	5.4e-16
IEEE	0,30	10000	1.7e-14	2.7e-15

Accuracy is diminished if v is near a negative integer.

k0

Modified Bessel function, third kind, order zero.

SYNOPSIS:

```
double x, y, k0();  
y = k0(x);
```

DESCRIPTION:

Returns the modified Bessel function of the third kind of order zero of the argument.

The range is partitioned into the two intervals [0,8] and (8, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

Tested at 2000 random points between 0 and 8. Peak absolute error (relative when $K0 > 1$) was 1.46e-14; rms, 4.26e-15.

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	3100	1.3e-16	2.1e-17
IEEE	0, 30	30000	1.2e-15	1.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	$x \leq 0$	MAXNUM

k0e

Modified Bessel function, third kind, order zero, exponentially scaled.

SYNOPSIS:

```
double x, y, k0e();  
y = k0e(x);
```

DESCRIPTION:

Returns the exponentially scaled, modified Bessel function of the third kind of order zero of the argument.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	1.4e-15	1.4e-16

k1

Modified Bessel function, third kind, order one.

SYNOPSIS:

```
double x, y, k1();  
y = k1(x);
```

DESCRIPTION:

Computes the modified Bessel function of the third kind, of order one of the argument.

The range is partitioned into the two intervals [0,2] and (2, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	0, 30	3300	8.9e-17	2.2e-17
IEEE	0, 30	30000	1.2e-15	1.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	$x \leq 0$	MAXNUM

k1e

Modified Bessel function, third kind, order one, exponentially scaled.

SYNOPSIS:

```
double x, y, k1e();  
y = k1e(x);
```

DESCRIPTION:

Returns the exponentially scaled, modified Bessel function of the third kind of order one of the argument:

$$k1e(x) = \exp(x) * k1(x).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	7.8e-16	1.2e-16

kn

Modified Bessel function, third kind, integer order.

SYNOPSIS:

```
double x, y, kn();  
int n;  
y = kn(n, x);
```

DESCRIPTION:

Returns the modified Bessel function of the third kind, of order n of the argument.

The range is partitioned into the two intervals [0,9.55] and (9.55, infinity). An ascending power series is used in the low range, and an asymptotic expansion in the high range.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,30	3000	1.3e-9	5.8e-11
IEEE	0,30	90000	1.8e-8	3.0e-10

Error is high only near the crossover point $x = 9.55$ between the two expansions used.

Hypergéométrie

- [hyperg](#) - hypergéométrie confluent
- [hyp2f1](#) - Fonction hypergéométrique de Gauss
- [hyp2f0](#)
- [onef2](#) - 1F2
- [threef0](#) - 3F0

hyperg

Confluent hypergeometric function.

SYNOPSIS:

```
double a, b, x, y, hyperg();
y = hyperg(a, b, x);
```

DESCRIPTION:

Computes the confluent hypergeometric function

$$F(a, b; x) = 1 + \frac{a}{b} x + \frac{a(a+1)}{b(b+1)} \frac{x^2}{2!} + \dots$$

Many higher transcendental functions are special cases of this power series.

As is evident from the formula, b must not be a negative integer or zero unless a is an integer with $0 \geq a > b$.

The routine attempts both a direct summation of the series and an asymptotic expansion. In each case error due to roundoff, cancellation and nonconvergence is estimated. The result with smaller estimated error is returned.

ACCURACY:

Tested at random points (a, b, x), all three variables ranging from 0 to 30.

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,30	2000	1.2e-15	1.3e-16

qtst1:

21800 max = 1.4200E-14 rms = 1.0841E-15 ave = -5.3640E-17

lstd:

25500 max = 1.2759e-14 rms = 3.7155e-16 ave = 1.5384e-18

IEEE	0,30	30000	1.8e-14	1.1e-15
------	------	-------	---------	---------

Larger errors can be observed when b is near a negative integer or zero. Certain combinations of arguments yield serious cancellation errors in the power series summation and also are not in the region of near convergence of the asymptotic series. An error message is printed if the self-estimated relative error is greater than 1.0e-12.

hyp2f1

Gauss hypergeometric function ${}_2F_1$.

SYNOPSIS:

```
double a, b, c, x, y, hyp2f1();
y = hyp2f1(a, b, c, x);
```

DESCRIPTION:

$$\text{hyp2f1}(a, b, c, x) = {}_2F_1(a, b; c; x)$$

$$= 1 + \sum_{k=0}^{\infty} \frac{a(a+1)\dots(a+k) b(b+1)\dots(b+k)}{c(c+1)\dots(c+k) (k+1)!} x^k$$

Cases addressed are:

Tests and escapes for negative integer a, b, or c

Linear transformation if c - a or c - b negative integer

Special case c = a or c = b

Linear transformation for x near +1

Transformation for x < -0.5

Psi function expansion if x > 0.5 and c - a - b integer Conditionally, a recurrence on c to make c-a-b > 0

|x| > 1 is rejected.

The parameters a, b, c are considered to be integer valued if they are within 1.0e-14 of the nearest integer (1.0e-13 for IEEE arithmetic).

ACCURACY:

Relative error (-1 < x < 1):

arithmetic	domain	# trials	peak	rms
IEEE	-1,7	230000	1.2e-11	5.2e-14

Several special cases also tested with a, b, c in the range -7 to 7.

ERROR MESSAGES:

A "partial loss of precision" message is printed if the internally estimated relative error exceeds 10^{-12} .

A "singularity" message is printed on overflow or in cases not addressed (such as $x < -1$).

hyp2f0

See the [hyperg](#) Help topic.

onef2

Voir la rubrique d'aide [struve](#) .

threef0

Voir la rubrique d'aide [struve](#) .

Elliptique

- [ellpe](#) - intégrale elliptique complète (E)
- [ellie](#) - intégrale elliptique incomplète (E)
- [ellpk](#) - intégrale elliptique complète (K)
- [ellik](#) - intégrale elliptique incomplète (K)
- [ellpj](#) - Fonctions elliptiques jacobines

ellpe

Complete elliptic integral of the second kind.

SYNOPSIS:

```
double m1, y, ellpe();
y = ellpe(m1);
```

DESCRIPTION:

Approximates the integral

$$E(m) = \int_0^{\pi/2} \sqrt{1 - m \sin^2 t} \, dt$$

Where $m = 1 - m1$, using the approximation:

$$P(x) - x \log x Q(x).$$

Though there are no singularities, the argument `m1` is used rather than `m`, for compatibility with `ellpk()`.

$E(1) = 1$; $E(0) = \pi/2$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	0, 1	13000	3.1e-17	9.4e-18
IEEE	0, 1	10000	2.1e-16	7.3e-17

ERROR MESSAGES:

message	condition	value returned
domain	$x < 0, x > 1$	0.0

ellie

Incomplete elliptic integral of the second kind.

SYNOPSIS:

```
double phi, m, y, ellie();
y = ellie(phi, m);
```

DESCRIPTION:

Approximates the integral:

$$E(\phi|m) = \int_0^{\phi} \sqrt{1 - m \sin^2 t} dt$$

of amplitude ϕ and modulus m , using the arithmetic - geometric mean algorithm.

ACCURACY:

Tested at random arguments with ϕ in $[-10, 10]$ and m in $[0, 1]$.

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0,2	2000	1.9e-16	3.4e-17
IEEE	-10,10	150000	3.3e-15	1.4e-16

ellpk

Complete elliptic integral of the first kind.

SYNOPSIS:

```
double m1, y, ellpk();
y = ellpk(m1);
```

DESCRIPTION:

Approximates the integral:

$$K(m) = \int_0^{\pi/2} \frac{dt}{\sqrt{1 - m \sin^2 t}}$$

where $m = 1 - m_1$, using the approximation:

$$P(x) - \log x Q(x).$$

The argument m_1 is used rather than m , so that the logarithmic singularity at $m = 1$ will be shifted to the origin; this preserves maximum accuracy.

$K(0) = \pi/2$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	0,1	16000	3.5e-17	1.1e-17
IEEE	0,1	30000	2.5e-16	6.8e-17

ERROR MESSAGES:

message	condition	value returned
---------	-----------	----------------

domain $x < 0, x > 1$ 0.0

ellik

Incomplete elliptic integral of the first kind.

SYNOPSIS:

```
double phi, m, y, ellik();
y = ellik(phi, m);
```

DESCRIPTION:

Approximates the integral:

$$F(\text{phi_}m) = \int_0^{\text{phi}} \frac{dt}{\sqrt{1 - m \sin^2 t}}$$

of amplitude phi and modulus m, using the arithmetic - geometric mean algorithm.

ACCURACY:

Tested at random points with m in [0, 1] and phi as indicated.

Relative error:

arithmetic	domain	# trials	peak	rms
------------	--------	----------	------	-----

ellpj

Jacobian Elliptic Functions.

SYNOPSIS:

```
double u, m, sn, cn, dn, phi;
int ellpj();
ellpj(u, m, _&sn, _&cn, _&dn, _&phi);
```

DESCRIPTION:

Evaluates the Jacobian elliptic functions $\text{sn}(u|m)$, $\text{cn}(u|m)$, and $\text{dn}(u|m)$ of parameter m between 0 and 1, and real argument u .

These functions are periodic, with quarter-period on the real axis equal to the complete elliptic integral $\text{ellpk}(1.0-m)$.

Relation to incomplete elliptic integral:

If $u = \text{ellik}(\text{phi}, m)$, then $\text{sn}(u|m) = \sin(\text{phi})$, and $\text{cn}(u|m) = \cos(\text{phi})$.

Phi is called the amplitude of u .

Computation is by means of the arithmetic-geometric mean algorithm, except when m is within $1e-9$ of 0 or 1.

In the latter case with m close to 1, the approximation applies only for $\text{phi} < \pi/2$.

ACCURACY:

Tested at random points with u between 0 and 10, m between 0 and 1.

Absolute error (* = relative error):

arithmetic	function	# trials	peak	rms
DEC	sn	1800	4.5e-16	8.7e-17
IEEE	phi	10000	9.2e-16*	1.4e-16*
IEEE	sn	50000	4.1e-15	4.6e-16
IEEE	cn	40000	3.6e-15	4.4e-16
IEEE	dn	100000	3.9e-15	1.7e-16

Larger errors occur for m near 1.

Peak error observed in consistency check using addition theorem for $\text{sn}(u+v)$ was $4e-16$ (absolute). Also tested by the earlier relation to the incomplete elliptic integral. Accuracy deteriorates when u is large.

Probabilité

- [bdtr](#) - Distribution binomiale
- [bdtrc](#) - Binôme complémenté
- [bdtri](#) - Binôme inverse
- [chdtr](#) - Distribution du Chi carré
- [chdtrc](#) - Chi carré complémenté
- [chdtri](#) - Chi carré inverse
- [fdtr](#) - Distribution F
- [fdtrc](#) - F complété
- [fdtri](#) - Distribution inverse de F
- [gdtr](#) - Distribution gamma
- [gdtrc](#) - gamma complété
- [nbdtr](#) - Distribution binomiale négative
- [nbdtrc](#) - Binôme négatif complémenté
- [ndtr](#) - Distribution normale
- [ndtri](#) - Distribution normale inverse
- [pdtr](#) - Distribution de Poisson
- [pdtrc](#) - Poisson complémenté
- [pdtri](#) - Distribution de Poisson inverse
- [stdtr](#) - Distribution t de Student

bdtr

Binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, bdtr();
y = bdtr(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms 0 through k of the Binomial probability density:

$$\sum_{j=0}^k \binom{n}{j} p^j (1-p)^{n-j}$$

The terms are not summed directly; instead the incomplete beta integral is employed, according to the formula:

$$y = \text{bdtr}(k, n, p) = \text{incbet}(n-k, k+1, 1-p).$$

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

	a,b	Relative error:		
		# trials	peak	rms
arithmetic domain				
For p between 0.001 and 1:				
IEEE	0,100	100000	4.3e-15	2.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	k < 0	0.0
	n < k	
	x < 0, x > 1	

bdtrc

Complemented binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, bdtrc();
y = bdtrc(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms $k+1$ through n of the Binomial probability density:

$$\sum_{j=k+1}^n \binom{n}{j} p^j (1-p)^{n-j}$$

The terms are not summed directly; instead the incomplete beta integral is employed, according to the formula:

$$y = \text{bdtrc}(k, n, p) = \text{incbet}(k+1, n-k, p).$$

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p) .

a,b		Relative error:		
arithmetic	domain	# trials	peak	rms
For p between 0.001 and 1:				
IEEE	0,100	100000	6.7e-15	8.2e-16
For p between 0 and .001:				
IEEE	0,100	100000	1.5e-13	2.7e-15

ERROR MESSAGES:

message	condition	value returned
---------	-----------	----------------

domain $x < 0, x > 1, n < k$ 0.0

bdtri

Inverse binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, bdtri();
p = bdtr(k, n, y);
```

DESCRIPTION:

Finds the event probability p such that the sum of the terms 0 through k of the Binomial probability density is equal to the given cumulative probability y .

This is accomplished using the inverse beta integral function and the relation:

$$1 - p = \text{incbi}(n-k, k+1, y).$$

ACCURACY:

Tested at random points (a,b,p).

a,b		Relative error:		
arithmetic	domain	# trials	peak	rms
For p between 0.001 and 1:				
IEEE	0,100	100000	2.3e-14	6.4e-16
IEEE	0,10000	100000	6.6e-12	1.2e-13
For p between 10^{-6} and 0.001:				
IEEE	0,100	100000	2.0e-12	1.3e-14
IEEE	0,10000	100000	1.5e-12	3.2e-14

See the [incbi](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	$k < 0, n \leq k$	0.0
	$x < 0, x > 1$	

chdtr

Chi-square distribution.

SYNOPSIS:

```
double df, x, y, chdtr();
y = chdtr(df, x);
```

DESCRIPTION:

Returns the area under the left hand tail (from 0 to x) of the Chi square probability density function, with v degrees of freedom.

$$P(x | v) = \frac{1}{2^{v/2} \Gamma(v/2)} \int_0^x t^{v/2-1} e^{-t/2} dt$$

where x is the Chi-square variable.

The incomplete gamma integral is used, according to the formula:

$$y = \text{chdtr}(v, x) = \text{igam}(v/2.0, x/2.0).$$

The arguments must both be positive.

ACCURACY:

See the [igam\(\)](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0 or v < 1	0.0

chdtrc

Complemented Chi-square distribution.

SYNOPSIS:

```
double v, x, y, chdtrc();
y = chdtrc(v, x);
```

DESCRIPTION:

Returns the area under the right hand tail (from x to infinity) of the Chi square probability density function with v degrees of freedom:

$$P(x | v) = \frac{1}{2^{v/2} \Gamma(v/2)} \int_x^{\infty} t^{v/2-1} e^{-t/2} dt$$

where x is the Chi-square variable.

The incomplete gamma integral is used, according to the formula:

$$y = \text{chdtr}(v, x) = \text{igamc}(v/2.0, x/2.0).$$

The arguments must both be positive.

ACCURACY:

See the [igamc\(\)](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0 or v < 1	0.0

chdtri

Inverse of complemented Chi-square distribution.

SYNOPSIS:

```
double df, x, y, chdtri();  
x = chdtri(df, y);
```

DESCRIPTION:

Finds the Chi-square argument x , such that the integral from x to infinity of the Chi-square density is equal to the given cumulative probability y .

This is accomplished using the inverse gamma integral function and the relation:

$$x/2 = \text{igami}(df/2, y);$$

ACCURACY:

See the [igami](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	$y < 0$ or $y > 1$	0.0
	$v < 1$	

fdtr

F distribution.

SYNOPSIS:

```
int df1, df2;
double x, y, fdtr();
y = fdtr(df1, df2, x);
```

DESCRIPTION:

Returns the area from zero to x under the F density function (also known as Snedcor's density, or the variance ratio density).

This is the density of $x = (u1/df1)/(u2/df2)$, where u1 and u2 are random variables having Chi square distributions with df1 and df2 degrees of freedom, respectively.

The incomplete beta integral is used, according to the formula

$$P(x) = \text{incbet}(df1/2, df2/2, (df1*x)/(df2 + df1*x)).$$

The arguments a and b are greater than zero, and x is nonnegative.

ACCURACY:

Tested at random points (a,b,x).

x	a,b		Relative error:		
	domain	domain	# trials	peak	rms
IEEE	0,1	0,100	100000	9.8e-15	1.7e-15
IEEE	1,5	0,100	100000	6.5e-15	3.5e-16
IEEE	0,1	1,10000	100000	2.2e-11	3.3e-12
IEEE	1,5	1,10000	100000	1.1e-11	1.7e-13

See the [incbet](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	a<0, b<0, x<0	0.0

fdtrc

Complemented F distribution.

SYNOPSIS:

```
int df1, df2;
double x, y, fdtrc();
y = fdtrc(df1, df2, x);
```

DESCRIPTION:

Returns the area from x to infinity under the F density function (also known as Snedcor's density or the variance ratio density).

$$1-P(x) = \frac{\int_x^{\infty} t^{a-1} (1-t)^{b-1} dt}{B(a,b)}$$

The incomplete beta integral is used, according to the formula

$$P(x) = \text{incbet}(df2/2, df1/2, (df2/(df2 + df1*x))).$$

ACCURACY:

Tested at random points (a,b,x) in the indicated intervals.

x	a,b		Relative error:		
	domain	domain	# trials	peak	rms
IEEE	0,1	1,100	100000	3.7e-14	5.9e-16
IEEE	1,5	1,100	100000	8.0e-15	1.6e-15
IEEE	0,1	1,10000	100000	1.8e-11	3.5e-13
IEEE	1,5	1,10000	100000	2.0e-11	3.0e-12

ERROR MESSAGES:

message	condition	value returned
domain	a<0, b<0, x<0	0.0

fdtri

Inverse of complemented F distribution.

SYNOPSIS:

```
int df1, df2;
double x, p, fdtri();
x = fdtri(df1, df2, p);
```

DESCRIPTION:

Finds the F density argument x, such that the integral from x to infinity of the F density is equal to the given probability p.

This is accomplished using the inverse beta integral function and the relations:

$$z = \text{incbi}(\text{df2}/2, \text{df1}/2, p)$$

$$x = \text{df2} (1-z) / (\text{df1} z).$$

Note: These relations hold for the inverse of the uncomplemented F distribution:

$$z = \text{incbi}(\text{df1}/2, \text{df2}/2, p)$$

$$x = \text{df2} z / (\text{df1} (1-z)).$$

ACCURACY:

Tested at random points (a,b,p).

arithmetic domain	a,b	# trials	Relative error:	
			peak	rms
For p between .001 and 1:				
IEEE	1,100	100000	8.3e-15	4.7e-16
IEEE	1,10000	100000	2.1e-11	1.4e-13
For p between 10 ⁻⁶ and 10 ⁻³ :				
IEEE	1,100	50000	1.3e-12	8.4e-15
IEEE	1,10000	50000	3.0e-12	4.8e-14

See the [fdtrc](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
---------	-----------	----------------

domain $p \leq 0$ or $p > 1$ 0.0
 $v < 1$

gdr

Gamma distribution function.

SYNOPSIS:

```
double a, b, x, y, gdr();
y = gdr(a, b, x);
```

DESCRIPTION:

Returns the integral from zero to x of the gamma probability density function:

$$y = \frac{1}{\Gamma(b)} \int_0^x t^{b-1} e^{-at} dt$$

The incomplete gamma integral is used, according to the relation:

```
y = igam(b, ax).
```

ACCURACY:

See the [igam\(\)](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0	0.0

gdtrc

Complemented gamma distribution function.

SYNOPSIS:

```
double a, b, x, y, gdtrc();
y = gdtrc(a, b, x);
```

DESCRIPTION:

Returns the integral from x to infinity of the gamma probability density function:

$$y = \int_x^{\infty} \frac{b^{-a}}{\Gamma(a)} t^{a-1} e^{-bt} dt$$

The incomplete gamma integral is used, according to the relation:

```
y = igamc(b, ax).
```

ACCURACY:

See the [igamc\(\)](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	$x < 0$	0.0

nbdtr

Negative binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, nbdtr();
y = nbdtr(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms 0 through k of the negative binomial distribution:

$$\sum_{j=0}^k \binom{n+j-1}{j} p^j (1-p)^{n-j}$$

In a sequence of Bernoulli trials, this is the probability that k or fewer failures precede the nth success.

The terms are not computed individually; instead the incomplete beta integral is employed, according to the formula:

$$y = \text{nbdtr}(k, n, p) = \text{incbet}(n, k+1, p).$$

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

a,b		Relative error:		
arithmetic domain	# trials	peak	rms	
IEEE	0,100	100000	1.7e-13	8.8e-15

See the [incbet](#) Help topic.

nbdtrc

Complemented negative binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, nbdtrc();
y = nbdtrc(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms k+1 to infinity of the negative binomial distribution:

$$\sum_{j=k+1}^{\infty} \binom{n+j-1}{j} p^j (1-p)^{n-j}$$

The terms are not computed individually; instead the incomplete beta integral is employed, according to the formula:

$$y = \text{nbdtrc}(k, n, p) = \text{incbet}(k+1, n, 1-p).$$

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

a,b		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0,100	100000	1.7e-13	8.8e-15

See the [incbet](#) Help topic.

ndtr

Normal distribution function.

SYNOPSIS:

```
double x, y, ndtr();
y = ndtr(x);
```

DESCRIPTION:

Returns the area under the Gaussian probability density function, integrated from minus infinity to x:

$$\text{ndtr}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x \exp(-t/2) dt$$

$$= (1 + \text{erf}(z)) / 2$$

$$= \text{erfc}(z) / 2$$

where $z = x/\sqrt{2}$.

Computation is via the functions erf and erfc, with care to avoid error amplification in computing $\exp(-x^2)$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	-13,0	30000	1.3e-15	2.2e-16

ERROR MESSAGES:

message	condition	value returned
underflow	$x > 37.519379347$	0.0

ndtri

Inverse of Normal distribution function.

SYNOPSIS:

```
double x, y, ndtri();
x = ndtri(y);
```

DESCRIPTION:

Returns the argument, x, for which the area under the Gaussian probability density function (integrated from minus infinity to x) is equal to y.

For small arguments $0 < y < \exp(-2)$, the program computes $z = \sqrt{-2.0 * \log(y)}$; then the approximation is $x = z - \log(z)/z - (1/z) P(1/z) / Q(1/z)$.

There are two rational functions P/Q, one for $0 < y < \exp(-32)$ and the other for y up to $\exp(-2)$.

For larger arguments, $w = y - 0.5$, and $x/\sqrt{2\pi} = w + w^{*3} R(w^{*2})/S(w^{*2})$.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	0.125, 1	5500	9.5e-17	2.1e-17
DEC	6e-39, 0.135	3500	5.7e-17	1.3e-17
IEEE	0.125, 1	20000	7.2e-16	1.3e-16
IEEE	3e-308, 0.135	50000	4.6e-16	9.8e-17

ERROR MESSAGES:

message	condition	value returned
domain	$x \leq 0$	-MAXNUM
domain	$x \geq 1$	MAXNUM

pdtr

Poisson distribution.

SYNOPSIS:

```
int k;  
double m, y, pdtr();  
y = pdtr(k, m);
```

DESCRIPTION:

Returns the sum of the first k terms of the Poisson distribution:

$$\sum_{j=0}^k \frac{m^j}{j!} e^{-m}$$

The terms are not summed directly; instead the incomplete gamma integral is employed, according to the relation:

$$y = \text{pdtr}(k, m) = \text{igamc}(k+1, m).$$

The arguments must both be positive.

ACCURACY:

See the [igamc](#) Help topic.

pdtrc

Complemented poisson distribution.

SYNOPSIS:

```
int k;  
double m, y, pdtrc();  
y = pdtrc(k, m);
```

DESCRIPTION:

Returns the sum of the terms k+1 to infinity of the Poisson distribution:

$$\inf_{j=k+1} \frac{e^{-m} m^j}{j!}$$

The terms are not summed directly; instead the incomplete gamma integral is employed, according to the formula:

$$y = \text{pdtrc}(k, m) = \text{igam}(k+1, m).$$

The arguments must both be positive.

ACCURACY:

See the [igam](#) Help topic.

pdtri

Inverse Poisson distribution.

SYNOPSIS:

```
int k;  
double m, y, pdtr();  
m = pdtri(k, y);
```

DESCRIPTION:

Finds the Poisson variable x such that the integral from 0 to x of the Poisson density is equal to the given probability y . This is accomplished using the inverse gamma integral function and the relation

$$m = \text{igami}(k+1, y).$$

ACCURACY:

See the [igami](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	$y < 0$ or $y \geq 1$	0.0
	$k < 0$	

stdtr

Student's t distribution.

SYNOPSIS:

```
double t, stdtr();
short k;
y = stdtr(k, t);
```

DESCRIPTION:

Computes the integral from minus infinity to t of the Student t distribution with integer $k > 0$ degrees of freedom:

$$\int_{-\infty}^t \frac{1}{\sqrt{k\pi} \left(1 + \frac{x^2}{k}\right)^{\frac{k+1}{2}}} dx$$

Relation to incomplete beta integral:

$$1 - \text{stdtr}(k,t) = 0.5 * \text{incbet}(k/2, 1/2, z)$$

where

$$z = k/(k + t^2).$$

For $t < -2$, this is the method of computation.

For higher t , a direct method is derived from integration by parts.

Since the function is symmetric about $t=0$, the area under the right tail of the density is found by calling the function with $-t$ instead of t .

ACCURACY:

Tested at random $1 \leq k \leq 25$. The 'domain' refers to t .

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-100,-2	50000	5.9e-15	1.4e-15
IEEE	-2,100	500000	2.7e-15	4.9e-17

Divers

- [polylog](#) - Polylogarithmes
- [spence](#) - Dilogarithme
- [zetac](#) - Fonction zêta de Riemann
- [zeta](#) - Fonction zeta à deux arguments
- [struve](#) - Fonction de Struve

polylog

Polylogarithms.

SYNOPSIS:

```
double x, y, polylog();
int n;
y = polylog(n, x);
```

The polylogarithm of order n is defined by the series:

$$Li(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^n}.$$

For $x = 1$,

$$Li(1) = \sum_{k=1}^{\infty} \frac{1}{k^n} = \text{Riemann zeta function (n)}.$$

When $n = 2$, the function is the dilogarithm, related to Spence's integral:

$$Li(x) = \int_0^x \frac{-\ln(1-t)}{t} dt = \int_x^{1-x} \frac{\ln t}{1-t} dt = \text{spence}(1-x).$$

References:

Lewin, L., *Polylogarithms and Associated Functions*,
North Holland, 1981.

Lewin, L., ed., *Structural Properties of Polylogarithms*,
American Mathematical Society, 1991.

ACCURACY:

Relative error:

arithmetic	domain	n	# trials	peak	rms
IEEE	0, 1	2	50000	6.2e-16	8.0e-17
IEEE	0, 1	3	100000	2.5e-16	6.6e-17
IEEE	0, 1	4	30000	1.7e-16	4.9e-17
IEEE	0, 1	5	30000	5.1e-16	7.8e-17

spence

Dilogarithm.

SYNOPSIS:

```
double x, y, spence();
y = spence(x);
```

DESCRIPTION:

Computes the integral:

$$\text{spence}(x) = - \int_1^x \frac{\log t}{t-1} dt$$

for $x \geq 0$. A rational approximation gives the integral in the interval (0.5, 1.5). Transformation formulas for $1/x$ and $1-x$ are employed outside the basic expansion range.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
IEEE	0,4	30000	3.9e-15	5.4e-16
DEC	0,4	3000	2.5e-16	4.5e-17

zetac

Riemann zeta function.

SYNOPSIS:

```
double x, y, zetac();
```

```
y = zetac(x);
```

DESCRIPTION:

inf.

- -x

$\zeta(x) = \sum_{k=2}^{\infty} k^{-x}$, $x > 1$,

-

k=2

Is related to the Riemann zeta function by:

$$\text{Riemann zeta}(x) = \zeta(x) + 1.$$

Extension of the function definition for $x < 1$ is implemented.

Zero is returned for $x > \log_2(\text{MAXNUM})$.

An overflow error might occur for large negative x , due to the gamma function in the reflection formula.

ACCURACY:

Tabulated values have full machine accuracy.

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	1,50	10000	9.8e-16	1.3e-16
DEC	1,50	2000	1.1e-16	1.9e-17

zeta

Riemann zeta function of two arguments.

SYNOPSIS:

```
double x, q, y, zeta();
y = zeta(x, q);
```

DESCRIPTION:

$$\text{zeta}(x, q) = \sum_{k=0}^{\infty} \frac{x^{-k}}{(k+q)}$$

where $x > 1$ and q is not a negative integer or zero.

The Euler-Maclaurin summation formula is used to obtain the expansion

$$\text{zeta}(x, q) = \sum_{k=1}^n \frac{x^{-k}}{(k+q)} - \frac{1}{2(n+q)} + \sum_{j=1}^{\infty} \frac{B_{2j}}{(2j)! (n+q)}$$

$$+ \frac{1-x}{x-1} - \frac{1}{x} + \sum_{j=1}^{\infty} \frac{B_{2j} x(x+1)\dots(x+2j)}{(2j)! (n+q)}$$

where the B_{2j} are Bernoulli numbers.

Note that $\text{zeta}(x, 1) = \text{zeta}(x) + 1$.

(see [zetac](#))

ACCURACY:

REFERENCE:

Gradshteyn, I. S., and I. M. Ryzhik, *Tables of Integrals, Series, and Products*, p. 1073; Academic Press, 1980.

struve

Struve function.

SYNOPSIS:

```
double v, x, y, struve();  
y = struve(v, x);
```

DESCRIPTION:

Computes the Struve function $H_v(x)$ of order v , argument x . Negative x is rejected unless v is an integer.

This module also contains the hypergeometric functions $1F2$ and $3F0$, and a routine for the Bessel function $Y_v(x)$ with noninteger v .

ACCURACY:

Not accurately characterized, but spot checked against tables.

Matrice

- [fftr](#) - Transformée de Fourier rapide
- [simq](#) - Equations linéaires simultanées
- [minv](#) - Inversion Matrice
- [mmmpy](#) - Multiplication Matrice
- [mvmpy](#) - Matrice multipliée par vecteur
- [mtransp](#) - Transposition Matrice
- [eigens](#) - Vecteurs propres (matrice symétrique)_

fftr

FFT of Real Valued Sequence.

SYNOPSIS:

```
double x[], sine[];
int m;
fftr(x, m, sine);
```

DESCRIPTION:

Computes the (complex valued) discrete Fourier transform of the real valued sequence x[]. The input sequence x[] contains $n = 2 * m$ samples. The program fills array sine[k] with $n/4 + 1$ values of $\sin(2 \text{ PI } k / n)$.

Data format for complex valued output is real part followed by imaginary part. The output is developed in the input array x[].

The algorithm takes advantage of the fact that the FFT of an n point real sequence can be obtained from an n/2 point complex FFT.

A radix 2 FFT algorithm is used.

Execution time on an LSI-11/23 with floating point chip is 1.0 sec for $n = 256$.

REFERENCE:

E. Oran Brigham, *The Fast Fourier Transform*; Prentice-Hall, Inc., 1974

simq

Solution of simultaneous linear equations $AX = B$ by Gaussian elimination with partial pivoting.

SYNOPSIS:

```
double A[n*n], B[n], X[n];
int n, flag;
int IPS[];
int simq();
rcode = simq(A, B, X, n, flag, IPS);
```

DESCRIPTION:

B, X, IPS are vectors of length n.

A is an $n \times n$ matrix (i.e. a vector of length $n*n$), stored row-wise; that is, $A(i,j) = A[ij]$, where $ij = i*n + j$, which is the transpose of the normal column-wise storage.

The contents of matrix A are destroyed.

Set flag=0 to solve.

Set flag=-1 to do a new back substitution for a different B vector using the same A matrix previously reduced when flag=0.

The routine returns nonzero on error; messages are printed.

ACCURACY:

Depends on the conditioning (range of eigenvalues) of matrix A.

REFERENCE:

Computer Solution of Linear Algebraic Systems

by George E. Forsythe and Cleve B. Moler; Prentice-Hall, 1967.

minv

Matrix inversion.

SYNOPSIS:

```
int n, errcod;
double A[n*n], X[n*n];
double B[n];
int IPS[n];
int minv();
```

```
errcod = minv(A, X, n, B, IPS);
```

DESCRIPTION:

Finds the inverse of the n by n matrix A . The result goes to X . B and IPS are scratch-pad arrays of length n . The contents of matrix A are destroyed.

The routine returns nonzero on error; error messages are printed by the subroutine `simq()`.

JavaScript:

```
function test_minv()
{
  /*
  * Finds the inverse of the n by n matrix A. The result goes
  * to X. B and IPS are scratch pad arrays of length n.
  * The contents of matrix A are destroyed
  */
  Session.Output("calling cephes.minv( A,X,n,B,IPS) where:");
  var n = 10; // n x n matrix A (10x10)
  var A = [
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
    [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
  ]
}
```

```
[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
```

```
];
```

```
var X = new Array(10); // output
```

```
var B = new Array(10); // scratch pad
```

```
var IPS = new Array(10); // scratch pad
```

```
Session.Output(" n = " + n);
```

```
Session.Output(" length of A is" + n*n);
```

```
Session.Output("A is matrix of " + dimensionsOfArray(A));
```

```
var ir = cephes.minv(A,X,n,B,IPS);
```

```
var s = cephes.geterrormsg();
```

```
if(s.length>0)
```

```
{
```

```
    Session.Output("error output by minv: " + s);
```

```
}
```

```
else
```

```
{
```

```
    Session.Output("minv returned " + ir);
```

```
    Session.Output("X is matrix of " + dimensionsOfArray(X));
```

```
    printMatrix("X",X,10,10);
```

```
}
```

```
}
```

mmpy

Matrix-Matrix multiply

SYNOPSIS

```
int r, c;  
double A[r*c], B[c*r], Y[r*r];  
mmpy( r, c, A, B, Y );
```

DESCRIPTION

Multiply an r (rows) by c (columns) matrix A on the left by a c (rows) by r (columns) matrix B on the right to produce an r by r matrix Y.

mvmpy

Matrix-Vector multiply

SYNOPSIS

```
int r, c;  
double A[r*c], V[c], Y[r];  
mvmpy( r, c, A, V, Y );
```

DESCRIPTION

Multiply r (rows) by c (columns) matrix A on the left by column vector V of dimension c on the right to produce a (column) vector Y output of dimension r .

mtransp

Matrix Transpose

SYNOPSIS

```
int n;  
double A[n*n], T[n*n];  
mtransp( n, A, T )
```

DESCRIPTION

Transpose the n by n square matrix A and put the result in T .
 T can occupy the same storage as A .

eigens

Eigenvalues and eigenvectors of a real symmetric matrix.

SYNOPSIS:

```
int n;  
double A[n*(n+1)/2], EV[n*n], E[n];  
void eigens(A, EV, E, n);
```

DESCRIPTION:

The algorithm is due to J. vonNeumann.

A[] is a symmetric matrix stored in lower triangular form. That is, $A[\text{row}, \text{column}] = A[(\text{row} * \text{row} + \text{row}) / 2 + \text{column}]$ or the equivalent with row and column interchanged. The indices row and column run from 0 through n-1.

EV[] is the output matrix of eigenvectors stored columnwise. That is, the elements of each eigenvector appear in sequential memory order. The jth element of the ith eigenvector is $EV[n * i + j] = EV[i][j]$.

E[] is the output matrix of eigenvalues. The ith element of E corresponds to the ith eigenvector (the ith row of EV).

On output, the matrix A will have been diagonalized and its original contents are destroyed.

ACCURACY:

The error is controlled by an internal parameter called RANGE which is set to 1e-10. After diagonalization, the off-diagonal elements of A will have been reduced by this factor.

ERROR MESSAGES:

None.

JavaScript:

```
function test_eigens()  
{  
    var A = [  
        [0.1,0.2,0.3,0.4],  
        [0.5,0.6,0.7,0.8],  
        [0.9,0.8,0.7,0.6],  
        [0.5,0.4,0.3,0.2]
```



```
];
var EV = new Array();
var E = new Array();
var N = 4;
Session.Output("calling cephes.eigens( A, EV, E, N) where:");
Session.Output(" A is NxN input matrix and N = " + N);
printMatrix("A",A,N,N);
cephes.eigens(A, EV, E, N);
Session.Output(" EV is matrix of " + dimensionsOfArray(EV));
printMatrix("Y",EV,N,N);
Session.Output(" ");
Session.Output(" E is matrix of " + dimensionsOfArray(E));
printMatrix("Y",E,N,N);
Session.Output(" ");
}
```

```
function printMatrix(name, M, rows, cols)
{
    for(var r = 0; r < rows; r++)
    {
        for(var c = 0; c < cols; c++)
        {
            Session.Output(name + "[" + r + "]" + c + "] = " + M[r][c]);
        }
    }
}
```

```
var str="";
```

```
function dimensionsOfArrayX(v)
{
    str += v.length;
    if(v.length)
    {
        var e = v[0];
        if(Array.isArray(e))
        {
            str += " x ";
            dimensionsOfArrayX(e);
        }
    }
}
```

```
}  
function dimensionsOfArray(v)  
{  
    str = "";  
    dimensionsOfArrayX(v);  
    return str;  
}
```

Intégration Numérique

- [simpsn](#) - La règle de Simpson

simpsn

Simpson Numerical Integration

SYNOPSIS

```
double simpsn( f, delta )
double f[];      /* tabulated function */
double delta;    /* spacing of arguments */
double simpsn( f, delta );
```

DESCRIPTION

Numerical integration of function tabulated at equally spaced arguments
Uses 8th order Cote integration formula.

Arithmétique Complexe

- [cadd](#) - Addition complexe
- [csub](#) - Soustraction
- [cmul](#) - Multiplication
- [cdiv](#) - Division
- [cabs](#) - valeur absolue
- [csqrt](#) - Racine carrée

cadd

Addition.

SYNOPSIS:

```
typedef struct {
    double r;   real part
    double i;   imaginary part
}cplx;
```

```
cplx *a, *b, *c;
```

```
cadd(a, b, c);   c = b + a
```

DESCRIPTION:

```
c.r = b.r + a.r
c.i = b.i + a.i
```

ACCURACY:

In DEC arithmetic, the test $(1/z) * z = 1$ had peak relative error $3.1e-17$, rms $1.2e-17$. The test $(y/z) * (z/y) = 1$ had peak relative error $8.3e-17$, rms $2.1e-17$.

Tests in the rectangle $\{-10,+10\}$:

Relative error:				
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	$1.4e-17$	$3.4e-18$
IEEE	cadd	100000	$1.1e-16$	$2.7e-17$
DEC	csub	10000	$1.4e-17$	$4.5e-18$
IEEE	csub	100000	$1.1e-16$	$3.4e-17$
DEC	cmul	3000	$2.3e-17$	$8.7e-18$
IEEE	cmul	100000	$2.1e-16$	$6.9e-17$
DEC	cdiv	18000	$4.9e-17$	$1.3e-17$
IEEE	cdiv	100000	$3.7e-16$	$1.1e-16$

JavaScript:

```
var a = {"r":0.5,"i",0.5};
```

```
var b = {"r":0.5,"i",0.5};  
var c = cephes.cadd(a,b);  
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

csub

Subtraction.

SYNOPSIS:

```
typedef struct {  
    double r;    real part  
    double i;    imaginary part  
}cmplx;
```

```
cmplx *a, *b, *c;  
csub(a, b, c);    c = b - a
```

DESCRIPTION:

```
c.r = b.r - a.r  
c.i = b.i - a.i
```

ACCURACY:

In DEC arithmetic, the test $(1/z) * z = 1$ had peak relative error $3.1e-17$, rms $1.2e-17$. The test $(y/z) * (z/y) = 1$ had peak relative error $8.3e-17$, rms $2.1e-17$.

Tests in the rectangle $\{-10,+10\}$:

Relative error:				
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	$1.4e-17$	$3.4e-18$
IEEE	cadd	100000	$1.1e-16$	$2.7e-17$
DEC	csub	10000	$1.4e-17$	$4.5e-18$
IEEE	csub	100000	$1.1e-16$	$3.4e-17$
DEC	cmul	3000	$2.3e-17$	$8.7e-18$
IEEE	cmul	100000	$2.1e-16$	$6.9e-17$
DEC	cdiv	18000	$4.9e-17$	$1.3e-17$
IEEE	cdiv	100000	$3.7e-16$	$1.1e-16$

JavaScript:

```
var a = {"r":0.5,"i",0.5};  
var b = {"r":0.5,"i",0.5};  
var c = cephes.csub(a,b);
```



```
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cmul

Multiplication.

SYNOPSIS:

```
typedef struct {
    double r;   real part
    double i;   imaginary part
} cmplx;
```

```
cmplx *a, *b, *c;
```

```
cmul(a, b, c);   c = b * a
```

DESCRIPTION:

```
c.r = b.r * a.r - b.i * a.i
c.i = b.r * a.i + b.i * a.r
```

ACCURACY:

In DEC arithmetic, the test $(1/z) * z = 1$ had peak relative error $3.1e-17$, rms $1.2e-17$. The test $(y/z) * (z/y) = 1$ had peak relative error $8.3e-17$, rms $2.1e-17$.

Tests in the rectangle $\{-10,+10\}$:

Relative error:				
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	$1.4e-17$	$3.4e-18$
IEEE	cadd	100000	$1.1e-16$	$2.7e-17$
DEC	csub	10000	$1.4e-17$	$4.5e-18$
IEEE	csub	100000	$1.1e-16$	$3.4e-17$
DEC	cmul	3000	$2.3e-17$	$8.7e-18$
IEEE	cmul	100000	$2.1e-16$	$6.9e-17$
DEC	cdiv	18000	$4.9e-17$	$1.3e-17$
IEEE	cdiv	100000	$3.7e-16$	$1.1e-16$

JavaScript:

```
var a = {"r":0.5,"i",0.5};
var b = {"r":0.5,"i",0.5};
var c = cephes.cmul(a,b);
```

```
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cdiv

Division.

SYNOPSIS:

```
typedef struct {
    double r;   real part
    double i;   imaginary part
}cplx;
```

```
cplx *a, *b, *c;
```

```
cdiv(a, b, c);   c = b / a
```

DESCRIPTION:

```
d = a.r * a.r + a.i * a.i
c.r = (b.r * a.r + b.i * a.i)/d
c.i = (b.i * a.r - b.r * a.i)/d
```

ACCURACY:

In DEC arithmetic, the test $(1/z) * z = 1$ had peak relative error $3.1e-17$, rms $1.2e-17$. The test $(y/z) * (z/y) = 1$ had peak relative error $8.3e-17$, rms $2.1e-17$.

Tests in the rectangle $\{-10,+10\}$:

Relative error:				
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	$1.4e-17$	$3.4e-18$
IEEE	cadd	100000	$1.1e-16$	$2.7e-17$
DEC	csub	10000	$1.4e-17$	$4.5e-18$
IEEE	csub	100000	$1.1e-16$	$3.4e-17$
DEC	cmul	3000	$2.3e-17$	$8.7e-18$
IEEE	cmul	100000	$2.1e-16$	$6.9e-17$
DEC	cdiv	18000	$4.9e-17$	$1.3e-17$
IEEE	cdiv	100000	$3.7e-16$	$1.1e-16$

JavaScript:

```
var a = {"r":0.5,"i",0.5};
```

```
var b = {"r":0.5,"i",0.5};  
var c = cephes.cdiv(a,b);  
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cabs

Complex absolute value.

SYNOPSIS:

```
double cabs();
cmplx z;
double a;
a = cabs(&z);
```

DESCRIPTION:

If $z = x + iy$

then

$$a = \sqrt{x^2 + y^2}.$$

Overflow and underflow are avoided by testing the magnitudes of x and y before squaring. If either is outside half of the floating point full scale range, both are rescaled.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-30,+30	30000	3.2e-17	9.2e-18
IEEE	-10,+10	100000	2.7e-16	6.9e-17

JavaScript:

```
var z = {"r":3.14,"i":3.14};
var a = cephes.cabs(z);
```

where a is an object of schema

```
{
  "r" : double,
  "i" : double
}
```

csqrt

Complex square root.

SYNOPSIS:

```
void csqrt();
cmplx z, w;

csqrt(&z, &w);
```

DESCRIPTION:

If $z = x + iy$, $r = |z|$, then

$$\operatorname{Im} w = \left[(r - x)/2 \right]^{1/2},$$

$$\operatorname{Re} w = y / 2 \operatorname{Im} w.$$

Note that $-w$ is also a square root of z . The root chosen is always in the upper half plane.

Because of the potential for a cancellation error in $r - x$, the result is sharpened by doing a Heron iteration (see [sqrt](#)) in complex arithmetic.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	25000	3.2e-17	9.6e-18
IEEE	-10,+10	100000	3.2e-16	7.7e-17

JavaScript:

```
var x = {"r":4.5,"i":3.14} ;
var a = cephes.csqrt(x);
```

returns a, complex object of schema

```
{
  "r" : double,
```

```
"i": double  
}
```


Exponentiel Complexe et Trigonométrie

- [cexp](#) - Exponentiel
- [clog](#) - Logarithme
- [ccos](#) - Cosinus
- [cacos](#) - Arc cosine
- [csin](#) - Sine
- [casin](#) - Arc sine
- [ctan](#) - Tangente
- [catan](#) - Arc tangente
- [ccot](#) - Cotangente

cexp

Complex exponential function.

SYNOPSIS:

```
void cexp();  
cmplx z, w;  
  
cexp(&z, &w);
```

DESCRIPTION:

Returns the exponential of the complex argument z into the complex result w .

If

$$z = x + iy,$$
$$r = \exp(x),$$

then

$$w = r \cos y + i r \sin y.$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10,+10	8700	3.7e-17	1.1e-17
IEEE	-10,+10	30000	3.0e-16	8.7e-17

clog

Complex natural logarithm.

SYNOPSIS:

```
void clog();  
cmplx z, w;  
  
clog(&z, &w);
```

DESCRIPTION:

Returns a complex logarithm to the base e (2.718...) of the complex argument x.

If $z = x + iy$, $r = \sqrt{x^2 + y^2}$,
then
 $w = \log(r) + i \arctan(y/x)$.

The arctangent ranges from $-\pi$ to $+\pi$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	7000	8.5e-17	1.9e-17
IEEE	-10,+10	30000	5.0e-15	1.1e-16

Larger relative errors can be observed for z near $1 + i0$. In IEEE arithmetic the peak absolute error is $5.2e-16$, rms absolute error $1.0e-16$.

CCOS

Complex circular cosine.

SYNOPSIS:

```
void ccos();  
cmplx z, w;  
  
ccos(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \cos x \cosh y - i \sin x \sinh y.$$

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	8400	4.5e-17	1.3e-17
IEEE	-10,+10	30000	3.8e-16	1.0e-16

cacos

Complex circular arc cosine.

SYNOPSIS:

```
void cacos();
```

```
cmplx z, w;
```

```
cacos(&z, &w);
```

DESCRIPTION:

$w = \arccos z = \text{PI}/2 - \arcsin z$.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5200	1.6e-15	2.8e-16
IEEE	-10,+10	30000	1.8e-14	2.2e-15

csin

Complex circular sine.

SYNOPSIS:

```
void csin();
```

```
cmplx z, w;
```

```
csin(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \sin x \cosh y + i \cos x \sinh y.$$

ACCURACY:

arithmetic	Relative error:			
	domain	# trials	peak	rms
DEC	-10,+10	8400	5.3e-17	1.3e-17
IEEE	-10,+10	30000	3.8e-16	1.0e-16

casin

Complex circular arc sine.

SYNOPSIS:

```
void casin();
```

```
cmplx z, w;
```

```
casin(&z, &w);
```

DESCRIPTION:

Inverse complex sine:

$$2$$
$$w = -i \operatorname{clog}(iz + \operatorname{csqrt}(1 - z^2)).$$

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10,+10	10100	2.1e-15	3.4e-16
IEEE	-10,+10	30000	2.2e-14	2.7e-15

Larger relative error can be observed for z near zero. Also tested by $\operatorname{csin}(\operatorname{casin}(z)) = z$.

ctan

Complex circular tangent.

SYNOPSIS:

```
void ctan();
```

```
cmplx z, w;
```

```
ctan(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \frac{\sin 2x + i \sinh 2y}{\cos 2x + \cosh 2y}.$$

On the real axis the denominator is zero at odd multiples of $\pi/2$. The denominator is evaluated by its Taylor series near these points.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5200	7.1e-17	1.6e-17
IEEE	-10,+10	30000	7.2e-16	1.2e-16

Also tested by $\text{ctan} * \text{ccot} = 1$ and $\text{catan}(\text{ctan}(z)) = z$.

catan

Complex circular arc tangent.

SYNOPSIS:

```
void catan();
```

```
cmplx z, w;
```

```
catan(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$\operatorname{Re} w = -\arctan\left(\frac{2x}{1-x^2-y^2}\right) + k\pi$$

$$\operatorname{Im} w = -\log\left(\frac{(x+iy)^2}{(x-iy)^2}\right)$$

$$\operatorname{Re} w = -\arctan\left(\frac{2x}{1-x^2-y^2}\right) + k\pi$$

$$\operatorname{Im} w = -\log\left(\frac{(x+iy)^2}{(x-iy)^2}\right)$$

$$\operatorname{Re} w = -\arctan\left(\frac{2x}{1-x^2-y^2}\right) + k\pi$$

$$\operatorname{Im} w = -\log\left(\frac{(x+iy)^2}{(x-iy)^2}\right)$$

$$\operatorname{Im} w = -\log\left(\frac{(x+iy)^2}{(x-iy)^2}\right)$$

$$\operatorname{Re} w = -\arctan\left(\frac{2x}{1-x^2-y^2}\right) + k\pi$$

$$\operatorname{Im} w = -\log\left(\frac{(x+iy)^2}{(x-iy)^2}\right)$$

Where k is an arbitrary integer.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5900	1.3e-16	7.8e-18
IEEE	-10,+10	30000	2.3e-15	8.5e-17

The check $\operatorname{catan}(\operatorname{ctan}(z)) = z$, with $|x|$ and $|y| < \pi/2$, had peak relative error 1.5e-16, rms relative error 2.9e-17. See also `clog()`.

ccot

Complex circular cotangent.

SYNOPSIS:

```
void ccot();
cmplx z, w;

ccot(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \frac{\sin 2x - i \sinh 2y}{\cosh 2y - \cos 2x}.$$

On the real axis, the denominator has zeros at even multiples of $\pi/2$. Near these points it is evaluated by a Taylor series.

ACCURACY:

Relative error:				
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	3000	6.5e-17	1.6e-17
IEEE	-10,+10	30000	9.2e-16	1.2e-16

Also tested by [ctan](#) * ccot = 1 + i0.

erreurs

Printing an error message

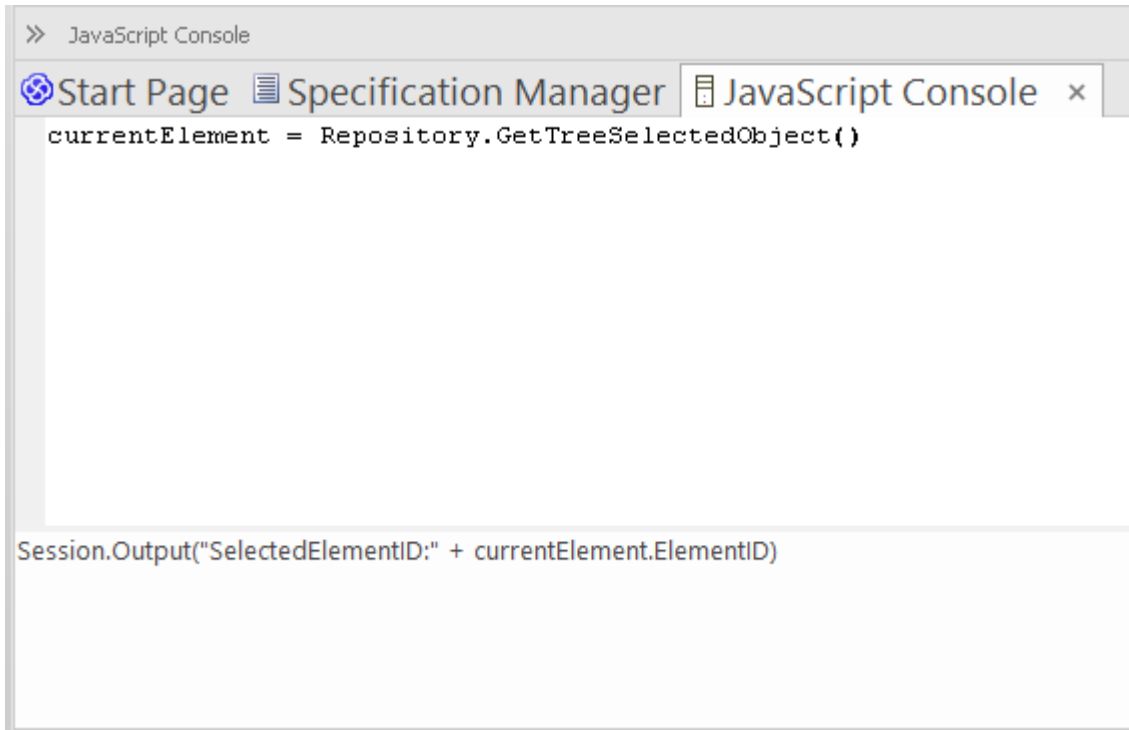
```
var cephes_errors = [  
  "unknown", /* error code 0 */  
  "domain", /* error code 1 */  
  "singularity", /* et seq. */  
  "overflow",  
  "underflow",  
  "total loss of precision",  
  "partial loss of precision" ];  
  
function printError()  
{  
  var er = cephes.geterror();  
  if(er>0)  
  {  
    Session.Output( "cephes error " + er + " " + cephes_errors[er]);  
  }  
}
```

Testing for error

```
if(cephes.inerror())  
{  
  printError();  
}
```

JavaScript Console

La console JavaScript est un interpréteur de ligne de commande qui accepte des commandes JavaScript sur une seule ligne qui seront exécutées une par une. Vous saisissez les commandes dans le panneau de saisie de texte en bas de l'écran et, lorsque vous appuyez sur la touche Entrée pour exécuter la commande, elle est ajoutée à la fenêtre de sortie supérieure avec toute sortie de la commande. Considérez cet exemple.



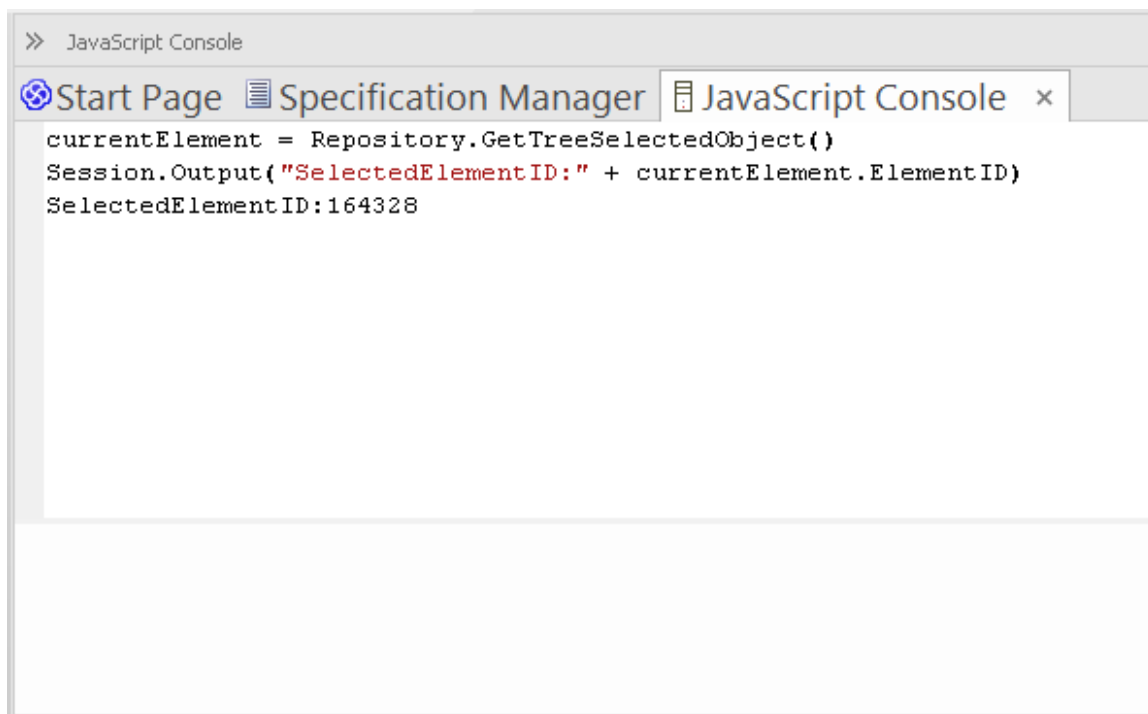
Dans le panneau inférieur, l'utilisateur a tapé :

```
currentElement = Référéntiel .GetTreeSelectedObject()
```

Ils ont appuyé sur la touche Entrée, et la commande s'est affichée dans le panneau supérieur. L'utilisateur a alors tapé, dans le panneau inférieur :

```
Session.Output("ID d'élément sélectionné :"+currentElement.ElementID)
```

Lorsqu'ils appuient sur la touche Entrée, la console affiche cette commande et la sortie de la commande.

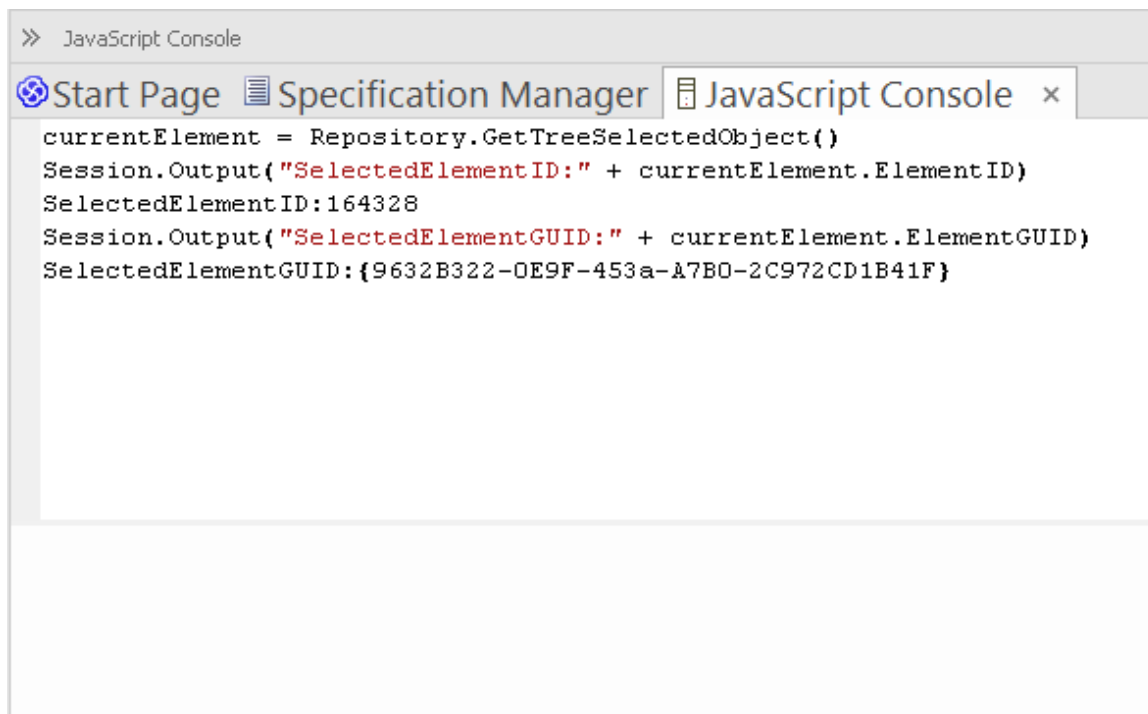


```
>> JavaScript Console
Start Page Specification Manager JavaScript Console x
currentElement = Repository.GetTreeSelectedObject()
Session.Output("SelectedElementID:" + currentElement.ElementID)
SelectedElementID:164328
```

L'utilisateur entre ensuite une troisième commande :

```
Session.Output("SelectedElementGUID: " + ElementGUID)
```

Cela donne le résultat affiché ici, dans le panneau supérieur :



```
>> JavaScript Console
Start Page Specification Manager JavaScript Console x
currentElement = Repository.GetTreeSelectedObject()
Session.Output("SelectedElementID:" + currentElement.ElementID)
SelectedElementID:164328
Session.Output("SelectedElementGUID:" + currentElement.ElementGUID)
SelectedElementGUID:{9632B322-OE9F-453a-A7B0-2C972CD1B41F}
```

Cette fonctionnalité est disponible dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect .

Accéder

Ruban	Spécialisation > Outils > JavaScript
-------	--------------------------------------

	Simuler > Console > JavaScript
--	--------------------------------

Commandes de la console

Les commandes de la console sont précédées du caractère ! et indiquent à la console d'effectuer une action.

Les commandes de console disponibles incluent :

- !clear - efface l'affichage de la console
- !save - enregistre l'affichage de la console dans un fichier
- !help - imprime une liste de commandes
- !close - ferme la console
- !include <scriptname> - exécute l'élément de script nommé ; le nom du script est au format GroupName.ScriptName (les espaces sont autorisés dans les noms)
- ? - liste les commandes (identique à !help)
- ? <nom de la variable ou de la fonction> - renvoie la valeur .

Pour lister ces commandes dans l'onglet « Console » lui-même, tapez ? dans le panneau inférieur (sans le caractère ! précédent) et appuyez sur la touche Entrée.

Si vous avez l'intention d'exécuter des scripts, vous souhaitez peut-être ouvrir également la Bibliothèque de scripts (fenêtre Scriptant), afin de pouvoir voir les scripts disponibles à l'appel. Sélectionnez l'option de ruban « Spécialisation > Outils > Bibliothèque de scripts ».

Fenêtre de la console

La fenêtre Console est un interpréteur de ligne de commande grâce auquel vous pouvez rapidement activer un moteur de script et entrer des commandes pour agir sur le script (JScript, JavaScript et VBScript).

Vous pouvez ouvrir la fenêtre Console pour JavaScript via les rubans Simuler et Spécialiser. Vous pouvez ouvrir la fenêtre Console pour VBScript et JScript via le ruban Spécialiser.

Pour les trois langages de script, vous saisissez les commandes dans le champ situé en bas de la fenêtre. Lorsque vous appuyez sur la touche Entrée, la console de script exécute les commandes et affiche immédiatement le résultat. Les commandes de la console sont décrites dans la rubrique d'aide *JavaScript Console*.

Accéder

Ruban	Spécialisation > Outils > JavaScript Spécialisation > Outils > VBScript Spécialisation > Outils > JScript Simuler > Console > JavaScript
-------	---------------------------------------------------------------------------------------------------------------------------------------------------

Notes

- Cette fonctionnalité est disponible dans les éditions Corporate, Unified et Ultimate

Interface Solveurs

L'interface Solveurs vous permet d'appeler un ensemble de commandes en JavaScript qui définissent et exécutent une classe Solveur pour effectuer des opérations mathématiques sur des données. La fonction principale de la classe Solveur est de fournir une intégration avec des outils externes tels que MATLAB et Octave pendant une simulation, et soit d'exposer les résultats dans Octave ou MATLAB, soit de les ramener dans Enterprise Architect pour y être représentés, peut-être dans un graphique dynamique. Plus généralement, l'interface Solveurs peut être utilisée dans Add-Ins basés sur des modèles et des scripts personnalisés.

Pour appeler des fonctions depuis Octave ou MATLAB, vous devez être familiarisé avec les fonctions disponibles dans la bibliothèque de produits appropriée, comme décrit dans la documentation du produit.

Solveur Constructeur

Constructeur	Description
Solver(string solverName)	Crée un nouveau Solveur connecté à une nouvelle instance de l'application d'assistance spécifiée.

Méthodes Solveur

Méthode	Description
get(string name)	Récupère une valeur nommée depuis l'environnement Solveur .
set(string name, object value)	Affecte une nouvelle valeur à une variable nommée dans l'environnement Solveur .
exec(string name, string arguments, int returnValues)	Exécute une fonction nommée. Les fonctions réelles dépendent du type de Solveur utilisé.

Éditeur de Script

En utilisant l' Éditeur de Script vous pouvez effectuer un certain nombre d'opérations sur un fichier de script ouvert, telles que :

- Enregistrer les modifications apportées au script actuel
- Enregistrer le script actuel sous un nom différent
- Exécuter le script
- Déboguer le scénario
- Arrêter l'exécution du script
- Vue la sortie du script dans l'onglet « Scripts » de la fenêtre Sortie système

L'éditeur est basé sur, et fournit les facilités de, l' Éditeur de Code commun dans le domaine de travail de l'application.

Accéder

Ruban	Spécialiser > Outils > Bibliothèque de scripts > développer le groupe de scripts et cliquez-droit sur [nom du script] > Modifier le script ou Spécialisation > Outils > Bibliothèque de scripts > développez le groupe de scripts et double-cliquez sur [nom du script]
-------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Facilités

Facilité	Détail
Objets Scriptant	Enterprise Architect ajoute aux fonctionnalités et fonctionnalités disponibles du langage de script de l'éditeur en fournissant des objets intégrés ; il s'agit soit de bibliothèques Type fournissant Intelli-sense à des fins d'édition, soit d'objets Runtime fournissant l'accès aux objets des types décrits dans les bibliothèques Type . Les objets de script Intelli-sense disponibles sont : <ul style="list-style-type: none"> • EA • MathLib • Système Les objets de script d'exécution sont : <ul style="list-style-type: none"> • Référentiel (Type : IDualRepository, une instance de EA.Repository , l'interface d'automatisation Enterprise Architect) • Maths (Type : IMath, une instance de MathLib ; cela expose les fonctions de la bibliothèque mathématique Cephés à utiliser dans les scripts) • Session (Type : ISession, une instance de System)
Édition de script Intelli-sense (syntaxe requise)	Intelli-sense est disponible non seulement dans l' Éditeur de Script , mais aussi dans la 'Console de Script' ; Intelli-sense dans sa forme la plus basique est présenté pour la fonctionnalité intégrée du moteur de script. Pour Intelli-sense sur les objets de script Enterprise Architect supplémentaires (comme répertoriés), vous devez déclarer des variables selon la syntaxe qui spécifie

	<p>un type ; il n'est pas nécessaire d'utiliser cette syntaxe pour exécuter correctement un script, elle est uniquement présente pour que l'Intelli-sense correct puisse être affiché pour un élément.</p> <p>La syntaxe peut être vue, par exemple, dans :</p> <p>Dim e comme EA.Element</p> <p>Ensuite, lorsque vous tapez, dans ce cas, e., l'éditeur affiche une liste de fonctions membres et de propriétés du type de e.</p> <p>Vous sélectionnez l'un d'entre eux pour compléter la ligne de script ; vous pouvez donc taper :</p> <p>VBTrace(e.</p> <p>Au fur et à mesure que vous tapez le point, l'éditeur présente la liste appropriée et vous pouvez double-cliquer sur, par exemple, Résumé ; celui-ci est inséré dans la ligne et vous continuez à taper ou à sélectionner le reste de l'instruction, dans ce cas en ajoutant l'espace de fin et la parenthèse :</p> <p>VBTrace(e.Abstract)</p>
Touches	<p>Dans l' Éditeur de Script ou la Console, Intelli-sense est présenté sur ces frappes de touches.</p> <ul style="list-style-type: none"> • Appuyez sur . (point) après un élément pour répertorier tous les membres du type de cet élément • Appuyez sur Ctrl+Espace sur un mot pour répertorier tous les éléments Intelli-sense dont le nom commence par la string au point où les touches ont été enfoncées • Appuyez sur Ctrl+Espace lorsque vous n'êtes pas sur un mot pour afficher tous les éléments Intelli-sense de niveau supérieur disponibles - ce sont les objets Intelli-sense déjà décrits ainsi que toutes les méthodes et propriétés intégrées du langage de script actuel
Inclure les bibliothèques de scripts	<p>Une instruction <i>Include</i> (!INC) permet à un script de référencer des constantes, des fonctions et des variables définies par un autre script accessible dans la fenêtre Scriptant . Les instructions Include sont généralement utilisées au début d'un script.</p> <p>Pour inclure une bibliothèque de scripts, utilisez cette syntaxe :</p> <p>!INC [Nom du groupe de scripts].[Nom du script]</p> <p>Par exemple:</p> <p>Scripts locaux !INC .EAConstants-VBScript</p>
Utilisation des fonctions mathématiques intégrées	<p>Diverses fonctions mathématiques sont disponibles au sein de l' Éditeur de Script , grâce à l'utilisation de l' objet Maths intégré.</p> <p>Vous pouvez accéder à l' objet Maths dans l' Éditeur de Script en tapant 'Maths' suivi d'un point. La fonctionnalité Intelli-sense affiche une liste des fonctions mathématiques disponibles fournies par la Bibliothèque Cephes Mathématiques .</p> <p>Par exemple :</p> <p>Session.Output "La racine carrée de 9 est " & Maths. sqrt (9)</p> <p>Session.Sortie "2^10 = " & Maths. pow (2,10)</p> <p>L' objet Maths est disponible dans les éditions Unified et Ultimate d' Enterprise Architect .</p>
Utilisation des objets COM / ActiveX	<p>VBScript, JScript et JavaScript peuvent tous deux créer et utiliser des objets ActiveX/COM. Cela peut vous aider à travailler avec des bibliothèques externes ou à interagir avec d'autres applications externes à Enterprise Architect . Par exemple, la classe Scripting.FileSystemObject peut être utilisée pour lire et écrire des fichiers sur la machine locale. La syntaxe de création d'un nouvel objet varie légèrement</p>

	<p>pour chaque langage, comme l'illustrent ces exemples :</p> <p>VBScript :</p> <pre>set fsObject = CreateObject(" Scripting.FileSystemObject ")</pre> <p>JScript:</p> <pre>fsObject = nouvel ActiveXObject(" Scripting.FileSystemObject ");</pre> <p>JavaScript :</p> <pre>fsObject = nouveau COMObject(" Scripting.FileSystemObject ");</pre>
Utilisation JavaScript avec des serveurs COM hors processus	<p>Les utilisateurs de JavaScript dans Enterprise Architect peuvent accéder aux serveurs COM hors processus. L'application doit être enregistrée sur la machine comme fournissant support du serveur local. La syntaxe permettant de créer ou d'obtenir une référence à un serveur hors processus est la suivante :</p> <pre>var serveur = nouveau COMObject(<i>progID</i> , true);</pre> <p>où <i>progID</i> est l' ID du programme enregistré pour le composant COM (' Excel .Application', par exemple).</p>
Bibliothèque de scripts système	<p>Lorsque Enterprise Architect est installé sur votre système, il inclut une bibliothèque de scripts par défaut qui fournit un certain nombre de fonctions de script utiles, allant des fonctions string simples aux fonctions permettant de définir votre propre importation et exportation CSV ou XML.</p> <p>Pour utiliser la bibliothèque de scripts, vous devez l'activer dans la dialogue « MDG Technologies » (option du ruban « Spécialiser > Technologies > Gérer la technologie »).</p> <p>Faites défiler la liste des technologies et sélectionnez la case à cocher « Activé » en regard de « EAScriptLib ».</p>

Notes

- L'Éditeur de Script est disponible dans les éditions Corporate , Unified et Ultimate
- Les scripts Enterprise Architect supporte la déclaration de variables correspondant aux types Enterprise Architect ; cela permet à l'éditeur de présenter Intelli-sense, mais n'est pas nécessaire pour exécuter le script

Objet de Session

L'objet d'exécution Session fournit un mécanisme d'entrée/rétroaction commun à tous les langages de script, donnant accès aux objets des types décrits dans la bibliothèque Type système. Il est disponible via la fenêtre Scriptant pour tout exécuter de script dans Enterprise Architect .

Propriétés

Propriétés	Détail
Attributes	<ul style="list-style-type: none"> • UserName - Renvoie le nom d'utilisateur Windows actuel (lecture seule) • Version - Renvoie la version de cet objet (lecture seule)
Méthodes	<ul style="list-style-type: none"> • Entrée (string Prompt) - affiche une boîte dialogue invitant l'utilisateur à saisir une valeur ; renvoie la valeur string qui a été saisie par l'utilisateur • Sortie(string Sortie) - écrit le texte à l'emplacement de sortie par défaut actuel ; pendant : <ul style="list-style-type: none"> - Exécution normale du script, la sortie est écrite dans l'onglet « Script » de la fenêtre de sortie système - Débogage de Script , la sortie est écrite dans la fenêtre Débuguer - Utilisation de la console de script, la sortie est écrite dans la console • Prompt(string Prompt, long PromptType) - affiche une dialogue modale contenant le texte prompt et les types de boutons spécifiés ; renvoie la valeur « PromptResult » correspondant au bouton sur lequel l'utilisateur a cliqué
Valeurs de PromptType	<ul style="list-style-type: none"> • promptOK = 1 • promptOUI NON = 2 • promptYESNOCANCEL = 3 • promptOKANNULER = 4
Valeurs de PromptResult	<ul style="list-style-type: none"> • résultatOK = 1 • résultatAnnuler = 2 • résultatOui = 3 • résultatNo = 4
Exemple de session.Invite	<p>(VBScript)</p> <p>Si (Session.Prompt("Continuer ?", promptYESNO) = resultYes) Alors...</p>

Flux de travail

Les flux de travail valident le travail et les actions des utilisateurs par rapport à la politique et aux procédures de votre modèle, offrant ainsi une approche robuste pour appliquer la politique de l'entreprise et renforcer les directives de développement de projet.

Les administrateurs de projet peuvent écrire des workflows pour gérer la manière dont les utilisateurs interagissent avec un modèle, comme la gestion de la sécurité, la conformité du personnel et l'accès au modèle, ainsi que la surveillance des modifications apportées par les utilisateurs. Les administrateurs peuvent également utiliser des workflows pour contrôler la capacité d'un utilisateur à modifier un élément de modèle, en tenant compte de facteurs tels que les droits d'accès, l'appartenance à un groupe et même la valeur d'une modification proposée.

Application des flux de travail

Considération	Description
Gouvernance du projet	<p>Une bonne gouvernance d'entreprise repose sur des directives de développement de projets et une politique d'entreprise bien rédigées et transparentes.</p> <p>Un projet peut être compromis si les politiques et procédures appropriées sont mal comprises et ne sont pas suivies correctement - une gouvernance efficace peut être entravée par l'erreur humaine et les coûts de récupération suite à une conformité insuffisante des développeurs.</p>
Politiques, procédures et développement	<p>La politique et les procédures de l'entreprise peuvent être intégrées au processus de développement pour gérer les flux de travail, déterminer les droits d'accès, étendre les autorisations de sécurité basées sur les rôles et répondre aux événements de changement de propriété.</p> <p>Cette approche réduit les coûts de conformité, améliore le développement collaboratif et vous donne l'assurance que les projets sont développés correctement dès la première fois.</p> <p>Les équipes de développement peuvent adhérer aux meilleures pratiques qui régissent la validation du modèle, la gestion des changements, les contrôles d'accès et les principes généraux de développement.</p>

Options de flux de travail

Il existe deux options pour utiliser les scripts Workflow :

- VBScript
- JavaScript .

La version VBScript est plus ancienne et se limite à une série de commandes. La version JavaScript est une édition plus récente et permet un accès complet à toutes les fonctions d'automatisation.

Le JavaScript est documenté sous *EA_Connect* et les rubriques d'aide *Événements du module Add-In Workflow* .

Le VBScript est documenté dans la rubrique d'aide *Fonctions de script de workflow* .

Notes

- Les workflows sont disponibles dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect

Fonctions de script de workflow

Note : à partir de la version 15.0 d' Enterprise Architect , les scripts de workflow VBScript sont disponibles mais obsolètes. Vous pouvez maintenant utiliser l'événement de modèle Add-In Enterprise Architect EA_Connect pour répondre aux événements Add-In Workflow, qui ont une gamme plus large de fonctionnalités et ne dépendent pas de Visual Basic.

Les scripts de workflow sont créés dans la fenêtre Scriptant , sous le type de groupe Workflow en tant que VBScripts. Ils sont exécutés par le moteur de workflow Enterprise Architect pour gérer les entrées utilisateur.

Vous pouvez utiliser une gamme de fonctions et de structures de données pour développer vos scripts.

Accéder

Ouvrez la fenêtre Scriptant en utilisant l'une des méthodes décrites ici, puis cliquez sur le bouton Nouveau groupe pour créer un nouveau groupe de scripts Workflow, avant de cliquer sur le bouton Nouveau script pour créer un nouveau script.

Ruban	Spécialisation > Outils > Scriptant
-------	-------------------------------------

Fonctions de workflow et structures de données

Fonction	Description
Implémentation du script	<p>Lorsqu'un modèle est lancé, le moteur de workflow est initialisé avec les appartenances actuelles des utilisateurs et des groupes ; ces informations déterminent qui peut accéder et modifier les parties d'un modèle donné.</p> <p>Lorsqu'un événement sélectionné se produit, le moteur de script est initialisé avec des valeurs incluant le nom de l'auteur et les droits d'accès, ainsi que le nom de l'élément et les détails de la version.</p> <p>Le script de workflow implémente des règles régissant la gestion des changements, le contrôle d'accès et la validation du modèle ; si un utilisateur tente d'apporter des modifications en violation de la politique de l'entreprise, le script refuse la mise à jour.</p> <p>L'utilisateur est informé de la raison de l'échec de la validation et l'activité est enregistrée.</p> <p>Ces rappels contribuent à renforcer la politique de l'entreprise, à réduire les erreurs humaines et à fournir à la direction des commentaires précieux sur le projet.</p>
Fonctions de saisie utilisateur	<p>Il s'agit de fonctions qu'Enterprise Enterprise Architect appelle pour valider et contrôler la saisie de l'utilisateur.</p> <p>Pour chacune des fonctions appelées par Enterprise Architect , un ensemble d'objets est renseigné.</p>
Fonctions pour créer une recherche	<p>Il s'agit de fonctions qu'Enterprise Enterprise Architect appelle pour créer une recherche avec des tâches utilisateur.</p>
Structures de données de flux de travail Enterprise	<p>Il s'agit d'objets de structure de données de workflow qu'Enterprise Enterprise</p>

Architect rempli	Architect rempli.
Structures de données de flux de travail que vous remplissez	Ce sont des objets de structure de données Workflow que vous pouvez remplir.
Fonctions que vous appelez	Il s'agit des fonctions qu'Enterprise Architect vous permet d'appeler.

Notes

- Si vous apportez des modifications à un script de workflow répertorié dans la fenêtre Scriptant , cliquez sur le bouton Actualiser Scripts dans la barre d'outils de la fenêtre Scriptant pour recharger le script avec les modifications
- Workflow Scriptant est disponible dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect
- Workflow Scriptant nécessite que la sécurité utilisateur soit activée pour fonctionner
- Vous avez besoin de l'autorisation « Admin Workflow » pour développer et gérer Scripts de workflow

Fonctions - Valider et contrôler la saisie de l'utilisateur

Enterprise Architect fait appel à un certain nombre de fonctions pour valider et contrôler les saisies utilisateur. Pour chaque fonction, un ensemble d'objets est renseigné.

Valider/Contrôler la saisie de l'utilisateur

Fonction	Action
AllowPhaseUpdate(AncienneValeur, NewValue)	Valider une modification apportée par un utilisateur à une phase. Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AllowStatusUpdate(AncienneValeur, NewValue)	Valider une modification apportée par un utilisateur à un statut. Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AutoriserTagUpdate(TagName, AncienneValeur, NewValue)	Valider une modification qu'un utilisateur a apportée à une Valeur Étiquetée . Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AllowVersionUpdate(AncienneValeur, NewValue)	Valider une modification apportée par un utilisateur à une version. Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
CanEditPhase()	Activer ou désactiver le contrôle d'édition d'une phase Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
CanEditStatus()	Activer ou désactiver le contrôle de modification d'un statut. Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
CanEditTag(NomDeLaBalise)	Activer ou désactiver le contrôle d'édition d'une Valeur Étiquetée . Valeur de retour : <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle

	<ul style="list-style-type: none"> • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
PeutEditVersion()	<p>Activer ou désactiver le contrôle d'édition d'une version.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
OnPreNewElement(TypeElement, Élément (Stéréotype))	<p>Autoriser ou interdire la création de l'élément spécifié.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur de créer l'élément/connecteur • Faux pour empêcher cet utilisateur de créer l'élément
OnPreNewConnector(TypeDeConnecteur, Sous-type de connecteur, Connecteur (stéréotype))	<p>Autoriser ou interdire la création du connecteur spécifié.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur de créer l'élément/connecteur • Faux pour empêcher cet utilisateur de créer l'élément
PreAllowPhaseUpdate(AncienneValeur, NewValue)	<p>Déterminez quelles informations sont nécessaires pour valider ce changement.</p> <p>Valeur de retour : Liste séparée par des points-virgules de données supplémentaires requises pour valider cette modification.</p> <p>Type de données pris en charge :</p> <ul style="list-style-type: none"> • Tests - remplissez le tableau Tests dans l' object WorkflowContext
PreAllowStatusUpdate(AncienneValeur, NewValue)	<p>Déterminez quelles informations sont nécessaires pour valider ce changement.</p> <p>Valeur de retour : Liste séparée par des points-virgules de données supplémentaires requises pour valider cette modification.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>
PreAllowTagUpdate(NomDeLaBalise, AncienneValeur, NewValue)	<p>Déterminez quelles informations sont nécessaires pour valider ce changement.</p> <p>Valeur de retour : Liste séparée par des points-virgules de données supplémentaires requises pour valider cette modification.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>
PreAllowVersionUpdate(AncienneValeur, NewValue)	<p>Déterminez quelles informations sont nécessaires pour valider ce changement.</p> <p>Valeur de retour : Liste séparée par des points-virgules de données supplémentaires requises pour valider cette modification.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>

Fonctions - Créer une recherche avec des tâches utilisateur

Il s'agit de fonctions qu'Enterprise Architect appelle pour créer une recherche avec des tâches utilisateur.

Fonctions

Fonction	Action
Obtenir des tâches de workflow	Décrivez les recherches que cet utilisateur doit exécuter . Valeur de retour : ignorée

Structures de données de flux de travail remplies

Il s'agit des structures de données de workflow (objets) qu'Enterprise Architect remplit.

Structures de données

Structure des données du flux de travail	Description
WorkflowUser	<p>Cet objet fournit des informations sur l'utilisateur actuellement connecté au modèle.</p> <p>Il est rempli par Enterprise Architect avant qu'une fonction ne soit appelée par Enterprise Architect ; il a ces propriétés :</p> <ul style="list-style-type: none"> Nom d'utilisateur - le nom d'utilisateur pour la connexion au système (si vous utilisez l'authentification Windows , cela correspond au nom d'utilisateur Windows) Prénom - tel qu'il apparaît dans la dialogue « Utilisateurs de sécurité » Nom de famille - tel qu'il apparaît dans la dialogue « Utilisateurs de sécurité » Nom complet - la combinaison <Prénom> <Nom> (le formulaire utilisé par Enterprise Architect pour les champs « Auteur » et similaires) Département - le département dans lequel travaille l'utilisateur, tel qu'il apparaît dans la dialogue « Utilisateurs de sécurité » <p>Appels : Cet objet appelle la fonction <code>IsMemberOf(GroupName)</code> .</p>
WorkflowContext	<p>Cet objet fournit des informations sur l' object actuellement dans le contexte.</p> <p>Il est rempli par Enterprise Architect avant que toute recherche, à l'exception de <code>GetWorkflowTasks</code>, ne soit exécuter ; il possède ces propriétés :</p> <ul style="list-style-type: none"> MetaType - le type de l' object actuel, soit un type de base Enterprise Architect , soit un métatype spécifié par le profil Nom - tel qu'il apparaît dans la dialogue ' Propriétés ' object Statut - tel que trouvé dans la dialogue ' Propriétés ' object Phase - telle qu'on la trouve dans la dialogue ' Propriétés ' object Version - telle que trouvée dans la dialogue ' Propriétés ' object Séréotypes - un tableau de chaînes pour les stéréotypes appliqués à cet object Étiquettes - une gamme de Valeur Étiquetés , offrant : <ul style="list-style-type: none"> Nom - le nom Valeur Étiquetée Value - la Valeur Étiquetée valeur Tests - un tableau de tests ; renseigné uniquement lors d'un appel <code>Allow*</code> après que l'appel <code>PreAllow*</code> a spécifié que des tests sont requis ; fournit ces détails, tels qu'ils se trouvent dans la fenêtre Cas de Test : <ul style="list-style-type: none"> Nom Statut Exécuté par Vérifié par Classe de test Type de test <p>Appels : Cet object appelle la fonction <code>TagValue(TagName)</code>.</p>

Fonctions

Fonction	Action
IsMemberOf(GroupName)	Vérifiez l'appartenance au groupe de l'utilisateur actuel. Valeur de retour : renvoie la valeur True si l'utilisateur actuel est membre du groupe avec le nom spécifié.
TagValue(TagName)	Obtenez la valeur d'une étiquette nommée. Valeur de retour : Renvoie la valeur de la première Valeur Étiquetée portant ce nom, ou une string vide si aucune Valeur Étiquetée portant ce nom n'existe.

Structures de données de flux de travail que vous remplissez

Ce sont les structures de données de workflow (objets) que vous pouvez remplir.

Structures de données

Structure des données du flux de travail	Description
WorkflowStatus	<p>Utilisez cette structure de données pour fournir des informations sur l'état de l'objet .</p> <ul style="list-style-type: none"> • LogEntry - défini sur True ou False pour indiquer si un élément log doit être enregistré ou non • Raison - indiquez quelle raison doit être enregistrée dans le log • Action - indique comment afficher le message log ; les valeurs valides sont : MessageBox, StatusBar et Output (par défaut)
WorkflowSearches	<p>Utilisez cette structure de données pour fournir un tableau de recherches. Utilisez Redim WorkflowSearches (x) pour spécifier le nombre de recherches fournies.</p> <p>Chaque recherche possède ces attributs :</p> <ul style="list-style-type: none"> • Nom - le nom de cette recherche • Groupe - le nom du groupe sous lequel cette recherche doit apparaître dans la zone de liste déroulante « Rechercher » • ID - le GUID pour cette recherche • Tâches - le tableau des tâches recherchées par cette recherche ; une entrée décrit comment trouver tous les objets requis pour accomplir une tâche particulière : <ul style="list-style-type: none"> - Nom - le nom de la tâche, tel qu'affiché dans la recherche Modèle vue ; les recherches de workflow sont regroupées par défaut par ce champ - Conditions - un ensemble de conditions, qui doivent toutes être remplies pour un objet à inclure dans cette tâche ; une condition est une comparaison de un seul champ à une valeur : <ul style="list-style-type: none"> - Colonne - le nom du champ - Opérateur - types d'opérateur, soit = (fournir uniquement les valeurs correspondantes) ou <> (fournir uniquement les valeurs non correspondantes) - Valeur - si elle contient une virgule, la string est traitée comme une liste de valeurs séparées par des virgules à comparer ; sinon la string est une valeur unique à comparer

Fonctions que vous appelez

indéfini

Fonctions


Fonction	Action
NewSearch(nom, groupe, guid, nombre de tâches)	indéfini
NewTask(nom, conditioncount)	indéfini
NewCondition(colonne, opérateur, valeur)	indéfini
SetLastError(message, méthode de sortie)	indéfini

Débogage de Script


Le débogage des scripts facilite le développement et la maintenance des scripts de modèles, ainsi que la surveillance de leur activité au moment de l'exécution. Lors du débogage d'un script, vous pouvez :

- Contrôler le flux d'exécution à l'aide des boutons « Débuguer », « Step Over », « Step Into », « Sortir » et « Stop Script » de la barre d'outils de l' Éditeur de Script
- Set Points d'Arrêt , Marqueurs d'Enregistrement et Point de Trace Marqueurs
- Utilisez la fenêtre Débuguer pour afficher la sortie générée par le script
- Utilisez la fenêtre Variables locales pour inspecter les valeurs des variables, y compris les objets de l'interface d'automatisation
- Utilisez la fenêtre Enregistrer et analyser pour enregistrer un diagramme Séquence de l'exécution du script

Accéder

Ruban	Spécialiser > Outils > Script Bibliothèque > cliquez-droit sur [nom du script] > Script Débogage
Autre	Barre d'outils de la fenêtre Éditeur de Script : cliquez sur l'icône de la barre d'outils 

Commencer à déboguer un script de modèle

Étape	Action
1	Ouvrez un script modèle dans l' Éditeur de Script .
2	Définissez tous Points d'Arrêt sur la ou les lignes de code appropriées.
3	Cliquez sur l'icône de la barre d'outils  (Débuguer).

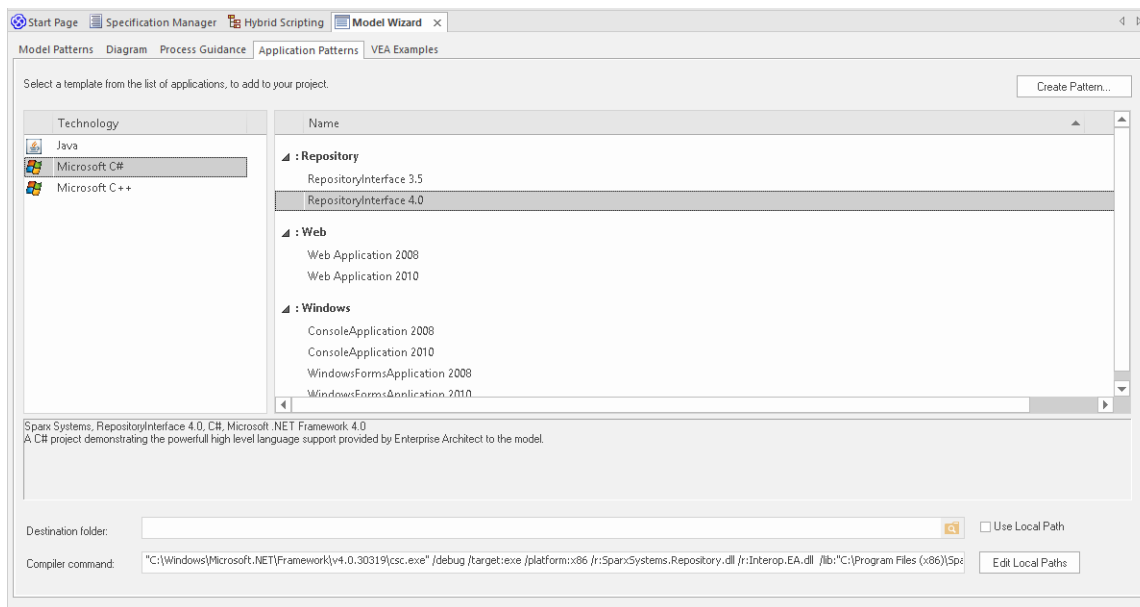
Notes

- Le débogage de script est pris en charge pour VBScript, JScript et JavaScript
- VBScript et JScript nécessitent que Microsoft Process Débuguer Manager soit installé sur la machine locale ; celui-ci est disponible via divers produits Microsoft, notamment le logiciel gratuit « Microsoft Script Débugueur »
- Points d'Arrêt ne sont pas enregistrés pour les scripts et ne persisteront pas lors de la prochaine ouverture du script
- Pendant le débogage, la sortie du script est redirigée vers la fenêtre Débuguer

Scriptant Hybride

Scriptant hybride étend les capacités de l'environnement de script standard aux langages de haut niveau tels que Java et C#. Scriptant hybride offre un avantage de vitesse par rapport aux scripts conventionnels et permet également aux auteurs de scripts de tirer parti des compétences existantes dans les langages de programmation courants.

Accéder

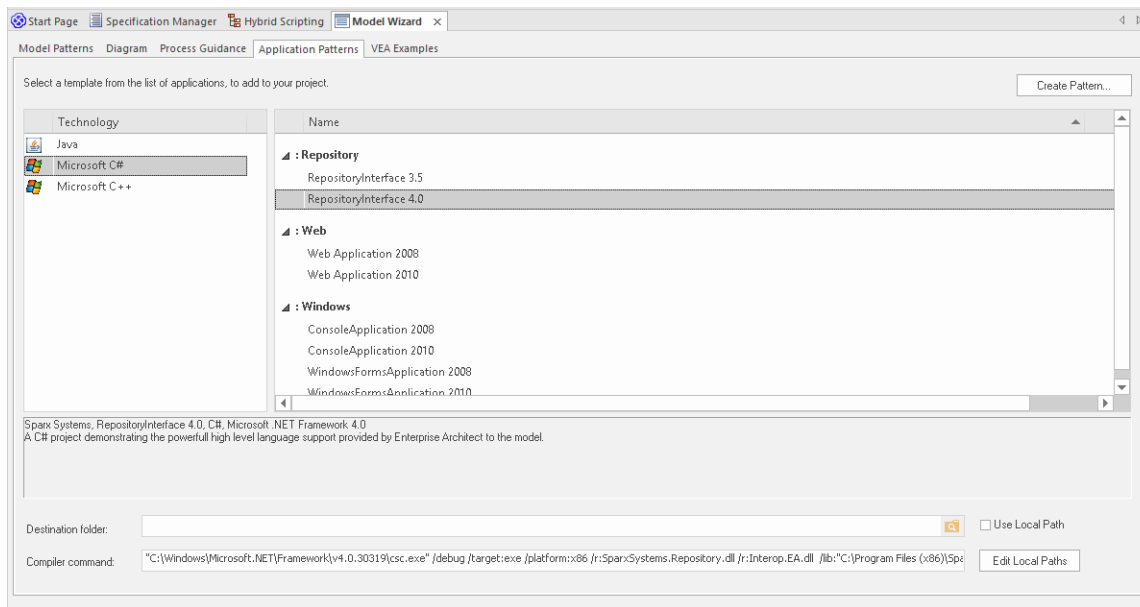


Ruban

Développer > Code source > Créer à partir de Motif > Motifs d'application

Fonctionnalités

- Vitesse d'exécution supérieure
- Interopérabilité améliorée
- support complète Analyseur d'Exécution Visuelle



Exemple C#

Cet exemple de programme montre à quel point il est facile de naviguer, d'interroger et de générer des rapports sur le modèle actuel à l'aide de n'importe quel langage Microsoft .NET . Cet exemple est écrit en C# .

Lors de l'exécution , il imprimera les noms de chaque Paquetage dans le modèle que vous utilisez actuellement.

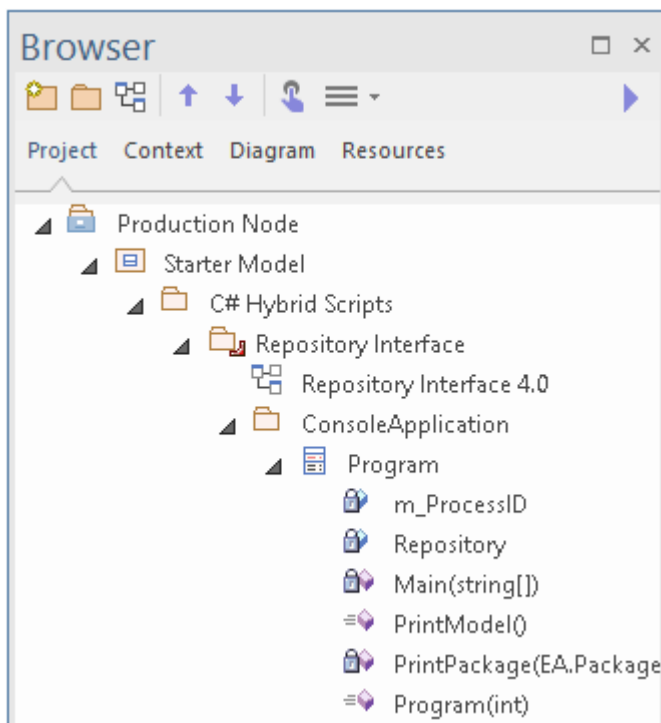
Créer le projet

Dans la fenêtre Navigateur , sélectionnez le Paquetage dans lequel créer le gabarit , puis utilisez l'option du ruban 'Développer > Code source > Créer à partir Motif ' pour afficher la fenêtre Motifs ; cliquez sur l'option 'Application Motifs '.

Depuis la page « Motifs d'application », sélectionnez le gabarit *Microsoft C# > RepositoryInterface* . (Vous pouvez choisir entre les versions 3.5 ou 4.0 du framework.) Spécifiez le dossier de destination sur le système de fichiers où le gabarit du projet sera créé, puis cliquez sur le bouton OK .

Ouvrir le projet

Une structure Paquetage similaire à celle-ci sera créée pour vous.




Développez la structure jusqu'à localiser le diagramme *Référentiel Interface nn* et ouvrez-le.

Overview:
This sample program demonstrates how easy it is to navigate, query and report on the current model using any Microsoft .NET language. This example is written in C#. When run, it will print the names of every Package in the model you are currently using.


Framework:
The build uses the C# compiler from the Microsoft .NET framework.

Version:
4.0


Note:
The links on the right operate on the active Analyzer Script. To use these links make sure you have selected the 'Repository Interface 4.0' script. You can use this Analyzer Scripts link to do this.

 Analyzer Scripts


Build the project

 Build

Run your program

 Run

Debug the program

 *DebugRun

Program

```
- m_ProcessID: int = 0
- Repository: EA.Repository = null

+ Main(string[]): void
+ PrintModel(): bool
- PrintPackage(EA.Package): void
+ Program(int)
```

Construire le script

Les commandes de ce diagramme s'appliqueront à la configuration de build active. Avant de les exécuter, double-cliquez sur le lien *Scripts d'Analyseur* et cochez la case à côté de la configuration de build ' Interface Référentiel '.

Exécuter le script

Double-cliquez sur le lien *Exécuter* pour ouvrir la console. La console s'arrêtera une fois l'exécution terminée pour que vous puissiez lire la sortie du programme ; cette sortie sera également envoyée à l'onglet « Script » de la fenêtre Sortie système. Vous pouvez modifier cela en changeant le code.

Débugger le Script

Sélectionnez la classe « Programme » dans la fenêtre Navigateur et appuyez sur Ctrl+E pour ouvrir le code source.

Placez un Point d'Arrêt dans une des fonctions puis double-cliquez sur le lien *DebugRun* . Lorsque le Point d'Arrêt est rencontré, la ligne de code sera mise en surbrillance dans l'éditeur, comme indiqué :

- ConsoleApplication
 - Program
 - Repository
 - m_ProcessID
 - Main()
 - PrintModel()
 - PrintPackage()
 - Program()
 - Trace()

```

21         Console.WriteLine(msg);
22     }
23     public Program(int pid)
24     {
25         m_ProcessID = pid;
26         Repository = SparxSystems.Services.GetRepository(m_ProcessID);
27         Trace("Running C# Console Application AppPattern .NET 3.5");
28     }
29     private void PrintPackage(EA.Package package)
30     {
31         Trace(package.Name);
32         EA.Collection packages = package.Packages;
33         for (short ip = 0; ip < packages.Count; ip++)
34         {
35             EA.Package child = (EA.Package)packages.GetAt(ip);
36             PrintPackage(child);
37         }
38     }

```

Exemple Java

Cet exemple de programme montre à quel point il est facile de naviguer, d'interroger et de générer des rapports sur le modèle actuel à l'aide d'un langage de haut niveau tel que Java.

Lorsqu'il exécute, il imprimera les noms de chaque Paquetage dans le modèle actuellement chargé.

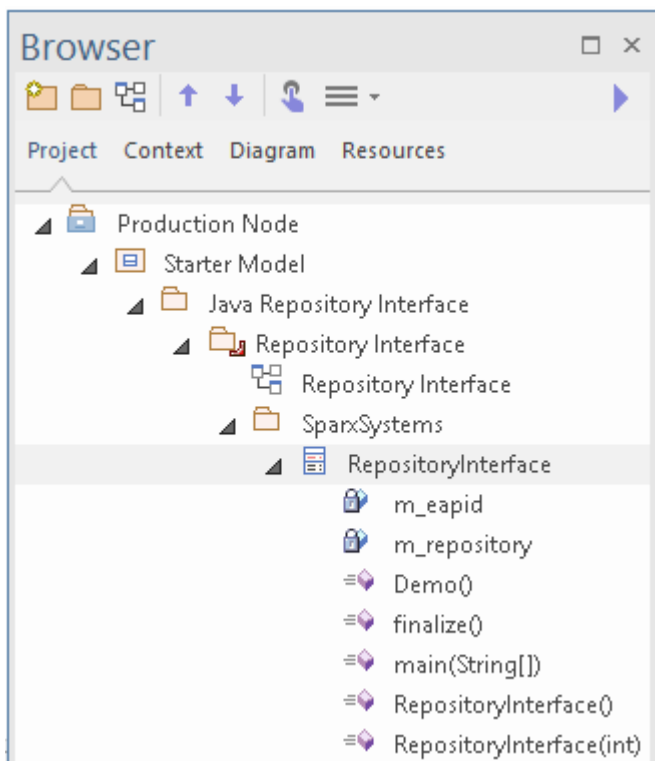
Créer le projet

Dans la fenêtre Navigateur, sélectionnez le Paquetage dans lequel créer le gabarit, puis utilisez l'option du ruban 'Développer > Code source > Créer à partir Motif' pour afficher la fenêtre Motifs; cliquez sur l'option 'Application Motifs'.

Depuis la page 'Motifs d'application', sélectionnez le gabarit *Java > RepositoryInterface*. Spécifiez le dossier de destination sur le système de fichiers dans lequel le gabarit du projet sera créé, puis cliquez sur le bouton OK.

Ouvrir le projet

Une structure Paquetage similaire à celle-ci sera créée pour vous.




Développez la structure jusqu'à localiser le diagramme 'Référentiel Interface' et ouvrez-le.


Overview:
This sample program demonstrates how easy it is to navigate, query and report on the current model using a high level language such as Java. When run, it will print the names of every Package in the currently loaded model.


Framework:
The build uses the compiler from the Java JDK 1.7 x86 framework.


Version:
1.7

Note:
In order to use the Build, Run and Debug links, you must first locate the 'Repository Interface' Analyzer Script generated by the wizard, and make it the active script for the model. You can use the 'Analyzer Scripts' link to do this.

 Analyzer Scripts

Build the project  Build

Run your program  Run

Debug the program  *DebugRun

RepositoryInterface	
-	m_eapid: int = 0
-	m_repository: org.sparx.Repository = null
+	Demo(): void
+	finalize(): void
~	main(String[]): void
+	RepositoryInterface()
+	RepositoryInterface(int)

Construire le script

Les commandes du diagramme s'appliqueront sur la configuration de build active. Avant de les exécuter, double-cliquez sur le lien *Scripts d'Analyseur* et cochez la case à côté de la configuration de build ' Interface Référentiel '.

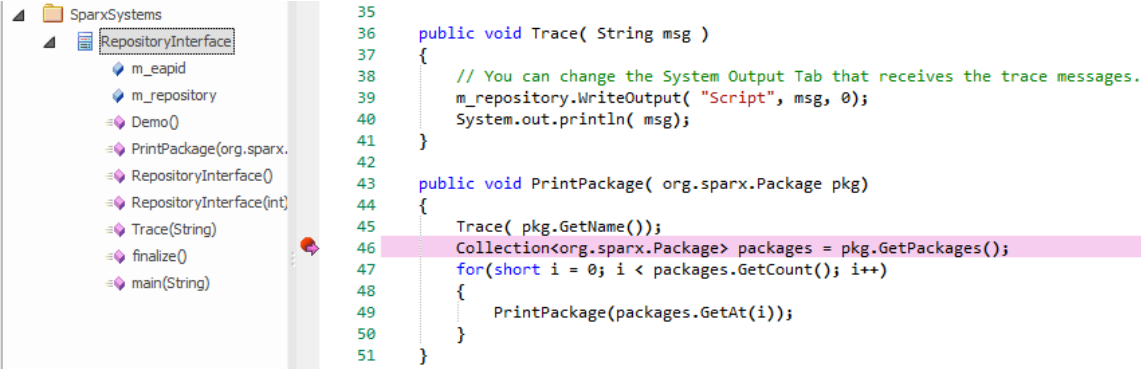
Exécuter le script

Double-cliquez sur le lien *Exécuter* ; une console s'ouvre. La console s'arrête une fois l'exécution terminée pour que vous puissiez lire le résultat. Le résultat du programme sera également affiché dans l'onglet « Script » de la fenêtre de sortie système. Vous pouvez modifier cela en modifiant le code.

Débugger le scénario

Sélectionnez la classe « Programme » dans la fenêtre Navigateur et appuyez sur Ctrl+E pour ouvrir le code source.

Placez un point d'arrêt dans l'une des fonctions, puis double-cliquez sur le lien *DebugRun* . Lorsque le point d'arrêt est rencontré, la ligne de code est mise en surbrillance dans l'éditeur, comme indiqué.



```

35
36 public void Trace( String msg )
37 {
38     // You can change the System Output Tab that receives the trace messages.
39     m_repository.WriteOutput( "Script", msg, 0);
40     System.out.println( msg);
41 }
42
43 public void PrintPackage( org.sparx.Package pkg)
44 {
45     Trace( pkg.GetName());
46     Collection<org.sparx.Package> packages = pkg.GetPackages();
47     for(short i = 0; i < packages.GetCount(); i++)
48     {
49         PrintPackage(packages.GetAt(i));
50     }
51 }
52

```

Modèle Add-Ins de modèle

Enterprise Architect offre la fonction de développer et de déployer Add-Ins entièrement au sein de votre modèle.

Quand utiliser un Add-In Modèle

Raison	Discussion
Coûts de déploiement élevés	Dans les organisations où l'installation de logiciels nouveaux ou mis à jour est coûteuse, Add-Ins de modèle peuvent offrir une solution de contournement. De nouvelles fonctionnalités peuvent être ajoutées à Enterprise Architect sans qu'il soit nécessaire d'installer de nouveaux logiciels sur les ordinateurs des utilisateurs.
Obligatoire pour tous les utilisateurs	Lorsque tous les utilisateurs d'un modèle ont besoin d'un Add-In pour utiliser le modèle comme prévu, il peut être difficile de garantir que le Add-In est installé et mis à jour sur tous les ordinateurs des utilisateurs. Les Add-Ins basés sur Modèle sont chargés automatiquement par tous les utilisateurs requis lors du chargement du modèle. Les déploiements alternatifs permettent aux utilisateurs de choisir d'utiliser un Add-In , avec un accès contrôlé par un groupe de sécurité.
Comportement spécifique Modèle	Pour les utilisateurs qui utilisent régulièrement plusieurs modèles, certaines fonctions ne sont probablement requises que dans certains modèles et pas dans d'autres. En utilisant Add-Ins basés sur des modèles, ces fonctions peuvent être ajoutées librement sans nécessiter de codage explicite basé sur le modèle.
Auto-documentation	En modélisation directement votre Add-In , la documentation le décrivant est toujours précise.

Quand ne pas utiliser un Add-In Modèle

Raison	Discussion
Interface Utilisateur complexe	L' Interface Utilisateur que les Add-Ins peuvent créer n'est pas actuellement aussi expressive que Add-Ins écrits dans un IDE traditionnel. Si vous devez montrer à vos utilisateurs des boîtes de dialogue ou des formulaires complexes, il peut être préférable d'utiliser une technologie alternative.
Utilisation sur de nombreux modèles	Les fonctionnalités Add-In requises sur plusieurs modèles peuvent ne pas être adaptées aux Add-Ins de modèle. Dans ce cas, vous devrez peut-être prendre en compte les coûts relatifs d'un Add-In traditionnel par rapport au déploiement d'un Add-In de modèle à l'aide de XMI, Paquetages contrôlés ou d'un service d'actifs réutilisables.

Notes

- Cette fonctionnalité est disponible dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect , à partir de la version 15.0


Créer un Add-In

Add-Ins basés sur Modèle sont définis dans le modèle, à l'aide de classes stéréotypées comme « JavascriptAddin ». À l'aide de ces classes stéréotypées, vous pouvez spécifier des réceptions, Méthodes et Propriétés qui définissent ensemble les comportements du Add-In et la manière dont il répond aux différents événements se produisant dans le système.

Les réceptions sont définies pour la classe, en spécifiant un signal qui sera reçu. Les réceptions vous permettent de spécifier le code JavaScript qui sera exécuté en réponse à la réception du signal correspondant. Les signaux pertinents pour votre Add-In basé sur Modèle doivent être inclus dans le modèle dans lequel vous définissez ou utilisez Add-Ins basés sur Modèle . Constructeur de Modèle propose un motif qui contient tous les signaux pertinents pour Add-Ins basés sur Modèle , offrant un moyen simple d'inclure ces signaux dans votre modèle.

Les fonctions définies comme méthodes de la classe peuvent être appelées par le code de réception, tandis que les attributs de classe peuvent être utilisés pour définir des variables globales disponibles pour le code en cours d'exécution.

Créer un Add-In JavaScript

Étape	Action
1	Cliquez sur l'icône  et sélectionnez la perspective « Gestion > Modèle Add-Ins ».
2	Créez ou ouvrez un diagramme (Classe) sur lequel travailler, puis ouvrez la page « Modèle Add-Ins » de la boîte à outils Diagramme . (Utilisez le menu Boîte à outils pour sélectionner la page « Modèle Add-ins » de la Boîte à outils.)
3	Créez un JavascriptAddin en déposant l'icône « JavascriptAddin » de la boîte à outils sur un diagramme . Le nom de votre classe JavascriptAddin sera utilisé dans le code JavaScript généré. Il doit s'agir d'un identifiant JavaScript valide.
4	Localisez la Bibliothèque de signaux. Les signaux sont utilisés pour définir les points d'entrée dans votre Add-In . Si ce n'est pas déjà fait dans votre modèle, la Bibliothèque Signal est disponible pour l'importation en tant que motif de modèle.
5	Ouvrez la liste des réceptions. Ajoutez une réception pour tout signal que vous souhaitez recevoir. Un point de départ raisonnable serait d'inclure : <ul style="list-style-type: none"> • EA_Connect • EA_GetMenuItems • EA_MenuClick
6	Ouvrez la fenêtre Comportement de votre classe ('Développer > Code source > Comportement'). Cela montre toutes les fonctionnalités comportementales disponibles auxquelles vous pouvez ajouter du code, y compris les réceptions créées précédemment. Voici quelques exemples des signaux évoqués précédemment : <p>EA_Connect</p> <pre>return "";</pre> <p>EA_GetMenuItems</p> <pre>if(MenuName == "-Exemple de module complémentaire")</pre>


	<pre>retour [" Item 1", " Item 2", "-", "À propos"]; autre retourner "-Exemple de module complémentaire"; EA_MenuClick Session.Prompt("Vous avez cliqué sur " + ItemName , 1);</pre>
7	Activez votre Add-In à l'aide de la dialogue « Gérer Add-Ins ». Si la sécurité est activée dans votre modèle, cela nécessite des droits d'administration du modèle.
8	Vous pouvez maintenant tester et développer davantage votre Add-In .

Répondre aux Événements

Pour que votre Add-In de modèle réponde aux événements, vous devez définir des réceptions sur la classe Add-In , correspondant aux signaux (ou événements) que vous souhaitez gérer. Vous pouvez ensuite définir le code du gestionnaire, à l'aide JavaScript , pour chacune des réceptions définies.

Vous pouvez également définir des fonctions supplémentaires en tant qu'opérations sur la classe, toujours à l'aide JavaScript . Ces fonctions peuvent ensuite être appelées à partir du code du gestionnaire de réception.

Définir les réceptions

Étape	Description
1	Sélectionnez un JavaScriptAddin sur un diagramme .
2	Depuis le ruban, sélectionnez l'option « Conception > Élément > Comportement ». La fenêtre de l'éditeur de code « Comportement » s'affiche.
3	Assurez-vous que l'arborescence de la structure est visible. Cliquez sur l'icône  pour basculer l'affichage de l'arborescence de structure.
4	Cliquez-droit sur la classe en haut de l'Arbre de Structure. Choisissez l'option « Ajouter une réception ». La dialogue « Sélectionner un signal » s'affiche.
5	Accédez à l'endroit où vous avez importé la Bibliothèque de référence de signal - sélectionnez le signal pour lequel vous souhaitez ajouter une réception. Cliquez sur le bouton OK .
6	Dans le panneau de droite, entrez le code JavaScript pour définir le comportement requis.
7	Répétez les étapes 4 à 6 pour tous les autres signaux que vous souhaitez gérer.

Modifier le code Add-In

La vue « Comportement » de la classe fournit une vue pratique pour éditer le code associé aux fonctionnalités comportementales de votre classe.

Accéder

Ruban	Développer > Code source > Comportement
-------	-----------------------------------------

Mise en évidence de la syntaxe

La vue « Comportement » de la classe met en évidence le code utilisant le langage attribué à la classe. Pour Add-Ins Modèle , il doit s'agir JavaScript .

Récupérer les valeurs de retour en JavaScript

Lors de la gestion d'une réception pour un événement avec des paramètres OUT/INOUT, les valeurs doivent être lues et attribuées à l'aide de l'attribut `.val` de ces paramètres.

Par exemple, pour définir la valeur du paramètre `TagValue` sur l'événement `EA_OnElementTagEdit` :

```
TagValue.val = "Hello World!"
```

Ajout d'opérations

Cliquez avec le bouton droit sur le nœud Classe en haut de l'arborescence de la structure pour ajouter une nouvelle opération.

Toutes les opérations doivent recevoir des noms valides pour les fonctions JavaScript .

Tout le code écrit sera généré dans une fonction sur un objet JavaScript . Par conséquent, pour appeler une fonction que vous avez écrite, vous devrez la préfixer avec : `this` .

Gestion des modules Add-In Modèle

Modèle Add-Ins sont gérés en fonction des groupes de sécurité auxquels ils sont affectés. Chaque Add-In est défini par défaut sur « non chargé » jusqu'à ce qu'il soit affecté à un groupe de sécurité particulier et soit activé pour tous les utilisateurs de ce groupe, soit rendu facultatif, ce qui signifie que chaque membre du groupe devra activer explicitement ce Add-In .

Accéder

Ruban	Spécialisation > Add-Ins > Gérer les compléments
-------	--------------------------------------------------

Add-Ins disponibles


La dialogue « Gérer Add-Ins » répertorie les Add-Ins actuellement activés pour votre modèle et fournit des informations sur chacun d'eux, comme expliqué dans ce tableau .

Colonne	Description
Groupes	<p>Pour les projets dans lesquels la sécurité est activée, vous pouvez sélectionner la liste des groupes de sécurité qui pourront accéder à chaque Add-In .</p> <p>Seuls les utilisateurs disposant de l'autorisation « Configurer Modèle Add-Ins du modèle » peuvent modifier cette colonne.</p>
Statut	<p>Cette colonne vous permet de sélectionner le comportement de chaque Add-In pour les utilisateurs au sein des groupes de sécurité inclus.</p> <ul style="list-style-type: none"> • Désactivé signifie que le Add-In ne peut être utilisé par aucun utilisateur • Activé signifie que le Add-In est chargé et exécuter pour tous les utilisateurs des groupes de sécurité sélectionnés • Facultatif signifie que chaque utilisateur peut choisir d'activer lui-même le Add-In ; par défaut, tous les Add-Ins seront désactivés jusqu'à ce que les utilisateurs les activent <p>Seuls les utilisateurs disposant de l'autorisation « Configurer Modèle Add-Ins du modèle » peuvent modifier cette colonne.</p>
Charge au démarrage	<p>Cette colonne permet à chaque utilisateur de spécifier qu'il souhaite utiliser les Add-Ins facultatifs disponibles pour son groupe.</p> <p>Si les utilisateurs ne font pas partie d'un groupe répertorié ou si le statut n'est pas facultatif, cela n'a aucun effet.</p>

Bibliothèque de référence des signaux

Toutes les diffusions Enterprise Architect peut envoyer à un module complémentaire sont définies dans un motif autonome qui fournit un moyen simple d'implémenter chaque signal dans vos Add-Ins de modèle.

Importer le Motif des types de diffusion

Étape	Action
1	Cliquez sur l'icône  et sélectionnez la perspective « Gestion > Modèle Add-Ins ». Cela ouvre automatiquement le Constructeur de Modèle sur la page Perspective Add-Ins Modèle .
2	Cliquez sur la cible Paquetage dans la fenêtre Navigateur .
3	Cliquez sur le motif « Types de diffusion ».
4	Cliquez sur le bouton Créer Modèle .

Exemples de Add-Ins

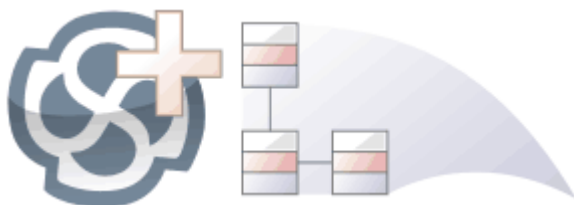
Il existe deux exemples fonctionnels de Add-Ins Modèle dans l'exemple de Modèle Enterprise Architect .

Ces exemples montrent comment :

- Ajouter du code aux réceptions
- Fonctions d'appel définies comme opérations de classe à partir du code de réception
- Utiliser Attributs de classe comme variables globales
- Créer des menus et des éléments de menu dans un module complémentaire
- Répondre à la sélection des éléments du menu du complément

Pour ouvrir le Modèle d'exemple, sélectionnez l'option du ruban ' Démarrer > Aide > Aide > Ouvrir le Modèle d'exemple '. Une fois le Modèle d'exemple chargé, recherchez ' Add-Ins basés sur Modèle '.

Enterprise Architect Modèle Add-In



Le Add-In facilite l'extension Enterprise Architect, permettant au programmeur d'améliorer l'interface utilisateur en ajoutant de nouveaux menus, sous-menus, fenêtres et autres contrôles pour exécuter diverses fonctions. Un Add-In est un objet COM ActiveX qui est informé des événements dans l'interface utilisateur, tels que les clics de souris et les sélections d'éléments, et qui a accès au contenu du référentiel via le Modèle Object. Add-Ins peuvent également être intégrés au système de gestion des licences.

Grâce à cette facilité, vous pouvez étendre Enterprise Architect pour créer de nouvelles fonctionnalités non disponibles dans le produit de base, et celles-ci peuvent être compilées et facilement distribuées à une communauté d'utilisateurs au sein d'une organisation, ou plus largement à une industrie entière. Grâce à la facilité Add-In il est même possible de créer support pour les langages et frameworks modélisation non pris en charge dans le produit de base.

Add-Ins présentent plusieurs avantages par rapport aux clients d'automatisation autonomes :

- Add-Ins peuvent (et doivent) être écrits en tant que composants en cours de traitement (DLL) ; cela permet de réduire la charge d'appel et d'améliorer l'intégration dans l'environnement Enterprise Architect
- Étant donné qu'une version actuelle d' Enterprise Architect est déjà en cours d'exécution, il n'est pas nécessaire de démarrer une deuxième copie d' Enterprise Architect via l'interface d'automatisation
- Étant donné que le Add-In reçoit des handles object associés à la copie en cours d'exécution d' Enterprise Architect, davantage d'informations sont disponibles sur l'activité de l'utilisateur actuel ; par exemple, quels objets diagramme sont sélectionnés
- Vous n'êtes pas obligé de faire autre chose que d'installer le Add-In pour le rendre utilisable ; c'est-à-dire que vous n'avez pas besoin de configurer Add-Ins à exécuter sur vos systèmes.
- Parce Enterprise Architect évolue constamment en réponse aux demandes des clients, l'interface Add-In est flexible
- L'interface Add-In n'a pas sa propre version, elle est plutôt identifiée par la version d' Enterprise Architect dans laquelle elle est apparue pour la première fois ; par exemple, la version actuelle de l'interface du Add-In Enterprise Architect est la version 2.1
- Lors de la création de votre Add-In, vous n'avez pas besoin de vous abonner à une bibliothèque de types (Add-Ins créés avant 2004 ne sont plus pris en charge - si un Add-In s'abonne à l'interface Addn_Tmpl.tlb (style 2003), il échoue au chargement ; dans ce cas, contactez le fournisseur ou l'auteur du Add-In et demandez une mise à niveau)
- Add-Ins n'ont pas besoin d'implémenter des méthodes qu'ils n'utilisent jamais
- Add-Ins prompt les utilisateurs via des menus contextuels dans l'arborescence et le diagramme
- Les états de vérification et de désactivation du menu peuvent être contrôlés par le Add-In

Add-Ins améliorent les fonctionnalités existantes d' Enterprise Architect grâce à divers mécanismes, tels que Scripts, les profils UML et l'interface d'automatisation. Une fois qu'un Add-In est enregistré, il peut être géré à l'aide du gestionnaire Add-In.

Le gestionnaire Add-In

Si vous souhaitez vérifier quels Add-Ins sont disponibles sur votre système et les activer ou les désactiver pour les utiliser, vous pouvez réviser la dialogue « Gestionnaire Add-In ». Cette dialogue répertorie les Add-Ins qui ont été enregistrés sur votre système et leur état actuel (activé ou désactivé).

Accéder

Ruban	Spécialisation > Add-Ins > Gérer les compléments
-------	--------------------------------------------------

Activer/désactiver Add-Ins

Action	Détail
Activer un Add-In	<p>Pour activer un Add-In afin qu'il soit disponible à l'utilisation, cochez la case « Charger au démarrage » correspondant au nom.</p> <p>Cliquez sur le bouton OK .</p> <ul style="list-style-type: none"> • Toutes fonctionnalités , facilités et aides spécifiques Add-In sont disponibles via l'option de menu contextuel « Spécialiser <nom du module complémentaire> » • Toutes les fenêtres Add-In définies sont renseignées avec des informations ; sélectionnez l'option de ruban « Spécialiser > Add-Ins > Windows de complément »
Désactiver un Add-In	<p>Pour désactiver un Add-In afin qu'il ne soit pas disponible à l'utilisation, décochez la case « Charger au démarrage » correspondant au nom.</p> <p>Cliquez sur le bouton OK .</p> <p>Toutes les options de menu, fonctionnalités et facilités spécifiques au Add-In sont masquées et rendues inactives.</p>

Notes

- Lorsque vous activez ou désactivez un Add-In , vous devez redémarrer Enterprise Architect pour appliquer la modification.

Créer et déployer Add-Ins

Cette rubrique vous dirige vers des informations sur la création, le test, le déploiement et la gestion Add-Ins .

Créer un Add-In

Tâche	Information
Créer le Add-In .	Quelques étapes de base sur l'utilisation d'un IDE pour créer votre module complémentaire. Consultez la rubrique <i>Créer Add-Ins</i> .
Définir Items du menu.	Quelques exemples de définition d'éléments de menu dans un module complémentaire. Voir la rubrique <i>Définir Items de menu</i> .
Répondre à Menu Événements .	Description et syntaxe d'utilisation de EA_MenuClick . Voir la rubrique <i>EA_MenuClick</i> .
Gérer Événements Add-In .	Voir la rubrique <i>Événements Add-In</i> .

Déployez votre Add-In

Considération	Information
Déployer le module complémentaire	Pour plus de détails sur l'enregistrement de la DLL du module complémentaire, consultez la rubrique <i>Déployer Add-In</i> .
Trucs et pièges	Voir la rubrique <i>Trucs et pièges</i> .

Gérer Add-Ins

Tâche	Information
Enregistrer un Add-In (développé en interne ou importé).	Les applications importées sont appelées logiciels commerciaux prêts à l'emploi (COTS). Consultez la rubrique <i>Add-In</i> .
Le gestionnaire Add-In .	Voir la rubrique <i>Gestionnaire Add-In</i> .

Créer Add-Ins

Avant de commencer, vous devez disposer d'un outil de développement d'applications capable de créer des objets COM ActiveX prenant en charge l'interface IDispatch, tels que :

- Embarcadero Delphi ou Borland Delphi
- Microsoft Visual Basic
- Microsoft Visual Studio .NET

Vous devez réfléchir à la manière de définir les éléments de menu. Pour vous aider, vous pouvez réviser quelques exemples d'interfaces d'automatisation (exemples de code utilisés pour créer Add-Ins pour Enterprise Architect) sur la page Web Sparx Systems .

Créer un Add-In Enterprise Architect

Étape	Action
1	Utilisez un outil de développement pour créer un projet DLL COM ActiveX. Les utilisateurs de Visual Basic, par exemple, choisissent Fichier-Créer un nouveau projet-DLL ActiveX.
2	Connectez-vous à l'interface en utilisant la syntaxe appropriée à la langue.
3	Créez une classe COM et implémentez chacun des Événements généraux Add-In applicables à votre Add-In . Il vous suffit de définir des méthodes pour que les événements répondent.
4	Ajoutez une clé de registre qui identifie votre Add-In à Enterprise Architect , comme décrit dans la rubrique Déployer Add-Ins .

Définir Items du menu

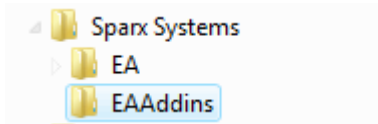
Tâches

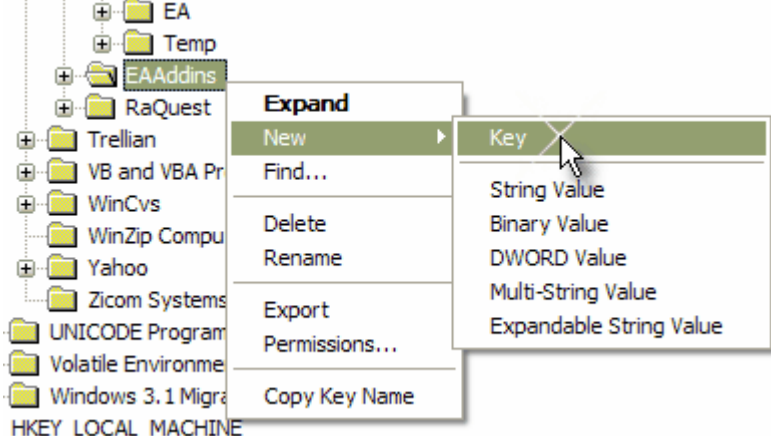
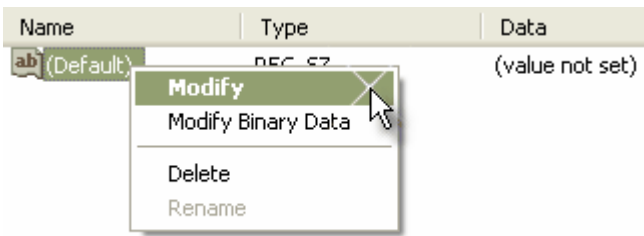
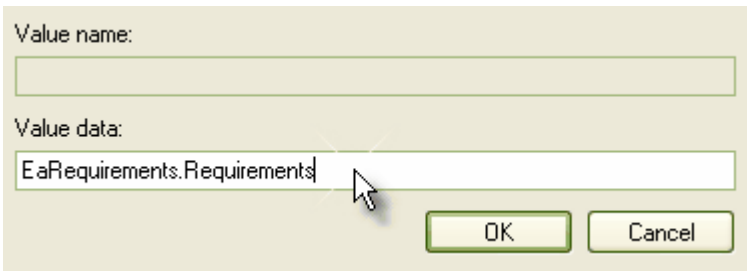
Tâche	Détail
Définir les Items de menu	<p>Les éléments de menu sont définis en répondant à l'événement GetMenuItems.</p> <p>La première fois que cet événement est appelé, MenuName est une string vide représentant le menu de niveau supérieur. Pour un Add-In simple avec une seule option de menu, vous pouvez renvoyer une string .</p> <pre>Function EA_GetMenuItems (Référentiel as EA.Repository , MenuLocation As String , MenuName As String) As Variant EA_GetMenuItems = " Add-In de &Joe" End Function</pre>
Définir les sous-menus	<p>Pour définir des sous-menus, préfixez un menu parent avec un tiret. Le parent et les sous-éléments sont définis de la manière suivante :</p> <pre>Function EA_GetMenuItems (Référentiel as EA.Repository , MenuLocation As String , MenuName As String) As Variant Sélectionnez Case MenuName Cas " " ' Item du menu parent EA_GetMenuItems = "-&Joe's Add-In " Cas "-&Joe's Add-In " 'Définir les Items de sous-menu en utilisant la notation Array. 'Dans cet exemple, " Diagramme " et " Treeview " composent le sous-menu " Joe's Add-In ". EA_GetMenuItems = Array("& Diagramme ", "&Treeview") Case Else MsgBox "Menu invalide", vbCritical End Select End Function</pre>
Définir d'autres sous-menus	<p>De même, vous pouvez définir d'autres sous-éléments :</p> <pre>Function EA_GetMenuItems (Référentiel as EA.Repository , MenuLocation As String , MenuName As String) As Variant Sélectionnez Case MenuName Cas " " EA_GetMenuItems = "-Complément Add-In Joe" Case "-Joe's Add-In" EA_GetMenuItems = Array("& Diagramme ", "&TreeView") Cas "-& Diagramme " EA_GetMenuItems = "& Propriétés " Case Else MsgBox "Menu invalide", vbCritical End Select</pre>

	End Function
Activer/désactiver les options de menu	<p>Pour activer ou désactiver les options de menu par défaut, vous pouvez utiliser cette méthode pour montrer des éléments particuliers à l'utilisateur :</p> <pre>Sub EA_GetMenuState (Référentiel As EA.Repository , Location As String , MenuName As String , ItemName As String , IsEnabled As Boolean, IsChecked As Boolean) Sélectionnez l'emplacement du cas Cas "TreeView" "Toujours activer Mallette " Diagramme " "Toujours activer Cas "MainMenu" Sélectionner le nom de l' ItemName cas Cas "&Traduire", "Enregistrer &Projet" Si GetIsProjectSelected() Alors IsEnabled = Faux Fin si End Select End Select IsChecked = GetIsCurrentSelection() Sous-titre de fin</pre>

Déployer Add-Ins

Déployer Add-Ins sur les sites des utilisateurs

Étape	Action
1	<p>Ajoutez le fichier DLL Add-In à un répertoire approprié sur l'ordinateur de l'utilisateur, c'est-à-dire :</p> <p>C:\Program Files\(\nouveau répertoire)</p>
2	<p>Enregistrez la DLL en fonction de votre plateforme :</p> <ul style="list-style-type: none"> • S'il est compilé en tant que DDL Win32 natif, tel que VB ou C++, enregistrez le DDL à l'aide de la commande regsvr32 à partir de l' prompt de commande regsvr32 "C:\Program Files\MyCompany\EAAddin\EAAddin.dll" • Si compilé en tant que DLL .NET , telle que C# ou VB.NET, enregistrez la DLL à l'aide de la commande RegAsm à partir de l' prompt de commande C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe "C:\Program Files\MonEntreprise\EAAddin\EAAddin.dll" /codebase
3	<p>Placez une nouvelle entrée dans le registre à l'aide de l'éditeur de registre (exécuter regedit) afin qu'Enterprise Architect reconnaisse la présence de votre Add-In .</p>
4	<p>Ajoutez une nouvelle clé « EAAddIns » sous l'un de ces emplacements :</p> <ul style="list-style-type: none"> • Pour l'utilisateur actuel uniquement <ul style="list-style-type: none"> - Sur Enterprise Architect 32 bits [HKEY_CURRENT_USER\Software\ Sparx Systems \EAAddins] - Sur Enterprise Architect 64 bits [HKEY_CURRENT_USER\Software\ Sparx Systems \EAAddins64] • Pour plusieurs utilisateurs sur une machine <ul style="list-style-type: none"> - Sur Enterprise Architect 32 bits [HKEY_LOCAL_MACHINE\Software\ Sparx Systems \EAAddins] - Sur Enterprise Architect 64 bits [HKEY_LOCAL_MACHINE\Software\ Sparx Systems \EAAddins64]  <p>Note : les éditions Enterprise Architect 32 et 64 bits tenteront uniquement de charger Add-Ins sous la clé correspondante : EAAddIns ou EAAddIns64, respectivement.</p>
5	<p>Ajoutez une nouvelle clé sous cette clé avec le nom du projet.</p>

	 <p>(ProjectName) n'est pas nécessairement le nom de votre DLL, mais le nom du projet ; dans Visual Basic, c'est la valeur de la propriété Name correspondant au fichier projet.</p>
6	<p>Spécifiez la valeur par défaut en modifiant la valeur par défaut de la clé.</p> 
7	<p>Entrez la valeur de la clé en tapant (nom du projet).(nom de la classe), par exemple :</p> <p>EaRequirements.Requirements</p> <p>où <i>EaRequirements</i> est le nom du projet, comme indiqué dans cet exemple :</p> 

Conseils et astuces

Considérations

Item	Détail
GUID	Le GUID d'enregistrement et le nom utilisés pour enregistrer un Add-In doivent être identiques pour les versions 32 bits et 64 bits Enterprise Architect . Seuls le nom et/ou l'emplacement de la DLL doivent être différents.
Enterprise Architect 64 bits et Add-Ins C++	Outre l'utilisation de la clé d'enregistrement correcte (voir l'étape 4 de la rubrique d'aide <i>Déployer des modules complémentaires</i>), il n'existe aucune configuration spéciale pour exécuter un objet COM 64 bits sous Enterprise Architect 64 bits.
Add-Ins .NET	<p>Lors de la génération d'un assemblage .NET , vous devez définir explicitement la « Plateforme cible » sur x86/x64. Le fait de la laisser sur « N'importe quel processeur » peut entraîner des problèmes lors de exécuter Enterprise Architect 32 bits sur une version 64 bits de Windows.</p> <p><i>Ajoutez une cible x64 à votre projet et reconstruisez le projet.</i></p> <p>Visual Studio rencontre un problème lorsqu'il tente d'enregistrer un assembly .NET lorsque Interop.EA est inclus dans un projet. Il tente d'utiliser regasm pour l'architecture pour laquelle Interop.EA a été créé. Nous avons constaté que l'ajout au script de post-construction aide, si vous décochez l'option « Enregistrer pour COM interop » dans les paramètres du projet (en supposant que vous ayez l'autorisation d'écrire dans le registre).</p> <pre>si \$(PlatformName) == x64 ("%Windir%\Microsoft.NET\Framework64\v4.0.30319\regasm" "\$(TargetPath) ") si \$(PlatformName) == x86 ("%Windir%\Microsoft.NET\Framework64\v4.0.30319\regasm" "\$(TargetPath) ")</pre> <p>Note : au moment de la rédaction de cet article, Add-ins .NET ne fonctionneront pas sous Wine et avec Wine-Mono.</p>
API Java	L'API Java charge le dernier Enterprise Architect installé et n'est pas affectée lors de l'utilisation de la version 32 ou 64 de la DLL, à condition que la DLL SSJavaCOM puisse être trouvée par l'environnement d'exécution Java.
Note pour les utilisateurs de Visual Basic 5/6	<p>Les utilisateurs de Visual Basic doivent note que le numéro de version de l'interface Enterprise Architect est stocké dans le fichier de projet VBP sous une forme similaire à celle-ci :</p> <pre>Référence=*G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#.\.\.\.\Program Files\ Sparx Systems \EA\EA.TLB#Enterprise Architect Object Modèle 2.02</pre> <p>Si vous rencontrez des problèmes lors du passage d'une version d' Enterprise Architect à une autre, ouvrez le fichier VBP dans un éditeur de texte et supprimez cette ligne. Ouvrez ensuite le projet dans Visual Basic et utilisez Project-References pour créer une nouvelle référence au modèle Object Enterprise Architect .</p>
Informations State de détection	<p>Il est possible qu'un Add-In contienne des informations d'état, ce qui signifie que les données peuvent être stockées dans des variables membres en réponse à un événement et récupérées lors d'un autre. Cette pratique comporte certains dangers :</p> <ul style="list-style-type: none"> Les objets d'automatisation Enterprise Architect ne se mettent pas à jour en réponse à l'activité de l'utilisateur, à l'activité sur d'autres postes de travail ou

	<p>même aux actions d'autres objets dans le même client d'automatisation ; la conservation des handles de ces objets entre les appels peut entraîner l'interrogation par le deuxième événement d'objets qui n'ont aucune relation avec l'état actuel d' Enterprise Architect</p> <ul style="list-style-type: none"> • Lorsque vous fermez Enterprise Architect , tous Add-Ins sont invités à s'arrêter ; s'il existe des clients d'automatisation externes, Enterprise Architect doit rester actif, auquel cas tous les Add-Ins sont rechargés, perdant toutes les données • Enterprise Architect agissant en tant que client d'automatisation ne se ferme pas si un Add-In contient toujours une référence à celui-ci (la libération de toutes les références dans l'événement Disconnect() évite ce problème) <p>Il est recommandé, sauf raison spécifique, que le Add-In utilise le paramètre de référentiel et sa méthode et ses propriétés pour fournir les données nécessaires.</p>
Enterprise Architect ne ferme pas	<p>Problèmes spécifiques .NET</p> <p>L'automatisation vérifie l'utilisation des objets et ne permet pas qu'aucun d'entre eux soit détruit jusqu'à ce qu'ils ne soient plus utilisés.</p> <p>Comme indiqué dans la rubrique <i>Interface d'automatisation</i> , si votre contrôleur d'automatisation a été écrit à l'aide du framework .NET , Enterprise Architect ne se ferme pas même après avoir libéré toutes vos références à celui-ci. Pour forcer la libération des pointeurs COM, appelez les fonctions de gestion de la mémoire comme indiqué :</p> <p>GC.Collect(); GC.WaitForPendingFinalizers();</p> <p>De plus, étant donné que les clients d'automatisation se connectent à Enterprise Architect , qui crée Add-Ins qui se connectent à leur tour à Enterprise Architect , il est possible de se retrouver dans une situation de blocage dans laquelle Enterprise Architect et les Add-Ins ne se lâcheront pas et ne resteront pas actifs. Un Add-In peut conserver des connexions à Enterprise Architect pour les raisons suivantes :</p> <ul style="list-style-type: none"> • Il conserve une référence privée à un objet Enterprise Architect (voir la section précédente <i>Informations sur l' State de conservation</i>), ou • Il a été créé par .NET et le mécanisme GC ne l'a pas encore publié <p>Deux actions sont nécessaires pour éviter les situations de blocage :</p> <ul style="list-style-type: none"> • Les contrôleurs d'automatisation doivent appeler Référentiel .CloseAddins() à un moment donné (peut-être à la fin du traitement) • Add-Ins doivent libérer toutes les références à Enterprise Architect dans l'événement Disconnect() ; consultez la rubrique <i>Événements Add-In</i> pour plus de détails <p>Il est possible que votre client Automation contrôle une instance en cours d'exécution d' Enterprise Architect où les Add-Ins ne sont pas conformes aux règles. Dans ce cas, vous pouvez appeler Référentiel .Exit() pour mettre fin à Enterprise Architect .</p> <p>Divers</p> <p>Lors du développement Add-Ins à l'aide du framework .NET vous devez sélectionner l'interopérabilité COM dans les propriétés du projet pour qu'il soit reconnu comme un Add-In .</p> <p>Certains environnements de développement n'enregistrent pas automatiquement les DLL COM lors de leur création. Vous devrez peut-être effectuer cette opération manuellement avant Enterprise Architect ne reconnaisse le Add-In .</p> <p>Vous pouvez utiliser votre clé de Add-In privée (comme requis pour le déploiement Add-In) pour stocker les informations de configuration pertinentes pour votre Add-In .</p>

Recherche Add-In

Enterprise Architect permet aux extensions de s'intégrer à la recherche Modèle . Des recherches peuvent être définies pour exécuter une méthode dans votre Add-In et afficher vos résultats de manière intégrée.

Détails

Item
La méthode qui exécute la recherche doit être structurée de cette manière.
Définit la structure XML attendue par Enterprise Architect pour spécifier les résultats de la recherche.
<p>En plus des résultats affichés, deux champs cachés supplémentaires peuvent être transmis au XML et offrent des fonctionnalités spéciales.</p> <ul style="list-style-type: none">• CLASSTYPE - Le renvoi d'un champ de CLASSTYPE, contenant la valeur Object_Type du tableau t_object, affiche l'icône appropriée dans la colonne dans laquelle vous placez le champ• CLASSGUID - Le retour d'un champ de CLASSGUID, contenant une valeur ea_guid, permet à la recherche Modèle de suivre l' object dans la fenêtre Navigateur et d'ouvrir la fenêtre Propriétés de l'élément en double-cliquant dans la recherche Modèle

EA_SampleSearch

Cela définit la signature requise pour la fonction appelée Enterprise Architect lors de l'exécution d'une recherche Add-In . Le nom peut être remplacé par n'importe quel nom de fonction valide dans votre langage de programmation cible.

Syntaxe

Paramètre	Type
Référentiel	EA.Repository Direction: DANS Description : un objet EA.Repository représentant le modèle Enterprise Architect sur le point d'être fermé. Interrogez ses membres pour récupérer les données du modèle et les informations sur l'état de l'interface utilisateur.
Texte de recherche	String Direction: DANS Description : Fournit la valeur (le cas échéant) saisie par l'utilisateur dans le champ du terme de recherche dans la fenêtre de recherche de modèle.
Résultats XML	String Direction : OUT Description : Fournit la valeur (le cas échéant) saisie par l'utilisateur dans le champ du terme de recherche dans la fenêtre de recherche de modèle.

Valeur de retour

La méthode doit renvoyer toute valeur non vide pour que les résultats soient affichés.

Format XML (recherche de données)

Cet exemple XML fournit le format du paramètre sSearchData de la méthode RunModelSearch.

```
<UID ReportViewData=\"MonIDRecherche\">
```

```
<!--
```

```
//L'attribut UID permet aux recherches de type XML de conserver les informations de colonne. Autrement dit, si vous exécuter la recherche, groupez par colonne ou ajustez
```

```
//largeurs des colonnes, puis fermez la fenêtre et exécutez à nouveau la recherche, les modifications de format/organisation sont conservées. Pour éviter la persistance des colonnes
```

```
//arrangements, laissez l'attribut valeur vide ou supprimez-le complètement. Utilisez cette section pour déclarer tous les champs possibles - colonnes qui apparaissent
```

```
//dans la fenêtre de recherche d' Enterprise Architect - qui sont utilisés ci-dessous dans <Rows/>. L'ordre des colonnes d'informations à ajouter ici doit
```

```
//correspond à l'ordre dans lequel l' exécuter de recherche dans Enterprise Architect s'affiche normalement. De plus, si vous ajoutez des résultats à un SQL personnalisé
```

```
//Recherchez, puis l'ordre utilisé dans votre SQL personnalisé doit correspondre à l'ordre utilisé ici.
```

```
-->
```

```
<Champs>
```

```
<Nom du champ=" " />
```

```
<Nom du champ=" " />
```

```
<Nom du champ=" " />
```

```
<Nom du champ=" " />
```

```
</Champs>
```

```
<Lignes>
```

```
<Ligne>
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
</Row>
```

```
<Ligne>
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
</Row>
```

```
<Ligne>
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
<Nom du champ=" " valeur=" " />
```

```
</Row>
```

```
</Rows>
```

</ReportViewData>

Événements Add-In

All Enterprise Architect Add-Ins can choose to respond to general Add-In events.

Events

Event
<i>EA_Connect</i> - Add-Ins can use this to identify their type and to respond to Enterprise Architect start up.
<i>EA_Disconnect</i> - Add-Ins can use this to respond to user requests to disconnect the model branch from an external project.
<i>EA_GetMenuItems</i> - Add-Ins can use this to provide the Enterprise Architect user interface with additional Add-In menu options in various context menus.
<i>EA_GetMenuState</i> - Add-Ins can use this to set a particular menu option to either enabled or disabled.
<i>EA_GetRibbonCategory</i> - Add-Ins can use this to identify the Ribbon panel in which to house their calling icon.
<i>EA_MenuClick</i> - received by an Add-In in response to user selection of a menu option.
<i>EA_OnOutputItemClicked</i> - informs Add-Ins that the user has clicked on a list entry in the system tab or one of the user defined output tabs.
<i>EA_OnOutputItemDoubleClicked</i> - informs Add-Ins that the user has used the mouse to double-click on a list entry in one of the user-defined output tabs.
<i>EA_ShowHelp</i> - Add-Ins can use this to show a Help topic for a particular menu option.

EA_OnAddinPropertiesTabChanging

Indicates that a value in a properties list added via Repository.AddPropertiesTab has been changed by the user.

Syntax

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the field changed: <ul style="list-style-type: none">• TabName: The name of the Add-Ins window tab changing• PropID: Unique ID assign to Property item within the xml definition.• ChangeValue: The value the Property is changing to.• OriginalValue: The original value assigned to the Property

Return Value

- Return False to indicate that this change was rejected
- Return True to indicate that the change is accepted

EA_Connect

Add-Ins can use EA_Connect events to identify their type and to respond to Enterprise Architect start up.

This event occurs when Enterprise Architect first loads your Add-In. Enterprise Architect itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

There are two key uses for EA_Connect:

- Initializing global Add-In data, along with identifying the Add-In as an MDG Add-In
- Initializing a Workflow script.

Syntax

Function EA_Connect (Repository As EA.Repository) As String

The EA_Connect function syntax has this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A string identifying a specialized type of Add-In:

Type	Details
"MDG"	MDG Add-Ins receive MDG Events and extra menu options.
"Workflow"	Workflow add-ins receive additional events to control user ability to change specific fields.
""	A non-specialized Add-In.

EA_Disconnect

Add-Ins can use the EA_Disconnect event to respond to user requests to disconnect the model branch from an external project.

This function is called when Enterprise Architect closes. If you have stored references to Enterprise Architect objects (not recommended anyway), you must release them here.

In addition, .NET users must call memory management functions as shown:

```
GC.Collect();  
GC.WaitForPendingFinalizers();
```

Syntax

```
Sub EA_Disconnect()
```

Return Value

None.

EA_GetMenuItems

The EA_GetMenuItems event enables the Add-In to provide the Enterprise Architect user interface with additional Add-In menu options in various context menus. When a user selects an Add-In menu option, an event is raised and passed back to the Add-In that originally defined that menu option.

This event is raised just before Enterprise Architect has to show particular menu options to the user, and its use is described in the *Define Menu Items* topic.

Syntax

Function EA_GetMenuItems (Repository As EA.Repository, MenuLocation As String, MenuName As String) As Variant

The EA_GetMenuItems function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MenuLocation	String Direction: IN Description: A string representing the part of the user interface that brought up the context menu. This can be TreeView, MainMenu, Diagram or Other. You can add further values for MenuLocation at any time. A MenuLocation of 'TreeView' would indicate that the menu was displayed in the Browser window; 'MainMenu' would indicate that the menu was displayed from a ribbon option, and 'Diagram' that the menu was displayed within a diagram. 'Other' would indicate an unspecified location, which might be one of these: <ul style="list-style-type: none"> • Calendar • Dialog • Element List • Gantt • Model View • Project View • Relationship Matrix • Reviews • Search • Specification Manager
MenuName	String Direction: IN Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.

Return Value

One of these types:

- A string indicating the label for a single menu option
- An array of strings indicating multiple menu options
- Empty (Visual Basic/VB.NET) or null (C#) to indicate that no menu should be displayed

In the case of the top-level menu it should be a single string or an array containing only one item, or empty/null.

EA_GetMenuState

Add-Ins can use the EA_GetMenuState event to set a particular menu option to either enabled or disabled. This is useful when dealing with locked Packages and other situations where it is convenient to show a menu option, but not enable it for use.

This event is raised just before Enterprise Architect has to show particular menu options to the user. Its use is further described in the *Define Menu Items* topic.

Syntax

Sub EA_GetMenuState (Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String, IsEnabled as Boolean, IsChecked as Boolean)

The EA_GetMenuState function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MenuLocation	String Direction: IN Description: A string representing the part of the user interface that brought up the menu. This can be TreeView, MainMenu or Diagram.
MenuName	String Direction: IN Description: The name of the parent menu for which sub-items must be defined. In the case of the top-level menu it is an empty string.
ItemName	String Direction: IN Description: The name of the option actually clicked; for example, 'Create a New Invoice'.
IsEnabled	Boolean Direction: OUT Description: Set to False to disable this particular menu option.
IsChecked	Boolean Direction: OUT Description: Set to True to check this particular menu option.

Return Value

None.

EA_GetRibbonCategory

Add-Ins can use EA_GetRibbonCategory events to identify the Ribbon in which the Add-In should place its menu icon.

This event occurs when Enterprise Architect first loads your Add-In. Enterprise Architect itself is loading at this time so that while a Repository object is supplied, there is limited information that you can extract from it.

The chief use for EA_GetRibbonCategory is in initializing the Add-In access point.

Syntax

Function EA_GetRibbonCategory (Repository As EA.Repository) As String

The EA_GetRibbonCategory function syntax has this parameter:

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A string matching the name of the selected ribbon (in English if you are using a translated version). The possible names are:

- Start
- Design
- Layout
- Publish
- Specialize
- Construct
- Code
- Simulate
- Execute
- Manage

It is not possible to include Add-Ins in the 'Specification - Specify' ribbon or 'Documentation - Edit' ribbon.

If the function isn't implemented (or if an invalid name is returned) the 'Add-In' menu will be available from the 'Specialize' ribbon, 'Add-Ins' panel.

EA_MenuClick

EA_MenuClick events are received by an Add-In in response to user selection of a menu option.

The event is raised when the user clicks on a particular menu option. When a user clicks on one of your non-parent menu options, your Add-In receives a MenuClick event, defined as:

```
Sub EA_MenuClick(Repository As EA.Repository, ByVal MenuLocation As String, ByVal MenuName As String,
ByVal ItemName As String)
```

This code is an example of use:

```
If MenuName = "-&Diagram" And ItemName = "&Properties" then
    MsgBox Repository.GetCurrentDiagram.Name, vbInformation
Else
    MsgBox "Not Implemented", vbCritical
End If
```

Notice that your code can directly access Enterprise Architect data and UI elements using Repository methods.

Syntax

```
Sub EA_MenuClick (Repository As EA.Repository, MenuLocation As String, MenuName As String, ItemName As String)
```

The EA_GetMenuClick function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MenuLocation	String Direction: IN Description: A string representing the part of the user interface that brought up the menu. This can be TreeView, MainMenu or Diagram.
MenuName	String Direction: IN Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.
ItemName	String Direction: IN Description: The name of the option actually clicked; for example, 'Create a New Invoice'.

Return Value

None.

EA_OnOutputItemClicked

EA_OnOutputItemClicked events inform Add-Ins that the user has clicked on a list entry in the system tab or one of the user defined output tabs.

Usually an Add-In responds to this event in order to capture activity on an output tab they had previously created through a call to `Repository.AddTab()`.

Note that every loaded Add-In receives this event for every click on an output tab in Enterprise Architect, irrespective of whether the Add-In created that tab. Add-Ins should therefore check the `TabName` parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

Syntax

EA_OnOutputItemClicked (Repository As EA.Repository, TabName As String, LineText As String, ID As Long)

The EA_OnOutputItemClicked function syntax has these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN Description: The name of the tab that the click occurred in. Usually this would have been created through 'Repository.AddTab()'.
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().

Return Value

None.

EA_OnOutputItemDoubleClicked

EA_OnOutputItemDoubleClicked events inform Add-Ins that the user has used the mouse to double-click on a list entry in one of the user-defined output tabs.

Usually an Add-In responds to this event in order to capture activity on an output tab they had previously created through a call to `Repository.AddTab()`.

Note that every loaded Add-In receives this event for every double-click on an output tab in Enterprise Architect, irrespective of whether the Add-In created that tab; Add-Ins should therefore check the `TabName` parameter supplied by this event to ensure that they are not responding to other Add-Ins' events.

Syntax

EA_OnOutputItemDoubleClicked (Repository As EA.Repository, TabName As String, LineText As String, ID As Long)

The EA_OnOutputItemClicked function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model; poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN Description: The name of the tab that the click occurred in; usually this would have been created through 'Repository.AddTab()'.
LineText	String Direction: IN Description: The text that had been supplied as the String parameter in the original call to 'Repository.WriteOutput()'.
ID	Long Direction: IN Description: The ID value specified in the original call to Repository.WriteOutput().

Return Value

None.

EA_ShowHelp

Add-Ins can use the EA_ShowHelp event to show a Help topic for a particular menu option. When the user has an Add-In menu option selected, pressing F1 can be related to the required Help topic by the Add-In and a suitable Help message shown.

This event is raised when the user presses F1 on a menu option that is not a parent menu.

Syntax

Sub EA_ShowHelp (Repository as EA.Repository, MenuLocation As String, MenuName as String, ItemName as String)

The EA_ShowHelp function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MenuLocation	String Direction: Description: A string representing the part of the user interface that brought up the menu. This can be Treeview, MainMenu or Diagram.
MenuName	String Direction: Description: The name of the parent menu for which sub-items are to be defined. In the case of the top-level menu this is an empty string.
ItemName	String Direction: Description: The name of the option actually clicked; for example, 'Create a New Invoice'.

Return Value

None.

Événements diffusés

Overview

Broadcast events are sent to all loaded Add-Ins. For an Add-In to receive the event, they must first implement the required automation event interface. If Enterprise Architect detects that the Add-In has the required interface, the event is dispatched to the Add-In.

MDG Events add 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.

Event Type
Add-In License Management Events
Custom Table Events
Compartment Events
Context Item Events
File Close Event
File New Event
File Open Event
Model Validation Events
On Tab Changed Event
Post Close Diagram Event
Post Initialization Event
Post New Events
Post Open Diagram Event
Pre-Deletion Events
Pre-Exit Instance (not currently used)
On the creation of new objects
Retrieve Model Template Event
Schema Composer Events
Tagged Value Events

Technology Events
Transformation Event

Add-In Gestion de Licence Événements

Enterprise Architect Add-Ins can respond to events associated with Add-In License Management.

License Management Events

Event
EA_AddinLicenseValidate
EA_AddinLicenseGetDescription
EA_GetSharedAddinName

EA_AddinLicenseValidate

When a user directly enters into the 'License Management' dialog a license key that doesn't match a Sparx Systems key, EA_AddinLicenseValidate is broadcast to all Enterprise Architect Add-Ins, providing them with a chance to use the Add-In key to determine the level of functionality to provide. When a key is retrieved from the Sparx Systems Keystore only the target Add-In will be called with the key.

For the Add-In to validate itself against this key, the Add-In's EA_AddinLicenseValidate handler should return confirmation that the license has been validated. As the EA_AddinLicenseValidate event is broadcast to all Add-Ins, one license can validate many Add-Ins.

If an Add-In elects to handle a license key by returning a confirmation to EA_AddinLicenseValidate, it is called upon to provide a description of the license key through the EA_AddinLicenseGetDescription event. If more than one Add-In elects to handle a license key, the first Add-In that returns a confirmation to EA_AddinLicenseValidate is queried for the license key description.

Syntax

Function EA_AddinLicenseValidate (Repository As EA.Repository, AddinKey As String) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
AddinKey	String Direction: IN Description: The Add-In license key that has been entered in the 'License Management' dialog.

Return Value

Returns True if the license key is validated for the current Add-In. Returns False otherwise.

EA_AddinLicenseGetDescription

Before the Enterprise Architect 'License Management' dialog is displayed, EA_AddInLicenseGetDescription is sent once for each Add-In key to the first Add-In that elected to handle that key.

The value returned by EA_AddinLicenseGetDescription is used as the key's plain text description.

Syntax

Function EA_AddinLicenseGetDescription (Repository as EA.Repository, AddinKey as String) As String

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
AddinKey	String Direction: IN Description: The Add-In license key that Enterprise Architect requires a description for.

Return Value

A String containing a plain text description of the provided AddinKey.

EA_GetSharedAddinName

As an Add-In writer you can distribute keys to your Add-In via the Enterprise Architect Keystore, provided that your keys are added using a prefix that allows the system to identify the Add-In to which they belong.

EA_GetSharedAddinName is called to determine what prefix the Add-In is using. If a matching key is found in the keystore the 'License Management' dialog will display the name returned by EA_AddinLicenseGetDescription to your users. Finally, when the user selects a key, that key will be passed to your Add-In to validate by calling EA_AddinLicenseValidate.

Syntax

Function EA_GetSharedAddinName (Repository as EA.Repository) As String

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.

Return Value

A String containing a product name code for the provided Add-In, such as MYADDIN. This will be shown in plain text in any keys added to the keystore.

Notes

Shared Add-In keys have the format:

EASK-YOURCODE-REALKEY

- EASK - Constant string that identifies a shared key for an Enterprise Architect Add-In
- YOURCODE - The code you select and verify with us:
 - Displayed to the administrator of the keystore
 - Recommended length of 6-10 characters
 - Contains ASCII characters 33-126, except for '-' (45)
- REALKEY - Encoding of the actual key or checksums
 - Recommended length of 8-32 characters
 - Contains ASCII characters 33-126

We recommend that you contact Sparx Systems directly with proposed values to ensure that you don't clash with any other Add-Ins.

For example, these keys would all be interpreted as belonging to an Add-In returning MYADDIN from this function:

- EASK-MYADDIN-Test
- EASK-MYADDIN-{7AC4D426-9083-4fa2-93B7-25E2B7FB8DC5}
- EASK-MYADDIN-7AC4D426-9083-4fa2-93B7
- EASK-MYADDIN-25E2B7FB8DC5
- EASK-MYADDIN-2hDfHKA5jf0GAjn92UvqAnxwC13dxQGJtH7zLHJ9Ym8=

Tableau personnalisé Événements

The Custom Table element has an Operation called 'script', reserved for script execution, that can be used in two different, mutually exclusive ways, either:

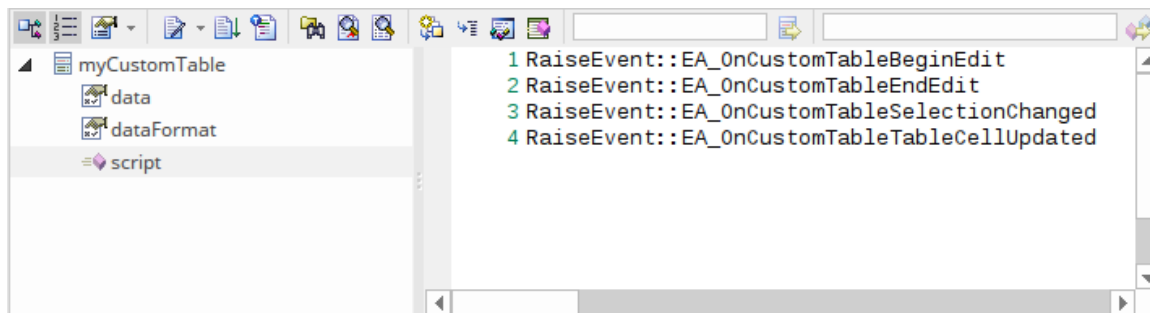
- To contain a script in JavaScript that can be executed from the element context menu; see the *Custom Table Artifact* Help topic, or
- To contain RaiseEvent broadcast calls to trigger actions from an Add-In written to read or update the Custom Table

Broadcasts

There are four reserved Add-In broadcast events that can only be enabled by listing the event in the 'script' Operation of the Custom Table element. To raise the broadcast events, list any or all of these broadcast calls in the operation named 'script'.

Syntax:

RaiseEvent::EA_OnCustomTableBeginEdit



EA_OnCustomTableBeginEdit

EA_OnCustomTableBeginEdit notifies Add-Ins that the Custom Table is beginning edit mode. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableBeginEdit (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableBeginEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Custom Table that is under edit: <ul style="list-style-type: none">ObjectID - A long value corresponding to the ElementID of the object

EA_OnCustomTableEndEdit

EA_OnCustomTableEndEdit notifies Add-Ins that a Custom Table element is ending edit mode. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableEndEdit (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableEndEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Custom Table that is under edit: <ul style="list-style-type: none">ObjectID - A long value corresponding to the ElementID of the object

Return Value

This function allows validation of the table data, and returns a Boolean value:

- True to save the current data in the grid, or
- False to abandon the current data

EA_OnCustomTableSelectionChanged

EA_OnCustomTableSelectionChanged notifies Add-Ins that a cell of the Custom Table has changed. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableSelectionChanged (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableSelectionChanged function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the Custom Table that has been changed: <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object• RowID - A long value corresponding to the selected row id• ColID - A long value corresponding to the selected column id

EA_OnCustomTableCellUpdated

EA_OnCustomTableCellUpdated notifies Add-Ins that a cell value has been updated. This broadcast event can only be enabled by the Custom Table's operation 'script' behavior.

Syntax

Function EA_OnCustomTableCellUpdated (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnCustomTableCellUpdated function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the Custom Table cell that has been changed: <ul style="list-style-type: none">• ObjectID - A long value corresponding to the ElementID of the object• RowID - A long value corresponding to the selected row id• ColID - A long value corresponding to the selected column id• Value - A variant value of the changed cell data

Compositeur de Schéma Événements

Enterprise Architect Add-Ins can respond to events associated with the Schema Composer to provide custom schema export formats.

The requirements for an Add-In to participate consist of implementing these three functions:

- EA_IsSchemaExporter
- EA_GetProfileInfo
- EA_GenerateFromSchema

EA_GenerateFromSchema

Respond to a 'Generate' request from the Schema Composer when using the profile type specified by the EA_IsSchemaExporter event. The SchemaComposer object can be used to traverse the schema. Export formats that have been requested by the user for generation will be listed in the exports parameter.

Syntax

Sub EA_GenerateFromSchema (Repository as EA.Repository, composer as EA.SchemaComposer, exports as String)

Parameter	Details
Repository	Type: EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
composer	Type: EA.SchemaComposer Direction: IN Description: Provides access to the types defined in the schema currently being generated. Use the <i>SchemaTypes</i> attribute to enumerate through the types and output to the appropriate export format.
exports	Type: String Direction: IN Description: Comma-separated list of export formats that the user has requested in the 'Generate' dialog.

Return Value

None.

EA_GetProfileInfo

Add-Ins can optionally implement this function to define the capabilities of the Schema Composer when working with the profile type specified by the EA_IsSchemaExporter event.

Syntax

Sub EA_GetProfileInfo (Repository as EA.Repository, profile as EA.SchemaProfile)

Parameter	Details
Repository	Type: EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
profile	Type: EA.SchemaProfile Direction: IN Description: An EA.SchemaProfile object representing the currently active profile type. Call the <i>SetCapability</i> function to enable or disable various capabilities of the Schema Composer. Call the <i>AddExportFormat</i> function to define additional export formats that this profile will support.

Return Value

None.

EA_IsSchemaExporter

Enterprise Architect Add-Ins can integrate with the Schema Composer by providing alternatives to offer users for the generation of schemas and sub models.

The Add-In must implement this function to be listed in the Schema Composer.

Syntax

Function EA_IsSchemaExporter(Repository as EA.Repository, ByRef displayName as String) As Boolean

Parameter	Details
Repository	Type: EA.Repository Direction: IN Description: An EA.Repository object representing the currently open model. Poll its members to retrieve model data and user interface status information.
displayName	Type: String Direction: OUT Description: The name of the custom schema set that will be provided by this Add-In.

Return Value

Return True to indicate that this Add-In will provide schema export functionality and be listed as a Schema Set when defining a new profile in the Schema Composer.

Compartment Events

Enterprise Architect Add-Ins can respond to various events associated with user-generated element compartments.

Compartment Broadcast Events

Event
EA_QueryAvailableCompartments
EA_GetCompartmentData

EA_QueryAvailableCompartments

This event occurs when Enterprise Architect's diagrams are refreshed. It is a request for the Add-In to provide a list of user-defined compartments.

The EA_GetCompartmentData event then queries each object for the data to display in each user-defined compartment.

Syntax

Function EA_QueryAvailableCompartments (Repository As EA.Repository) As Variant

The EA_QueryAvailableCompartments function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

A String containing a comma-separated list of user-defined compartments.

Example

Function EA_QueryAvailableCompartments(Repository As EA.Repository) As Variant

```
Dim sReturn As String
```

```
sReturn = ""
```

```
If m_FirstCompartmentVisible = True Then
```

```
    sReturn = sReturn + "first,"
```

```
End If
```

```
If m_SecondCompartmentVisible = True Then
```

```
    sReturn = sReturn + "second,"
```

```
End If
```

```
If m_ThirdCompartmentVisible = True Then
```

```
    sReturn = sReturn + "third,"
```

```
End If
```

```
If Len(sReturn) > 0 Then
```

```
    sReturn = Left(sReturn, Len(sReturn)-1)
```

```
End If
```

```
EA_QueryAvailableCompartments = sReturn
```

```
End Function
```


EA_GetCompartmentData

This event occurs when Enterprise Architect is instructed to redraw an element. It requests that the Add-In provide the data to populate the element's compartment.

Syntax

Function EA_GetCompartmentData (Repository As EA.Repository, sCompartment As String, sGUID As String, oType As EA.ObjectType) As Variant

The EA_QueryAvailableCompartments function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
sCompartment	String Direction: IN Description: The name of the compartment for which data is being requested.
sGUID	String Direction: IN Description: The GUID of the element for which data is being requested.
oType	ObjectType Direction: IN Description: The type of the element for which data is being requested.

Return Value

A variant containing a formatted string. The format is illustrated in this example:

Example

Function EA_GetCompartmentData(Repository As EA.Repository, sCompartment As String, sGUID As String, oType As EA.ObjectType) As Variant

```
If Repository Is Nothing Then
    Exit Function
End If
```



```
Dim sCompartmentData As String
Dim oXML As MSXML2.DOMDocument
Dim Nodes As MSXML2.IXMLDOMNodeList
Dim Node1 As MSXML2.IXMLDOMNode
Dim Node As MSXML2.IXMLDOMNode
Dim sData As String

sCompartmentData = ""
Set oXML = New MSXML2.DOMDocument
sData = ""
On Error GoTo ERR_GetCompartmentData
oXML.loadXML (Repository.GetTreeXMLByGUID(sGUID))
Set Node1 = oXML.selectSingleNode("//ModelItem")
If Node1 Is Nothing Then
    Exit Function
End If

sCompartmentData = sCompartmentData + "Name=" + sCompartment + ";"
sCompartmentData = sCompartmentData + "OwnerGUID=" + sGUID + ";"
sCompartmentData = sCompartmentData + "Options=SkipIfOnDiagram&_eq_^1&_sc_^"
Select Case sCompartment
Case "parts"
Set Nodes = Node1.selectNodes("ModelItem(@Metatype=""Part'')")
For Each Node In Nodes
    sData = sData + "Data&_eq_^" + Node.Attributes.getNamedItem("Name").nodeValue + "&_sc_^"
    sData = sData + "GUID&_eq_^" + Node.Attributes.getNamedItem("GUID").nodeValue + "&_sc_^,"
Next
Case "ports"
Set Nodes = Node1.selectNodes("ModelItem(@Metatype=""Port'')")
For Each Node In Nodes
    sData = sData + "Data&_eq_^" + Node.Attributes.getNamedItem("Name").nodeValue + "&_sc_^"
    sData = sData + "GUID&_eq_^" + Node.Attributes.getNamedItem("GUID").nodeValue + "&_sc_^,"
Next
End Select
If there is no data to display, then don't return any compartment data
If sData <> "" Then
    sCompartmentData = sCompartmentData + "CompartmentData=" + sData + ";"
Else
    sCompartmentData = ""
End If
EA_GetCompartmentData = sCompartmentData
Exit Function
```

```
ERR_GetCompartmentData:  
EA_GetCompartmentData = ""  
End Function
```

Context Item Events

Enterprise Architect Add-Ins can respond to events associated with changing context.

Context Item Broadcast Events

Event
EA_OnContextItemChanged
EA_OnContextItemDoubleClicked
EA_OnNotifyContextItemModified

EA_OnContextItemChanged

EA_OnContextItemChanged notifies Add-Ins that a different item is now in context.

This event occurs after a user has selected an item anywhere in the Enterprise Architect GUI. Add-Ins that require knowledge of the current item in context can subscribe to this broadcast function. If `ot = otRepository`, then this function behaves in the same way as EA_FileOpen.

Syntax

Sub EA_OnContextItemChanged (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnContextItemChanged function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the <code>ot</code> parameter: <ul style="list-style-type: none"> • <code>ot (ObjectType)</code> - GUID value • <code>otElement</code> - Element.ElementGUID • <code>otPackage</code> - Package.PackageGUID • <code>otDiagram</code> - Diagram.DiagramGUID • <code>otAttribute</code> - Attribute.AttributeGUID • <code>otMethod</code> - Method.MethodGUID • <code>otConnector</code> - Connector.ConnectorGUID • <code>otRepository</code> - NOT APPLICABLE, the GUID is an empty string
ot	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.

Return Value

None.

EA_OnContextItemDoubleClicked

EA_OnContextItemDoubleClicked notifies Add-Ins that the user has double-clicked the item currently in context.

This event occurs when a user has double-clicked (or pressed the Enter key) on the item in context, either in a diagram, in the Browser window or in a custom compartment. Add-Ins to handle events can subscribe to this broadcast function.

Syntax

Function EA_OnContextItemDoubleClicked (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnContextItemDoubleClicked function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter: <ul style="list-style-type: none">• otElement - Element.ElementGUID• otPackage - Package.PackageGUID• otDiagram - Diagram.DiagramGUID• otAttribute - Attribute.AttributeGUID• otMethod - Method.MethodGUID• otConnector - Connector.ConnectorGUID
ot	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.

Return Value

- Return True to notify Enterprise Architect that the double-click event has been handled by an Add-In
- Return False to enable Enterprise Architect to continue processing the event

EA_OnNotifyContextItemModified

EA_OnNotifyContextItemModified notifies Add-Ins that the current context item has been modified.

This event occurs when a user has modified the context item. Add-Ins that require knowledge of when an item has been modified can subscribe to this broadcast function.

Syntax

Sub EA_OnNotifyContextItemModified (Repository As EA.Repository, GUID As String, ot as EA.ObjectType)

The EA_OnNotifyContextItemModified function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
GUID	String Direction: IN Description: Contains the GUID of the new context item. The value corresponds to these properties, depending on the value of the ot parameter: <ul style="list-style-type: none">• ot(ObjectType) - GUID value• otElement - Element.ElementGUID• otPackage - Package.PackageGUID• otDiagram - Diagram.DiagramGUID• otAttribute - Attribute.AttributeGUID• otMethod - Method.MethodGUID• otConnector - Connector.ConnectorGUID
ot	EA.ObjectType Direction: IN Description: Specifies the type of the new context item.

Return Value

None.

EA_FileClose

The EA_FileClose event enables the Add-In to respond to a File Close event. When Enterprise Architect closes an opened Model file, this event is raised and passed to all Add-Ins implementing this method.

This event occurs when the model currently opened within Enterprise Architect is about to be closed (when another model is about to be opened or when Enterprise Architect is about to shutdown).

Syntax

Sub EA_FileClose (Repository As EA.Repository)

The EA_FileClose function syntax contains this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_FileNew

The EA_FileNew event enables the Add-In to respond to a File New event. When Enterprise Architect creates a new model file, this event is raised and passed to all Add-Ins implementing this method.

The event occurs when the model being viewed by the Enterprise Architect user changes, for whatever reason (through user interaction or Add-In activity).

Syntax

Sub EA_FileNew (Repository As EA.Repository)

The EA_FileNew function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_FileOpen

The EA_FileOpen event enables the Add-In to respond to a File Open event. When Enterprise Architect opens a new model file, this event is raised and passed to all Add-Ins implementing this method.

The event occurs when the model being viewed by the Enterprise Architect user changes, for whatever reason (through user interaction or Add-In activity).

Syntax

Sub EA_FileOpen (Repository As EA.Repository)

The EA_FileOpen function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnPostCloseDiagram

EA_OnPostCloseDiagram notifies Add-Ins that a diagram has been closed.

Syntax

Function EA_OnPostCloseDiagram (Repository As EA.Repository, DiagramID As Integer)

The EA_OnPostCloseDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the Enterprise Architect model about to be closed. Poll its members to retrieve model data and user interface status information.
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was closed.

Return Value

None.

EA_OnPostInitialized

EA_OnPostInitialized notifies Add-Ins that the Repository object has finished loading and any necessary initialization steps can now be performed on the object.

For example, the Add-In can create an 'Output' tab using Repository.CreateOutputTab.

Syntax

Sub EA_OnPostInitialized (Repository As EA.Repository)

The EA_OnPostInitialized function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnPostOpenDiagram

EA_OnPostOpenDiagram notifies Add-Ins that a diagram has been opened.

Syntax

Function EA_OnPostOpenDiagram (Repository As EA.Repository, DiagramID As Integer)

The EA_OnPostOpenDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
DiagramID	Integer Direction: IN Description: Contains the Diagram ID of the diagram that was opened.

Return Value

None.

EA_OnPostTransform

EA_OnPostTransform notifies Add-Ins that an MDG transformation has taken place with the output in the specified target Package.

This event occurs when a user runs an MDG transform on one or more target Packages; the notification is provided for each transform/target Package immediately after all transform processes have completed.

Syntax

Function EA_OnPostTransform (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostTransform function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty Objects for the transform performed: <ul style="list-style-type: none">• Transform: A string value corresponding to the name of the transform used• PackageID: A long value corresponding to Package.PackageID of the destination Package

Return Value

Reserved for future use.

EA_OnPreExitInstance

EA_OnPreExitInstance is not currently used.

Syntax

Sub EA_OnPreExitInstance (Repository As EA.Repository)

The EA_OnPreExitInstance function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

None.

EA_OnRetrieveModelTemplate

EA_OnRetrieveModelTemplate requests that an Add-In pass a model template to Enterprise Architect. This event occurs when a user executes the 'Add a New Model Using Wizard' command to add a model that has been defined by an MDG Technology.

Syntax

Function EA_OnRetrieveModelTemplate (Repository As EA.Repository, sLocation As String) As String

The EA_OnRetrieveModelTemplate function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
sLocation	String Direction: IN Description: The name of the template requested; this should match the location attribute in the <ModelTemplates> section of an MDG Technology File.

Return Value

Return a string containing the XMI export of the model that is being used as a template.

Return an empty string if access to the template is denied; the Add-In is to handle user notification of the error.

Example

```
Public Function EA_OnRetrieveModelTemplate(ByRef Rep As EA.Repository, ByRef sLocation As String) As String
Dim sTemplate As String
Select Case sLocation
Case "Templates\Template1.xml"
sTemplate = My.Resources.Template1
Case "Templates\Template2.xml"
sTemplate = My.Resources.Template2
Case "Templates\Template3.xml"
sTemplate = My.Resources.Template3
Case Else
MsgBox("Path for " & sLocation & " not found")
sTemplate = ""
```

End Select

EA_OnRetrieveModelTemplate = sTemplate

End Function

EA_OnTabChanged

EA_OnTabChanged notifies Add-Ins that the currently open tab has changed.

Diagrams do not generate the message when they are first opened - use the broadcast event EA_OnPostOpenDiagram for this purpose.

Syntax

Function EA_OnTabChanged (Repository As EA.Repository, TabName As String, DiagramID As Integer)

The EA_OnTabChanges function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
TabName	String Direction: IN Description: The name of the tab to which focus has been switched.
DiagramID	Long Direction: IN Description: The diagram ID, or 0 if switched to an Add-In tab.

Return Value

None

EA_LoadWindowManager

Enterprise Architect provides a set of Portals, each of which is a collection of shortcuts and information on performing specific areas of work on a project. The Portals help both new and experienced users quickly identify and set up the facilities they most often use in their assigned tasks.

You can add your own Portal to the system-installed set, to provide a convenient and concise call-up of one or more groups of facilities available in your Add-In.

Syntax

Function EA_Connect (Repository As EA.Repository) As String

The EA_Connect function syntax has this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Example Code

```
public String EA_LoadWindowManager(EA.Repository Repository)
{
    return Resource1.WindowManager;
}
```

Where Resource1.WindowManager is a resource file with these contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<perspectives>
<perspective name="Add-In">
<category name="Add-In" type="commandlist" projectrequired="true">
<item name="Hello World" command="CallAddin" addin="CS_AddinFramework" function="HelloWorld"/>
<item name="Model Dump" command="CallScript" group="Local Scripts" script="JScript - Recursive Model Dump
Example"/>
</category>
<category name="Open Diagrams" type="currentdiagramlist" state = "open"/>
<category name="Recent Diagrams" type="recentdiagramlist" state = "open"/>
<category name="Other Windows" type="otherwindowlist" state = "open"/>
</perspective>
</perspectives>
```

Note that the Add-In cannot specify the icon used.

Modèle Validation Événements

Perform Model Validation from an Add-In

Using Enterprise Architect broadcasts, it is possible to define a set of rules that are evaluated when the user instructs Enterprise Architect to perform model validation. An Add-In that performs model validation would involve these broadcast events.

Command	Detail
EA_OnInitializeUserRules	EA_OnInitializeUserRules is intercepted in order to define rule categories and rules.
EA_OnStartValidation	EA_OnStartValidation can be intercepted to perform any required processing prior to validation.
EA_OnEndValidation	EA_OnEndValidation can be intercepted to perform any required clean-up after validation has completed.
Validate Request	These functions intercept each request to validate an individual element, Package, diagram, connector, attribute and method.
Validate Element	EA_OnRunElementRule
Validate Package	EA_OnRunPackageRule
Validate Diagram	EA_OnRunDiagramRule
Validate Connector	EA_OnRunConnectorRule
Validate Attribute	EA_OnRunAttributeRule
Validate Method	EA_OnRunMethodRule
Validate Parameter	EA_OnRunParameterRule

EA_OnInitializeUserRules

EA_OnInitializeUserRules is called on Enterprise Architect start-up and requests that the Add-In provide Enterprise Architect with a rule category and list of rule IDs for model validation.

This function must be implemented by any Add-In that is to perform its own model validation. It must call Project.DefineRuleCategory once and Project.DefineRule for each rule; these functions are described in the *Project Interface* topic.

Syntax

Sub EA_OnInitializeUserRules (Repository As EA.Repository)

The EA_OnInitializeUserRules function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

EA_OnStartValidation

EA_OnStartValidation notifies Add-Ins that a user has invoked the model validation command from Enterprise Architect.

Syntax

Sub EA_OnStartValidation (Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnStartValidation function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Args	ParamArray of Variant Direction: IN Description: Contains a list of Rule Categories that are active for the current invocation of model validation.

EA_OnEndValidation

EA_OnEndValidation notifies Add-Ins that model validation has completed.

Use this event to arrange any clean-up operations arising from the validation.

Syntax

Sub EA_OnEndValidation (Repository As EA.Repository, ParamArray Args() as Variant)

The EA_OnEndValidation function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Args	ParamArray of Variant Direction: IN Description: Contains a list of Rule Categories that were active for the invocation of model validation that has just completed.

EA_OnRunElementRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each element in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given element, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunElementRule (Repository As EA.Repository, RuleID As String, Element As EA.Element)

The EA_OnRunElementRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
Element	EA.Element Direction: IN Description: The element to potentially perform validation on.

EA_OnRunPackageRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each Package in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given Package, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunPackageRule (Repository As EA.Repository, RuleID As String, PackageID As Long)

The EA_OnRunElementRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' method.
PackageID	Long Direction: IN Description: The ID of the Package to potentially perform validation on. Use the 'Repository.GetPackageByID' method to retrieve the Package object.

EA_OnRunDiagramRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each diagram in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given diagram, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunDiagramRule (Repository As EA.Repository, RuleID As String, DiagramID As Long)

The EA_OnRunDiagramRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
DiagramID	Long Direction: IN Description: The ID of the diagram to potentially perform validation on. Use the Repository.GetDiagramByID method to retrieve the diagram object.

EA_OnRunConnectorRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each connector in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given connector, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunConnectorRule (Repository As EA.Repository, RuleID As String, ConnectorID As Long)

The EA_OnRunConnectorRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
ConnectorID	Long Direction: IN Description: The ID of the connector to potentially perform validation on. Use the 'Repository.GetConnectorByID' method to retrieve the connector object.

EA_OnRunAttributeRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each attribute in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given attribute, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunAttributeRule (Repository As EA.Repository, RuleID As String, AttributeGUID As String, ObjectID As Long)

The EA_OnRunAttributeRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
AttributeGUID	String Direction: IN Description: The GUID of the attribute to potentially perform validation on. Use the 'Repository.GetAttributeByGuid' method to retrieve the attribute object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given attribute. Use the 'Repository.GetElementByID' method to retrieve the object.

EA_OnRunMethodRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each method in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given method, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunMethodRule (Repository As EA.Repository, RuleID As String, MethodGUID As String, ObjectID As Long)

The EA_OnRunMethodRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
MethodGUID	String Direction: IN Description: The GUID of the method to potentially perform validation on. Use the 'Repository.GetMethodByGuid' method to retrieve the method object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given method. Use the 'Repository.GetElementByID' method to retrieve the object.

EA_OnRunParameterRule

This event is triggered once for each rule defined in EA_OnInitializeUserRules to be performed on each parameter in the selection being validated.

If you don't want to perform the rule defined by RuleID on the given parameter, then simply return without performing any action.

On performing any validation, if a validation error is found, use the Repository.ProjectInterface.PublishResult method to notify Enterprise Architect.

Syntax

Sub EA_OnRunParameterRule (Repository As EA.Repository, RuleID As String, ParameterGUID As String, MethodGUID As String, ObjectID As Long)

The EA_OnRunMethodRule function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleID	String Direction: IN Description: The ID that was passed into the 'Project.DefineRule' command.
ParameterGUID	String Direction: IN Description: The GUID of the parameter to potentially perform validation on. Use this to retrieve the parameter by iterating through the 'Method.Parameters' collection.
MethodGUID	String Direction: IN Description: The GUID of the method that owns the given parameter. Use the 'Repository.GetMethodByGuid' method to retrieve the method object.
ObjectID	Long Direction: IN Description: The ID of the object that owns the given parameter. Use the 'Repository.GetElementByID' method to retrieve the object.

Exemple de validation Modèle

This example code is written in C# and provides a skeleton model validation implementation that you might want to use as a starting point in writing your own model validation rules.

Main.cs

```
using System;
namespace myAddin
{
    public class Main
    {
        public Rules theRules;
        public Main()
        {
            theRules = new Rules();
        }
        public string EA_Connect(EA.Repository Repository)
        {
            return "";
        }
        public void EA_Disconnect()
        {
            GC.Collect();
            GC.WaitForPendingFinalizers();
        }
        private bool IsProjectOpen(EA.Repository Repository)
        {
            try
            {
                EA.Collection c = Repository.Models;
                return true;
            }
            catch
            {
                return false;
            }
        }
        public object EA_GetMenuItems(EA.Repository Repository, string MenuLocation, string MenuName)
        {
            switch (MenuName)
            {
```

```
        case "":
            return "-&myAddin";
        case "-&myAddin":
            string() ar = { "&Test" };
            return ar;
    }
    return "";
}

public void EA_GetMenuState(EA.Repository Repository, string MenuLocation, string MenuName,
string ItemName, ref bool IsEnabled, ref bool IsChecked)
{
    // if no open project, disable all menu options
    if (IsProjectOpen(Repository))
        IsEnabled = true;
    else
        IsEnabled = false;
}

public void EA_MenuClick(EA.Repository Repository, string MenuLocation, string MenuName, string
ItemName)
{
    switch (ItemName)
    {
        case "&Test";
            DoTest(Repository);
            break;
    }
}

public void EA_OnInitializeUserRules(EA.Repository Repository)
{
    if (Repository != null)
    {
        theRules.ConfigureCategories(Repository);
        theRules.ConfigureRules(Repository);
    }
}

public void EA_OnRunElementRule(EA.Repository Repository, string RuleID, EA.Element element)
{
    theRules.RunElementRule(Repository, RuleID, element);
}

public void EA_OnRunDiagramRule(EA.Repository Repository, string RuleID, long IDiagramID)
{
    theRules.RunDiagramRule(Repository, RuleID, IDiagramID);
}
```



```
    }
    public void EA_OnRunConnectorRule(EA.Repository Repository, string RuleID, long IConnectorID)
    {
        theRules.RunConnectorRule(Repository, RuleID, IConnectorID);
    }
    public void EA_OnRunAttributeRule(EA.Repository Repository, string RuleID, string AttGUID, long IObjectID)
    {
        return;
    }
    public void EA_OnDeleteTechnology(EA.Repository Repository, EA.EventProperties Info)
    {
        return;
    }
    public void EA_OnImportTechnology(EA.Repository Repository, EA.EventProperties Info)
    {
        return;
    }
    private void DoTest(EA.Repository Rep)
    {
        // TODO: insert test code here
    }
}
}
```

Rules.cs

```
using System;
using System.Collections;
namespace myAddin
{
    public class Rules
    {
        private string m_sCategoryID;
        private System.Collections.ArrayList m_RuleIDs;
        private System.Collections.ArrayList m_RuleIDEx;
        private const string cRule01 = "Rule01";
        private const string cRule02 = "Rule02";
        private const string cRule03 = "Rule03";
        // TODO: expand this list as much as necessary
        public Rules()
        {
```

```
m_RuleIDs = new System.Collections.ArrayList();
m_RuleIDEx = new System.Collections.ArrayList();
}
private string LookupMap(string sKey)
{
    return DoLookupMap(sKey, m_RuleIDs, m_RuleIDEx);
}
private string LookupMapEx(string sRule)
{
    return DoLookupMap(sRule, m_RuleIDEx, m_RuleIDs);
}
private string DoLookupMap(string sKey, ArrayList arrValues, ArrayList arrKeys)
{
    if (arrKeys.Contains(sKey))
        return arrValues(arrKeys.IndexOf(sKey)).ToString();
    else
        return "";
}
private void AddToMap(string sRuleID, string sKey)
{
    m_RuleIDs.Add(sRuleID);
    m_RuleIDEx.Add(sKey);
}
private string GetRuleStr(string sRuleID)
{
    switch (sRuleID)
    {
        case cRule01:
            return "Error Message 01";
        case cRule02:
            return "Error Message 02";
        case cRule03:
            return "Error Message 03";
        // TODO: add extra cases as much as necessary
    }
    return "";
}
public void ConfigureCategories(EA.Repository Repository)
{
    EA.Project Project = Repository.GetProjectInterface();
    m_sCategoryID = Project.DefineRuleCategory("Enterprise Collaboration Architecture (ECA) Rules");
}
}
```

```
public void ConfigureRules(EA.Repository Repository)
{
    EA.Project Project = Repository.GetProjectInterface();
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError, GetRuleStr(cRule01)),
cRule01);
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError, GetRuleStr(cRule02)),
cRule02);
    AddToMap(Project.DefineRule(m_sCategoryID, EA.EnumMVErrortype.mvError, GetRuleStr(cRule03)),
cRule03);
    // TODO: expand this list
}
public void RunConnectorRule(EA.Repository Repository, string sRuleID, long lConnectorID)
{
    EA.Connector Connector = Repository.GetConnectorByID((int)lConnectorID);
    if (Connector != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule02:
                // TODO: perform rule 2 check
                break;
                // TODO: add more cases
            }
        }
    }
public void RunDiagramRule(EA.Repository Repository, string sRuleID, long lDiagramID)
{
    EA.Diagram Diagram = Repository.GetDiagramByID((int)lDiagramID);
    if (Diagram != null)
    {
        switch (LookupMapEx(sRuleID))
        {
            case cRule03:
                // TODO: perform rule 3 check
                break;
                // TODO: add more cases
            }
        }
    }
public void RunElementRule(EA.Repository Repository, string sRuleID, EA.Element Element)
{
    if (Element != null)
    {
```

```
switch (LookupMapEx(sRuleID))
{
    case cRule01:
        DoRule01(Repository, Element);
        break;
    // TODO: add more cases
}
}
}
private void DoRule01(EA.Repository Repository, EA.Element Element)
{
    if (Element.Stereotype != "myStereotype")
        return;
    // TODO: validation logic here
    // report validation errors
    EA.Project Project = Repository.GetProjectInterface();
    Project.PublishResult(LookupMap(cRule01), EA.EnumMVErrorType.mvError, GetRuleStr(cRule01));
}
}
}
```

Post-New Events

Enterprise Architect Add-Ins can respond to the creation of new elements, connectors, objects, attributes, methods and Packages using these broadcast events:

Post-New Broadcast Events

Event
EA_OnPostNewElement
EA_OnPostNewConnector
EA_OnPostNewDiagram
EA_OnPostNewDiagramObject
EA_OnPostNewAttribute
EA_OnPostNewMethod
EA_OnPostNewPackage
EA_OnPostNewGlossaryTerm

EA_OnPostNewElement

EA_OnPostNewElement notifies Add-Ins that a new element has been created on a diagram. It enables Add-Ins to modify the element upon creation.

This event occurs after a user has dragged a new element from the Toolbox or 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately after the element is added to the model.

Set Repository.SuppressEADialogs to True to suppress Enterprise Architect from showing its default 'Properties' dialog.

Syntax

Function EA_OnPostNewElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new element: <ul style="list-style-type: none">• ElementID: A long value corresponding to Element.ElementID

Return Value

Return True if the element has been updated during this notification. Return False otherwise.

EA_OnPostNewConnector

EA_OnPostNewConnector notifies Add-Ins that a new connector has been created on a diagram. It enables Add-Ins to modify the connector upon creation.

This event occurs after a user has dragged a new connector from the Toolbox or 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately after the connector is added to the model.

Syntax

Function EA_OnPostNewConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewConnector function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new connector: <ul style="list-style-type: none">ConnectorID: A long value corresponding to Connector.ConnectorID

Return Value

Return True if the connector has been updated during this notification. Return False otherwise.

EA_OnPostNewDiagram

EA_OnPostNewDiagram notifies Add-Ins that a new diagram has been created. It enables Add-Ins to modify the diagram upon creation.

Syntax

Function EA_OnPostNewDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new diagram: <ul style="list-style-type: none">• DiagramID: A long value corresponding to Diagram.PackageID

Return Value

Return True if the diagram has been updated during this notification. Return False otherwise.

EA_OnPostNewDiagramObject

EA_OnPostNewDiagramObject notifies Add-Ins that a new object has been created on a diagram. It enables Add-Ins to modify the object upon creation.

This event occurs after a user has dragged a new object directly from the Browser window or from the 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately after the object is added to the diagram.

Syntax

Function EA_OnPostNewDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the new element: <ul style="list-style-type: none">• ID: A long value corresponding to the ElementID of the object that has been added to the diagram• DiagramID: A long value corresponding to the DiagramID of the diagram to which the object has been added• DUID: A string value for the DUID; can be used with Diagram.GetDiagramObjectByID to retrieve the new DiagramObject

Return Value

Return True if the element has been updated during this notification. Return False otherwise.

EA_OnPostNewAttribute

EA_OnPostNewAttribute notifies Add-Ins that a new attribute has been created on a diagram. It enables Add-Ins to modify the attribute upon creation.

This event occurs when a user creates a new attribute on an element by either drag-and-dropping from the Browser window, using the 'Attributes' tab of the Features window, or using the in-place editor on the diagram. The notification is provided immediately after the attribute is created.

Syntax

Function EA_OnPostNewAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new attribute: <ul style="list-style-type: none">AttributeID: A long value corresponding to Attribute.AttributeID

Return Value

Return True if the attribute has been updated during this notification. Return False otherwise.

EA_OnPostNewMethod

EA_OnPostNewMethod notifies Add-Ins that a new method has been created on a diagram. It enables Add-Ins to modify the method upon creation.

This event occurs when a user creates a new method on an element by either drag-dropping from the Browser window, using the method's 'Properties' dialog, or using the in-place editor on the diagram. The notification is provided immediately after the method is created.

Syntax

Function EA_OnPostNewMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new method: <ul style="list-style-type: none">• MethodID: A long value corresponding to Method.MethodID

Return Value

Return True if the method has been updated during this notification. Return False otherwise.

EA_OnPostNewPackage

EA_OnPostNewPackage notifies Add-Ins that a new Package has been created on a diagram. It enables Add-Ins to modify the Package upon creation.

This event occurs when a user drags a new Package from the Toolbox or 'Resources' tab of the Browser window onto a diagram, or by selecting the New Package icon from the Browser window.

Syntax

Function EA_OnPostNewPackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewPackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the new Package: <ul style="list-style-type: none">PackageID: A long value corresponding to Package.PackageID

Return Value

Return True if the Package has been updated during this notification. Return False otherwise.

EA_OnPostNewGlossaryTerm

EA_OnPostNewGlossaryTerm notifies Add-Ins that a new glossary term has been created. It enables Add-Ins to modify the glossary term upon creation.

The notification is provided immediately after the glossary term is added to the model.

Syntax

Function EA_OnPostNewGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPostNewGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the new glossary term: <ul style="list-style-type: none">• TermID: A string value corresponding to Term.TermID• Term: A string value corresponding to the name of the glossary term being created• Meaning: A string value corresponding to meaning of the glossary term being created

Return Value

Return True if the glossary term has been updated during this notification. Return False otherwise.

Pre-Deletion Events

Enterprise Architect Add-Ins can respond to requests to delete elements, attributes, methods, connectors, diagrams, Packages and glossary terms using these broadcast events:

Pre-Deletion Broadcast Events

Event
EA_OnPreDeleteElement
EA_OnPreDeleteAttribute
EA_OnPreDeleteMethod
EA_OnPreDeleteConnector
EA_OnPreDeleteDiagram
EA_OnPreDeletePackage
EA_OnPreDeleteGlossaryTerm
EA_OnPreDeleteTechnology (Deprecated)

EA_OnPreDeleteElement

EA_OnPreDeleteElement notifies Add-Ins that an element is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the element.

This event occurs when a user deletes an element from the Browser window or on a diagram. The notification is provided immediately before the element is deleted, so that the Add-In can disable deletion of the element.

Syntax

Function EA_OnPreDeleteElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the element to be deleted: <ul style="list-style-type: none">ElementID: A long value corresponding to Element.ElementID

Return Value

- Return True to enable deletion of the element from the model
- Return False to disable deletion of the element

EA_OnPreDeleteAttribute

EA_OnPreDeleteAttribute notifies Add-Ins that an attribute is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the attribute.

This event occurs when a user attempts to permanently delete an attribute from the Browser window. The notification is provided immediately before the attribute is deleted, so that the Add-In can disable deletion of the attribute.

Syntax

Function EA_OnPreDeleteAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the attribute to be deleted: <ul style="list-style-type: none">AttributeID: A long value corresponding to Attribute.AttributeID

Return Value

- Return True to enable deletion of the attribute from the model
- Return False to disable deletion of the attribute

EA_OnPreDeleteMethod

EA_OnPreDeleteMethod notifies Add-Ins that a method (operation) is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the method.

This event occurs when a user attempts to permanently delete a method from the Browser window. The notification is provided immediately before the method is deleted, so that the Add-In can disable deletion of the method.

Syntax

Function EA_OnPreDeleteMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the method to be deleted: <ul style="list-style-type: none">MethodID: A long value corresponding to Method.MethodID

Return Value

- Return True to enable deletion of the method from the model
- Return False to disable deletion of the method

EA_OnPreDeleteConnector

EA_OnPreDeleteConnector notifies Add-Ins that a connector is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the connector.

This event occurs when a user attempts to permanently delete a connector on a diagram. The notification is provided immediately before the connector is deleted, so that the Add-In can disable deletion of the connector.

Syntax

Function EA_OnPreDeleteConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteConnector function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the connector to be deleted: <ul style="list-style-type: none">ConnectorID: A long value corresponding to Connector.ConnectorID

Return Value

- Return True to enable deletion of the connector from the model
- Return False to disable deletion of the connector

EA_OnPreDeleteDiagram

EA_OnPreDeleteDiagram notifies Add-Ins that a diagram is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the diagram.

This event occurs when a user attempts to permanently delete a diagram from the Browser window. The notification is provided immediately before the diagram is deleted, so that the Add-In can disable deletion of the diagram.

Syntax

Function EA_OnPreDeleteDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the diagram to be deleted: <ul style="list-style-type: none">DiagramID: A long value corresponding to Diagram.DiagramID

Return Value

- Return True to enable deletion of the diagram from the model
- Return False to disable deletion of the diagram

EA_OnPreDeleteDiagramObject

EA_OnPreDeleteDiagramObject notifies Add-Ins that a diagram object is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the element.

This event occurs when a user attempts to permanently delete an element from a diagram. The notification is provided immediately before the element is deleted, so that the Add-In can disable deletion of the element.

Syntax

Function EA_OnPreDeleteDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the element to be deleted: <ul style="list-style-type: none">• ID: A long value corresponding to DiagramObject.ElementID

Return Value

- Return True to enable deletion of the element from the model
- Return False to disable deletion of the element

EA_OnPreDeletePackage

EA_OnPreDeletePackage notifies Add-Ins that a Package is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the Package.

This event occurs when a user attempts to permanently delete a Package from the Browser window. The notification is provided immediately before the Package is deleted, so that the Add-In can disable deletion of the Package.

Syntax

Function EA_OnPreDeletePackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeletePackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the Package to be deleted: <ul style="list-style-type: none">PackageID: A long value corresponding to Package.PackageID

Return Value

- Return True to enable deletion of the Package from the model
- Return False to disable deletion of the Package

EA_OnPreDeleteGlossaryTerm

EA_OnPreDeleteGlossaryTerm notifies Add-Ins that a glossary term is to be deleted from the model. It enables Add-Ins to permit or deny deletion of the glossary term.

The notification is provided immediately before the glossary term is deleted, so that the Add-In can disable deletion of the glossary term.

Syntax

Function EA_OnPreDeleteGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the glossary term to be deleted: <ul style="list-style-type: none">• TermID: A long value corresponding to Term.TermID

Return Value

- Return True to enable deletion of the glossary term from the model
- Return False to disable deletion of the glossary term

Pre New-Object Events

When you create an Add-In, you can include broadcast events to intercept and respond to requests to create new objects, including elements, connectors, diagram objects, attributes, methods and Packages.

Events to intercept

Event
Creation of a new element
Creation of a new connector
Creation of a new diagram
Creation of a new diagram object
Creation of a new element by dropping onto a diagram from the Browser window.
Creation of a new attribute
Creation of a new method
Creation of a new Package
Creation of a new glossary term

EA_OnPreNewElement

EA_OnPreNewElement notifies Add-Ins that a new element is about to be created on a diagram. It enables Add-Ins to permit or deny creation of the new element.

This event occurs when a user drags a new element from the Toolbox or 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately before the element is created, so that the Add-In can disable addition of the element.

Syntax

Function EA_OnPreNewElement (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewElement function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the element to be created: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• FQStereotype: A string value corresponding to Element.FQStereotype• 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 enable addition of the new element to the model
- Return False to disable addition of the new element

EA_OnPreNewConnector

EA_OnPreNewConnector notifies Add-Ins that a new connector is about to be created on a diagram. It enables Add-Ins to permit or deny creation of a new connector.

This event occurs when a user drags a new connector from the Toolbox or 'Resources' tab of the Browser window, onto a diagram. The notification is provided immediately before the connector is created, so that the Add-In can disable addition of the connector.

Syntax

Function EA_OnPreNewConnector (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewConnector function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these 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 enable addition of the new connector to the model
- Return False to disable addition of the new connector

EA_OnPreNewDiagram

EA_OnPreNewDiagram notifies Add-Ins that a new diagram is about to be created. It enables Add-Ins to permit or deny creation of the new diagram.

The notification is provided immediately before the diagram is created, so that the Add-In can disable addition of the diagram.

Syntax

Function EA_OnPreNewDiagram (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewDiagram function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the diagram to be created: <ul style="list-style-type: none">• Type: A string value corresponding to Diagram.Type• ParentID: A long value corresponding to Diagram.ParentID• PackageID: A long value corresponding to Diagram.PackageID

Return Value

- Return True to enable addition of the new diagram to the model
- Return False to disable addition of the new diagram

EA_OnPreNewDiagramObject

EA_OnPreNewDiagramObject notifies Add-Ins that a new diagram object is about to be dropped on a diagram. It enables Add-Ins to permit or deny creation of the new object.

This event occurs when a user drags an object directly from the Enterprise Architect Browser window or from the 'Resources' tab of the Browser window onto a diagram. The notification is provided immediately before the object is created, so that the Add-In can disable addition of the object.

Syntax

Function EA_OnPreNewDiagramObject (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewDiagramObject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the object to be created: <ul style="list-style-type: none">• Type: A string value corresponding to the Type of object being added to the diagram• Stereotype: A string value corresponding to the Stereotype of the object being added to the diagram• ID: A long value corresponding to the ID of the element, Package or diagram being added to the diagram• DiagramID: A long value corresponding to the ID of the diagram to which the object is being added

Return Value

- Return True to enable addition of the object to the model
- Return False to disable addition of the object

EA_OnPreDropFromTree

When a user drags any kind of element from the Browser window onto a diagram, EA_OnPreDropFromTree notifies the Add-In that a new item is about to be dropped onto a diagram. The notification is provided immediately before the element is dropped, so that the Add-In can override the default action that would be taken for this drag.

Syntax

Function EA_OnPreDropFromTree (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDropFromTree function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the element to be created: <ul style="list-style-type: none">• ID: A long value of the type being dropped• Type: A string value corresponding to type of element being dropped• DiagramID: A long value corresponding to the ID of the diagram to which the element is being added• PositionX: The X coordinate into which the element is being dropped• PositionY: The Y coordinate into which the element is being dropped• DroppedID: A long value corresponding to the ID of the element the item has been dropped onto

Return Value

- Return True to allow the default behavior to be executed
- Return False to override this behavior

EA_OnPreNewAttribute

EA_OnPreNewAttribute notifies Add-Ins that a new attribute is about to be created on an element. It enables Add-Ins to permit or deny creation of the new attribute.

This event occurs when a user creates a new attribute on an element by either drag-dropping from the Browser window, using the 'Attributes' tab of the Features window, or using the in-place editor on the diagram. The notification is provided immediately before the attribute is created, so that the Add-In can disable addition of the attribute.

Syntax

Function EA_OnPreNewAttribute (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewAttribute function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the attribute to be created: <ul style="list-style-type: none">• Type: A string value corresponding to Attribute.Type• Stereotype: A string value corresponding to Attribute.Stereotype• ParentID: A long value corresponding to Attribute.ParentID• ClassifierID: A long value corresponding to Attribute.ClassifierID

Return Value

- Return True to enable addition of the new attribute to the model
- Return False to disable addition of the new attribute

EA_OnPreNewMethod

EA_OnPreNewMethod notifies Add-Ins that a new method is about to be created on an element. It enables Add-Ins to permit or deny creation of the new method.

This event occurs when a user creates a new method on an element by either drag-dropping from the Browser window, using the 'Operations' tab of the Features window, or using the in-place editor on the diagram. The notification is provided immediately before the method is created, so that the Add-In can disable addition of the method.

Syntax

Function EA_OnPreNewMethod (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewMethod function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the method to be created: <ul style="list-style-type: none">• Return Type: A string value corresponding to Method.ReturnType• Stereotype: A string value corresponding to Method.Stereotype• ParentID: A long value corresponding to Method.ParentID• ClassifierID: A long value corresponding to Method.ClassifierID

Return Value

- Return True to enable addition of the new method to the model
- Return False to disable addition of the new method

EA_OnPreNewPackage

EA_OnPreNewPackage notifies Add-Ins that a new Package is about to be created in the model. It enables Add-Ins to permit or deny creation of the new Package.

This event occurs when a user drags a new Package from the Toolbox or 'Resources' tab of the Browser window onto a diagram, or by selecting the New Package icon from the Browser window. The notification is provided immediately before the Package is created, so that the Add-In can disable addition of the Package.

Syntax

Function EA_OnPreNewPackage (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewPackage function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the Package to be created: <ul style="list-style-type: none">• Stereotype: A string value corresponding to Package.Stereotype• ParentID: A long value corresponding to Package.ParentID• DiagramID: A long value corresponding to the ID of the diagram to which the Package is being added

Return Value

- Return True to enable addition of the new Package to the model
- Return False to disable addition of the new Package

EA_OnPreNewGlossaryTerm

EA_OnPreNewGlossaryTerm notifies Add-Ins that a new glossary term is about to be created. It enables Add-Ins to permit or deny creation of the new glossary term.

The notification is provided immediately before the glossary term is created, so that the Add-In can disable addition of the element.

Syntax

Function EA_OnPreNewGlossaryTerm (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreNewGlossaryTerm function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the glossary term to be created: <ul style="list-style-type: none">• TermID: A string value corresponding to Term.TermID• Term: A string value corresponding to the name of the glossary term being created• Meaning: A string value corresponding to meaning of the glossary term being created

Return Value

- Return True to enable addition of the new glossary term to the model
- Return False to disable addition of the new glossary term


Valeur Étiquetée Événements

Enterprise Architect includes the Addin Broadcast Tagged Value type that allows an Add-In to respond to attempts to edit it. The function that is called depends on the type of object the Tagged Value is on.

Tagged Value Events

Event
EA_OnAttributeTagEdit
EA_OnConnectorTagEdit
EA_OnElementTagEdit
EA_OnMethodTagEdit

EA_OnAttributeTagEdit

EA_OnAttributeTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an attribute.

The Add-In displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.


Syntax

Sub EA_OnAttributeTagEdit (Repository As EA.Repository, AttributeID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnAttributeTagEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
AttributeID	Long Direction: IN Description: The ID of the attribute that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnConnectorTagEdit

EA_OnConnectorTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on a connector.

The Add-In displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax

Sub EA_OnConnectorTagEdit (Repository As EA.Repository, ConnectorID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnConnectorTagEdit function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
ConnectorID	Long Direction: IN Description: The ID of the connector that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnElementTagEdit

EA_OnElementTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an element.

The Add-In displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.


Syntax

Sub EA_OnElementTagEdit (Repository As EA.Repository, ObjectID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnElementTagEdit function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
ObjectID	Long Direction: IN Description: The ID of the object (element) that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

EA_OnMethodTagEdit

EA_OnMethodTagEdit is called when the user clicks the  button for a Tagged Value of type AddinBroadcast on an operation.

The Add-In displays fields to show and change the value and notes; this function provides the initial values for the Tagged Value notes and value, and takes on any changes on exit of the function.

Syntax

Sub EA_OnMethodTagEdit (Repository As EA.Repository, MethodID As Long, String TagName, String TagValue, String TagNotes)

The EA_OnMethodTagEdit function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
MethodID	Long Direction: IN Description: The ID of the method that this Tagged Value is on.
TagName	String Direction: IN Description: The name of the Tagged Value to edit.
TagValue	String Direction: INOUT Description: The current value of the tag; if the value is updated, the new value is stored in the repository on exit of the function.
TagNotes	String Direction: INOUT Description: The current value of the Tagged Value notes; if the value is updated, the new value is stored in the repository on exit of the function.

Technology Events

Enterprise Architect Add-Ins can respond to events associated with the use of MDG Technologies.

Technology Broadcast Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology
EA_OnPreDeleteTechnology (Deprecated)
EA_OnDeleteTechnology (Deprecated)
EA_OnImportTechnology (Deprecated)

EA_OnInitializeTechnologies

EA_OnInitializeTechnologies requests that an Add-In pass an MDG Technology to Enterprise Architect for loading. This event occurs on Enterprise Architect start up. Return your technology XML to this function and Enterprise Architect loads and enables it.

Syntax

Function EA_OnInitializeTechnologies (Repository As EA.Repository) As Object

The EA_OnInitializeTechnologies function syntax contains this parameter:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

Return the MDG Technology as a single XML string.

Example

```
Public Function EA_OnInitializeTechnologies(ByVal Repository As EA.Repository) As Object
    EA_OnInitializeTechnologies = My.Resources.MyTechnology
End Function
```

EA_OnPreActivateTechnology

EA_OnPreActivateTechnology notifies Add-Ins that an MDG Technology resource is about to be activated in the model.

This event occurs when a user selects to activate an MDG Technology resource in the model (by clicking on the Set Active button on the 'MDG Technologies' dialog or by selecting the technology in the list box in the Default Tools toolbar).

The notification is provided immediately after the user attempts to activate the MDG Technology, so that the Add-In can permit or disable activation of the Technology.

Syntax

Function EA_OnPreActivateTechnology (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreActivateTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the MDG Technology to be activated: <ul style="list-style-type: none">TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

- Return True to enable activation of the MDG Technology resource in the model
- Return False to disable activation of the MDG Technology resource

EA_OnPostActivateTechnology

EA_OnPostActivateTechnology notifies Add-Ins that an MDG Technology resource has been activated in the model.

This event occurs when a user activates an MDG Technology resource in the model (by clicking on the Set Active button on the 'MDG Technologies' dialog, or by selecting the technology in the list box in the Default Tools toolbar).

The notification is provided immediately after the user succeeds in activating the MDG Technology, so that the Add-In can update the Technology if necessary.

Syntax

Function EA_OnPostActivateTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnPostActivateTechnology function syntax contains these parameters:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects for the MDG Technology to be activated: <ul style="list-style-type: none">TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

Return True if the MDG Technology resource is updated during this notification. Return False otherwise.

EA_OnPreDeleteTechnology

Deprecated - refers to deleting a technology through the 'Resources' tab of the Browser window; this process is no longer recommended. See *Deploy An MDG Technology* for information on recommended methods for using technologies.

EA_OnPreDeleteTechnology notifies Add-Ins that an MDG Technology resource is about to be deleted from the model.

This event occurs when a user deletes an MDG Technology resource from the model.

The notification is provided immediately after the user confirms their request to delete the MDG Technology, so that the Add-In can disable deletion of the MDG Technology.

Related Broadcast Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Function EA_OnPreDeleteTechnology (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

The EA_OnPreDeleteTechnology function syntax contains these elements:

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains this EventProperty object for the MDG Technology to be deleted: <ul style="list-style-type: none">TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

- Return True to enable deletion of the MDG Technology resource from the model
- Return False to disable deletion of the MDG Technology resource

EA_OnDeleteTechnology

Deprecated - refers to deleting a technology through the 'Resources' tab of the Browser window; this process is no longer recommended. See *Deploy An MDG Technology* for information of recommended methods for using technologies.

EA_OnDeleteTechnology notifies Add-Ins that an MDG Technology resource has been deleted from the model.

This event occurs after a user has deleted an MDG Technology resource from the model. Add-Ins that require an MDG Technology resource to be loaded can catch this event to disable certain functionality.

Related Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Sub EA_OnDeleteTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnDeleteTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects: <ul style="list-style-type: none">• TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

None.

EA_OnImportTechnology

Deprecated - refers to importing a technology into the 'Resources' tab of the Browser window; this process is no longer recommended. See *Deploy An MDG Technology* for information of recommended methods for using technologies.

EA_OnImportTechnology notifies Add-Ins that you have imported an MDG Technology resource into the model.

This event occurs after you have imported an MDG Technology resource into the model. Add-Ins that require an MDG Technology resource to be loaded can catch this Add-In to enable certain functionality.

Related Events

Event
EA_OnInitializeTechnologies
EA_OnPreActivateTechnology
EA_OnPostActivateTechnology

Syntax

Sub EA_OnImportTechnology (Repository As EA.Repository, Info As EA.EventProperties)

The EA_OnImportTechnology function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects: <ul style="list-style-type: none"> TechnologyID: A string value corresponding to the MDG Technology ID

Return Value

None.

Technology Rules

The Technology Rules set of events provides hooks for Add-Ins to customize the behavior for their own modeling languages beyond that which can be specified through MDG Technologies alone. These events have been subdivided into categories to assist in exploring the events that are available.

Technology Rule Events

Events
The EARules_Initialize event is called for all Add-Ins during initialization. Specifying one or more profiles for which to define rules is a pre-requisite for all rule calls.
Diagram Appearance Rule events modify some facet of how elements are rendered on diagrams.
User Interface Rule events are used to show, hide or customize aspects of the user interface, or to customize available actions to work naturally with a language.

EARules_Initialize

EARules_Initialize enables Add-Ins to override internal behavior for one or more technologies.

This event occurs during Add-In initialization.

Syntax

Function EARules_Initialize (Repository As EA.Repository, RuleIndex As Integer, Base As String) As String

The EARules_Initialize function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
RuleIndex	Integer Direction: IN Description: Provides a count of the number of calls to this function. This allows you to define multiple rule sets without maintaining your own state for previous calls to this function.
Base	String Direction: OUT Description: This parameter can be assigned the name of an existing set of rules that will be treated as a superclass of the rule set defined by your Add-In. The rules are named by the profile name containing the stereotypes or diagram types that are being modified. You will usually use this value if you are extending the stereotypes within that profile. The customized rules that are built-in to Enterprise Architect are: <ul style="list-style-type: none"> • ArchiMate3.0 • BPMN2.0 • DMN1.1 • MARTE • Modelica • SPEM • SysML1.2 • SysML1.3 • SysML1.4 • SysPhS

Return Value

- Returns a non-empty string that matches the name of a profile or diagram profile to define rules for that profile; this function will be called again to allow rules for additional profiles
- Returns an empty string to specify that the Add-In does not define any additional technology-specific rules

Diagram Appearance Rule Events

Diagram Appearance Events

Event
EARules_ClosePartitionName is called to allow control over rendering a closing line after the name for an Activity Partition.
EARules_ElementDisplayName is called to allow alternative text to be displayed as the name during the rendering of an element.
EARules_GetCompartmentItem is called to allow altering the text to display for an item in a built-in compartment.
EARules_GetCompartmentName is called to determine the name displayed for built-in compartments.
EARules_GetNameUnderline is called by Enterprise Architect to determine if the name should be drawn with an underline.
EARules_GetPropertyString is called to provide alternative or additional text to the default element property string.
EARules_GetShapeScript is called to allow a custom shape script for elements without one defined.
EARules_ShowStereotype is called to allow control over stereotype visibility for the default rendering of an Activity Partition.
EARules_StereotypeDisplayName is called by Enterprise Architect to provide a custom keyword to display on the diagram in place of the stereotype name.

EARules_ClosePartitionName

EARules_ClosePartitionName allows an Add-In that is registered to provide rules for a language to determine if the name for an Activity Partition is displayed with a closing line.

This event occurs during diagram drawing.

Syntax

Function EARules_ClosePartitionName (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_ClosePartitionName function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.

Return Value

- Return a positive value to specify that the name should be closed
- Return zero to specify the name should not be closed
- Return a negative value to use the behavior from the base rules

EARules_ElementDisplayName

EARules_ElementDisplayName allows an Add-In registered to provide rules for a language to override the text displayed for the name in the default notation. An example of where you might want to use this is if an element should take its name automatically from another element.

This event occurs during the drawing of Activity Partitions and Shape Scripts.

Syntax

Function EARules_ElementDisplayName (Repository As EA.Repository, Language As String, Element as EA.Element, CurrentName as String, Base as Integer) As String

The EARules_ElementDisplayName function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
CurrentName	String Direction: IN Description: Specifies the name currently being used for this element.
Base	Integer Direction: OUT Description: Controls whether the base rules will be called if you return an empty value. A non-zero value and return of an empty string means that the parent rules will determine the display of the element name.

Return Value

A string to override the displayed name of the element on a diagram.

EARules_GetCompartmentItem

EARules_GetCompartmentName allows an Add-In registered to provide rules for a language to override the text displayed for individual items in a compartment.

This event occurs during diagram drawing.

Syntax

Function EARules_GetCompartmentItem (Repository As EA.Repository, Language As String, Element as EA.Element, Compartment as String, Item as String, Base as Integer) As String

The EARules_GetCompartmentItem function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
Compartment	String Direction: IN Description: Specifies the compartment being rendered.
Item	String Direction: IN Description: Specifies the item being rendered.
Base	Integer Direction: OUT Description: Controls whether the base rules will be called if you return an empty value. A non-zero value and return of an empty string means that the parent rules will determine the display of the compartment item.

Return Value

A string defining the text to be displayed for this compartment item.

EARules_GetCompartmentName

EARules_GetCompartmentName allows an Add-In registered to provide rules for a language to override the label displayed at the top of a compartment.

This event occurs during diagram drawing.

Syntax

Function EARules_GetCompartmentName (Repository As EA.Repository, Language As String, Element as EA.Element, Compartment as String, Base as Integer) As String

The EARules_GetCompartmentName function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
Compartment	String Direction: IN Description: Specifies the compartment being rendered.
Base	Integer Direction: OUT Description: Controls whether the base rules will be called if you return an empty value. A non-zero value and return of an empty string means that the parent rules will determine the display of the compartment name.

Return Value

A string defining the name to be rendered for the specified compartment.

EARules_GetNameUnderline

EARules_GetNameUnderline allows an Add-In registered to provide rules for a language to control if the name of an element is rendered with an underline.

This event occurs during diagram drawing.

Syntax

Function EARules_GetNameUnderline (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_GetNameUnderline function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.

Return Value

- Return a positive value to specify that the name should be underlined
- Return zero to specify that the name should not be underlined
- Return a negative value to use the behavior from the base rules

EARules_GetPropertyString

EARules_GetPropertyString allows an Add-In registered to provide rules for a language to override the text for the property string of an element. In standard UML notation this is rendered between '{' and '}' near the name.

This event occurs during diagram drawing.

Syntax

Function EARules_GetPropertyString (Repository As EA.Repository, Language As String, Element as EA.Element, Order as Integer) As String

The EARules_GetPropertyString function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
Order	Integer Direction: OUT Description: Controls the order in which text provided by base rules is added to the return value. <ul style="list-style-type: none">• Assign a negative value to place the base property string ahead of the return value• Assign a positive value to place the base property string after the return value• Assign zero to prevent the base property string from being displayed

Return Value

A string defining the contents of the property string used in the default element notation.

EARules_GetShapeScript

EARules_GetShapeScript allows an Add-In registered to provide rules for a language to customize the rendering of an element for the modeling language of the diagram being drawn.

This event occurs during diagram drawing.

Syntax

Function EARules_GetShapeScript (Repository As EA.Repository, Language As String, Element as EA.Element) As String

The EARules_GetShapeScript function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.

Return Value

- Return a string containing the Shape Script to use for this element
- Return an empty string to defer to the parent rules

EARules_ShowStereotype

EARules_ShowStereotype allows an Add-In registered to provide rules for a language to determine if the stereotype for an Activity Partition is displayed with the name.

This event occurs during diagram drawing.

Syntax

Function EARules_ShowStereotype (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_ShowStereotype function syntax contains these parameters:

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.

Return Value

- Return a positive value to specify that the name should be closed
- Return zero to specify that the name should not be closed
- Return a negative value to use the behavior from the base rules

EARules_StereotypeDisplayName

EARules_StereotypeDisplayName allows an Add-In registered to provide rules for a language to override the text displayed for the stereotype in the default notation. In standard UML notation, this is rendered between '«' and '»' near the name.

This event occurs during diagram drawing.

Syntax

Function EARules_StereotypeDisplayName (Repository As EA.Repository, Language As String, Stereotype as String, Base as Integer) As String

The EARules_StereotypeDisplayName function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Stereotype	String Direction: IN Description: Specifies the fully qualified name of the stereotype being queried.
Base	Integer Direction: OUT Description: Controls whether the base rules will be called if you return an empty value. A non-zero value and return of an empty string means that the parent rules will determine the display of the stereotype.

Return Value

A string to override the displayed name of the stereotype on a diagram.

User Interface Rule Events

User Interface Behavior Events

Event
EARules_AllowNesting is called to determine if creating a visual nesting of two elements on a diagram results in setting the ownership in the browser.
EARules_AppendChildDiagrams is called to build the context menu for adding child diagrams to an element.
EARules_AppendChildElements is called to build the context menu for adding child elements to an element.
EARules_CanOverrideStereotype is called to determine if the stereotype on a Type is assigned to a property when assigning the type.
EARules_CanProxy is called to determine if this element can be represented by another element.
EARules_CanReparent is called to determine if visually nesting on a diagram is intended to update ownership in the Browser.
EARules_CreateModel is called to allow creation of a wrapping element during creation of a diagram.
EARules_EnableElementProperty is called to determine if a particular property in the docked Properties window should allow edits for this element.
EARules_ForceLength is used to allow particular diagrams to require elements determined by EARules_IsAdjustable to have their length fixed to the size of the diagram.
EARules_AppendChildDiagrams is called to override a request for a UML diagram when the context specifies a different language is being used.
EARules_IsAdjustable is called to determine which elements can have their length automatically adjusted on diagrams where EARules_ForceLength has allowed the operation.
EARules_PropagateStereotype is called to determine if a classifier stereotype can be applied to an instance using that classifier.
EARules_ShowElementProperty is called to determine if a particular property in the docked Properties window should be displayed for this element.
EARules_ShowFrame is called to automatically insert the parent of the diagram as a frame.
EARules_ShowParentFrame is called to automatically insert the parent of the diagram as a frame.

EARules_AllowNesting

EARules_AllowNesting allows an Add-In registered to provide rules for a language to specify if two elements should be nested in the Browser window after being nested on a diagram.

This event occurs while a user drags one element onto another in a diagram.

Syntax

Function EARules_AllowNesting (Repository As EA.Repository, Language As String, Child as EA.Element, Parent as EA.Element, Diagram as EA.Diagram) As Integer

The EARules_AllowNesting function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Child	EA.Element Direction: IN Description: Specifies the element that has been graphically nested within another element.
Parent	EA.Element Direction: IN Description: Specifies the target element of a drag-and-drop operation within a diagram.
Diagram	EA.Diagram Direction: IN Description: The diagram that the move event has occurred on.

Return Value

- Return a positive value to allow the nesting to occur
- Return a negative value to use the base rules
- Return zero to prevent the nesting from occurring

EARules_AppendChildDiagrams

EARules_AppendChildDiagrams allows an Add-In registered to provide rules for a language to specify the list of items to be shown in the '[New Child Diagram]' menu.

This event occurs when a context menu is shown that includes 'New Child Diagram'.

Syntax

Function EARules_AppendChildDiagrams (Repository As EA.Repository, Language As String, Parent as EA.Element, Diagram as EA.Diagram, Order as Integer) As Variant

The EARules_AppendChildDiagrams function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Parent	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for new child diagrams.
Diagram	EA.Diagram Direction: IN Description: The diagram that is showing the parent element. If a user is showing a context menu outside a diagram, this could be null.
Order	EA.Diagram Direction: OUT Description: Allows controlling if and where the child diagrams specified in the parent rules are shown. A positive value means they will be shown after the items specified in this function. Zero means they are not shown at all. A negative value means that they are shown after the items specified in this function.

Return Value

This function supports returning either a single string with multiple items specified by a ';', or an array of strings.

Each item can be one of these:

- "-" - inserts a separator
- Any other text - shows that text as the item, and if the user clicks on it the Add-In is responsible for creating the requested diagram in EA_OnMenuClick

EARules_AppendChildElements

EARules_AppendChildElements allows an Add-In registered to provide rules for a language to specify the list of items to be shown in the 'New Child Element' menu.

This event occurs when a context menu is shown that includes 'New Child Element'.

Syntax

Function EARules_AppendChildElements (Repository As EA.Repository, Language As String, Parent as EA.Element, Diagram as EA.Diagram, Order as Integer) As Variant

The EARules_AppendChildElements function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Parent	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for new child diagrams.
Diagram	EA.Diagram Direction: IN Description: The diagram that is showing the parent element. If a user is showing a context menu outside a diagram, this could be null.
Order	EA.Diagram Direction: OUT Description: Allows the control of if and where the child diagrams specified in the parent rules are shown. A positive value means they will be shown after the items specified in this function. Zero means they are not shown at all. A negative value means that they are shown after the items specified in this function.

Return Value

This function supports returning either a single string with multiple items specified by a ';', or an array of strings.

Each item can be one of these:

- "-" - inserts a separator
- A valid toolbox string including an alias - should be of the form <profile>::<stereotype>(UML::<base>)=<alias>
- Any other text - shows that text as the item, and if the user clicks on it the Add-In is responsible for creating the requested element in EA_OnMenuClick

EARules_CanOverrideStereotype

EARules_CanOverrideStereotype allows an Add-In registered to provide rules for a language to control when the stereotype from a newly assigned type is propagated to the property.

This event occurs when a type is assigned to a property element.

Syntax

Function EARules_CanOverrideStereotype (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_CanOverrideStereotype function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for new child diagrams.

Return Value

- Return a positive value to allow the stereotype from the type to override the current stereotype
- Return zero to prevent any change to the element stereotype
- Return a negative value to use the behavior from the base rules

EARules_CanProxy

EARules_CanProxy allows an Add-In registered to provide rules for a language to specify that one element is a proxy for another.

This event should only be handled for rules extending BPMN.

Syntax

Function EARules_CanProxy (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_CanProxy function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Target	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for new child diagrams.

Return Value

- Return a positive value to specify that the target element is a proxy
- Return zero to specify that the target element is not a proxy
- Return a negative value to use the behavior from the base rules

EARules_CanReparent

EARules_CanReparent allows an Add-In registered to provide rules for a language to specify that the child diagrams for an element can be changed to other diagrams.

Syntax

Function EARules_CanReparent (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_CanReparent function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for new child diagrams.

Return Value

- Return a positive value to allow reparenting to occur
- Return zero to prevent reparenting
- Return a negative value to use the behavior from the base rules

EARules_CreateModel

EARules_CreateModel allows an Add-In registered to provide rules for a language to create a wrapping element when a new diagram is created.

This event occurs during diagram creation.

Syntax

Function EARules_CreateModel (Repository As EA.Repository, Language As String, Diagram as EA.Diagram) As Integer

The EARules_CreateModel function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Diagram	EA.Diagram Direction: IN Description: Specifies the diagram currently being created.

Return Value

- Return zero or a positive value if no further action is required
- Return a negative value to use the behavior from the base rule

EARules_EnableElementProperty

EARules_EnableElementProperty allows an Add-In registered to provide rules for a language to control if individual properties should be displayed as read-only in the docked Properties window.

This is called during selection of elements when the Properties window is visible.

Syntax

Function EARules_EnableElementProperty (Repository As EA.Repository, Language As String, Element as EA.Element, Namespace as String, Class as String, Property as String) As Integer

The EARules_EnableElementProperty function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
Namespace	String Direction: IN Description: Specifies the top level language this property comes from. This will be either "UML" or the name of a profile.
Class	String Direction: IN Description: Specifies the type this property was defined in. In the UML namespace that means the metaclass defined in UML. Otherwise it will be a stereotype.
Property	String Direction: IN Description: Specifies the metaclass or stereotype property to enable or disable.

Return Value

- Return a positive value to allow edits to the property

- Return zero to disable edits to the property
- Return a negative value to use the behavior from the base rules

EARules_ForceLength

EARules_ForceLength allows an Add-In registered to provide rules for a language to specify that some elements are sized to the width or height of the diagram.

This event occurs during diagram load and resize events.

Syntax

Function EARules_ForceLength (Repository As EA.Repository, Language As String, Diagram as EA.Diagram) As Integer

The EARules_ForceLength function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Diagram	EA.Diagram Direction: IN Description: Specifies the diagram currently being loaded.

Return Value

- Return a positive value to specify that this diagram should enforce the length of elements
- Return zero to specify that no elements should have their length enforced
- Return a negative value to use the behavior from the base rules

EARules_GetEquivalentDiagram

EARules_GetEquivalentDiagram allows an Add-In registered to provide rules for a language to override the diagram type created when a UML diagram would otherwise be created.

This event occurs during user requests to create a structure that includes a diagram.

Syntax

Function EARules_GetEquivalentDiagram (Repository As EA.Repository, Language As String, DiagramType as String) As String

The EARules_GetEquivalentDiagram function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
DiagramType	String Direction: IN Description: Specifies the type of diagram that has been requested.

Return Value

A string containing the qualified name of a diagram type to replace the UML diagram requested.

Return an empty string to allow the base rules to control the diagram type.

EARules_IsAdjustable

EARules_IsAdjustable allows an Add-In registered to provide rules for a language to specify which elements are sized to the width or height of the diagram.

This event occurs during diagram load and resize events.

Syntax

Function EARules_IsAdjustable (Repository As EA.Repository, Language As String, Element as EA.Element) As Integer

The EARules_IsAdjustable function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element.

Return Value

- Return a positive value to allow automatic resize to occur
- Return zero to prevent automatic resize
- Return a negative value to use the behavior from the base rules

EARules_PropagateStereotype

EARules_PropagateStereotype allows an Add-In registered to provide rules for a language to control if a particular stereotype from a classifier should be applied to an instance with that classifier.

This event occurs when a classifier is assigned to an instance.

Syntax

Function EARules_PropagateStereotype (Repository As EA.Repository, Language As String, Element as EA.Element, Stereotype as String) As Integer

The EARules_PropagateStereotype function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Element	EA.Element Direction: IN Description: Specifies the element showing a context menu that includes the option for applying the stereotype.
Stereotype	String Direction: IN Description: Specifies the qualified name of the stereotype to apply.

Return Value

- Return a positive value to allow the stereotype from the type to be applied
- Return zero to prevent the stereotype from being applied
- Return a negative value to use the behavior from the base rules

EARules_ShowElementProperty

EARules_ShowElementProperty allows an Add-In registered to provide rules for a language to control visibility for individual properties in the docked Properties window.

This is called during selection of elements when the Properties window is visible.

Syntax

Function EARules_ShowElementProperty (Repository As EA.Repository, Language As String, Element as EA.Element, Namespace as String, Class as String, Property as String) As Integer

The EARules_ShowElementProperty function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize
Element	EA.Element Direction: IN Description: Specifies the element currently being drawn.
Namespace	String Direction: IN Description: Specifies the top level language this property comes from. Will either be "UML" or the name of a profile.
Class	String Direction: IN Description: Specifies the type this property was defined in. In the UML namespace that means the metaclass defined in UML. Otherwise it will be a stereotype.
Property	String Direction: IN Description: Specifies the metaclass or stereotype property to display or hide.

Return Value

- Return a positive value to display the property

- Return zero to hide the property
- Return a negative value to use the behavior from the base rules

EARules_ShowFrame

EARules_ShowFrame allows an Add-In registered to provide rules for a language to determine if the owner of a diagram should always be displayed on the diagram as a diagram frame.

This event occurs during diagram load.

Syntax

Function EARules_ShowFrame (Repository As EA.Repository, Language As String, Diagram as EA.Diagram) As Integer

The EARules_ShowFrame function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Diagram	EA.Diagram Direction: IN Description: Specifies the diagram currently being loaded.

Return Value

- Return a positive value to specify that the parent element should be shown as a frame on this diagram
- Return zero to specify the frame should not be displayed on the diagram
- Return a negative value to use the behavior from the base rules

EARules_ShowParentFrame

EARules_ShowParentFrame allows an Add-In registered to provide rules for a language to determine if the owner of a diagram should always be displayed on the diagram as a diagram frame.

This event occurs during diagram load.

Syntax

Function EARules_ShowParentFrame (Repository As EA.Repository, Language As String, Diagram as EA.Diagram) As Integer

The EARules_ShowParentFrame function syntax contains these parameters.

Parameter	Description
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Language	String Direction: IN Description: Specifies the language for the rule that Enterprise Architect is requesting. This will match one of the values returned from EARules_Initialize.
Diagram	EA.Diagram Direction: IN Description: Specifies the diagram currently being loaded.

Return Value

- Return a positive value to specify that the parent element should be shown as a frame on this diagram
- Return zero to specify the name should not be displayed on the diagram
- Return a negative value to use the behavior from the base rules

Vues personnalisées

Enterprise Architect permet d'insérer des fenêtres personnalisées sous forme d'onglet dans la Vue Diagramme qui apparaît au centre du cadre Enterprise Architect .

La création d'une vue personnalisée vous aide à afficher facilement une interface personnalisée dans Enterprise Architect , ainsi que d'autres diagrammes et vues intégrées pour un accès rapide et facile.

Les utilisations possibles de ce facilité incluent :

- Rapports et graphiques présentant les données récapitulatives du modèle
- Vues alternatives d'un diagramme
- Vues alternatives du modèle
- Vues de données externes liées aux données du modèle
- Outils de documentation

Gardez à l'esprit que l'option « Ouvrir Diagrammes dans une seule fenêtre » dans la dialogue « Aspect de l'application » échangera diagrammes dans la Vue Diagramme plutôt que d'ouvrir plus d'onglets diagramme .

Créer une Vue personnalisée

Une vue personnalisée doit être conçue comme un contrôle personnalisé ActiveX et insérée via l'interface d'automatisation. Les contrôles personnalisés ActiveX peuvent être créés à l'aide de la plupart des outils de programmation les plus connus, notamment Microsoft Visual Studio. Consultez la documentation fournie par le fournisseur concerné pour savoir comment créer un contrôle personnalisé afin de produire un fichier OCX.

Une fois le contrôle personnalisé créé et enregistré sur le système cible, il peut être ajouté via la méthode AddTab() de l'objet Référentiel . Bien qu'il soit possible d'appeler AddTab() à partir de n'importe quel client d'automatisation, il est probable que vous l'appeliez à partir d'un Add-In , et que ce Add-In soit défini dans le même OCX qui fournit la vue personnalisée.

Exemple de code C#

Complément de classe publique

```
{
UserControl1 m_MyControl;
public void EA_Connect ( EA.Repository Rep)
{
}
object public EA_GetMenuItems ( EA.Repository Référentiel , string Location, string MenuName)
{
si(NomMenu == " ")
return "-& Démo de contrôle C# " ;
autre
{
String () ret = {"Afficher la Vue personnalisée", "Afficher le bouton"} ;
retour ret;
}
}
public void EA_MenuClick ( EA.Repository Rep, string Location, string MenuName, string ItemName )
{
if( ItemName == "Afficher la Vue personnalisée")
m_MyControl = (UserControl1) Rep.AddTab( " C# Demo","ContDemo.UserControl1");
else if( ItemName == "Afficher le bouton")
m_MyControl.ShowButton();
}
}
```

Fenêtre Ancrée Personnalisée

Fenêtres ancrées personnalisées peuvent être ajoutées à l'interface utilisateur Enterprise Architect . Une fois ajoutées, elles peuvent être affichées et ancrées de la même manière que les autres fenêtres ancrées intégrées Enterprise Architect .

Une fenêtre ancrée personnalisée doit être conçue comme un contrôle personnalisé ActiveX et insérée via l'interface d'automatisation. Les contrôles personnalisés ActiveX peuvent être créés à l'aide de la plupart des outils de programmation les plus connus, notamment Microsoft Visual Studio. Consultez la documentation fournie par le fournisseur concerné pour savoir comment créer un contrôle personnalisé afin de produire un fichier OCX.

Une fois le contrôle personnalisé créé et enregistré sur le système cible, il peut être ajouté à l'aide de la méthode AddWindow() de l' object Référentiel . Bien qu'il soit possible d'appeler AddWindow() à partir de n'importe quel client d'automatisation, il est probable que vous l'appeliez à partir d'un Add-In , et que ce Add-In soit défini dans le même OCX qui fournit la vue personnalisée.

Pour afficher les fenêtres ancrées personnalisées qui ont été ajoutées, sélectionnez l'option de ruban « Spécialiser > Add-Ins > Windows de complément ».

Fenêtres ancrées personnalisées peuvent également être rendues visibles par le client d'automatisation ou Add-In à l'aide de la méthode ShowAddinWindow(), ou masquées à l'aide de la méthode HideAddinWindow().

Exemple de code C#

Complément de classe publique

```
{
UserControl1 m_MyControl;
public void EA_Connect ( EA.Repository Rep)
{
m_MyControl = (UserControl1) Rep.AddWindow
(" Démo C# ","ContDemo.UserControl1");
}
object public EA_GetMenuItems ( EA.Repository Référentiel , string Location, string MenuName)
{
si(NomMenu == " ")
return "-& Démo de contrôle C# " ;
autre
{
String () ret = {"Afficher la fenêtre", "Afficher le bouton"} ;
retour ret;
}
}
public void EA_MenuClick ( EA.Repository Rep, string Location, string MenuName, string ItemName )
{
if( ItemName == "Afficher la fenêtre")
Rep.ShowAddinWindow( " Démo C# ");
else if( ItemName == "Afficher le bouton")
m_MyControl.ShowButton();
}
```

}

Add-Ins MDG

Add-Ins MDG sont des types spécialisés de Add-In qui ont fonctionnalités supplémentaires et des exigences supplémentaires, pour les auteurs Add-In qui souhaitent contribuer à l'objectif d' Enterprise Architect de génération pilotée Modèle .

L'une des responsabilités supplémentaires d'un Add-In MDG consiste à prendre possession d'une branche d'un modèle Enterprise Architect , ce qui est effectué via l'événement MDG_Connect . Contrairement aux événements Add-In généraux, les événements Add-In MDG sont envoyés uniquement au Add-In qui a pris possession d'une branche de modèle Enterprise Architect sur un poste de travail particulier.

Add-Ins MDG s'identifient comme tels lors EA_Connect en renvoyant la string « MDG ».

Contrairement Add-Ins ordinaires, la réponse aux événements du Add-In MDG n'est pas facultative et les méthodes doivent être publiées pour chacun des Événements MDG.

Événements MDG

An MDG Add-In must respond to all MDG Events. These events usually identify processes such as Build, Run, Synchronize, PreMerge and PostMerge, amongst others.

An MDG Link Add-In is expected to implement some form of forward and reverse engineering capability within Enterprise Architect, and as such requires access to a specific set of events, all to do with generation, synchronization and general processes concerned with converting models to code and code to models.

MDGAdd-In Events

Event
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

MDG_BuildProject

Add-Ins can use MDG_BuildProject to handle file changes caused by generation. This function is called in response to a user selecting the 'Execute > Source > Build > Build' ribbon option.

Respond to this event by compiling the project source files into a running application.

Syntax

Sub MDG_BuildProject (Repository As EA.Repository, PackageGuid As String)

The MDG_BuildProject function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

Return Value

None.

MDG_Connect

An Add-In uses MDG_Connect to handle a user driven request to connect a model branch to an external application. The function is called when the user attempts to connect a particular Enterprise Architect Package to an as yet unspecified external project. The Add-In calls the event to interact with the user to specify such a project.

The Add-In is responsible for retaining the connection details, which should be stored on a per-user or per-workstation basis. That is, users who share a common Enterprise Architect model over a network should be able to connect and disconnect to external projects independently of one another.

The Add-In should therefore not store connection details in an Enterprise Architect repository. A suitable place to store such details would be:

```
SHGetFolderPath(..CSIDL_APPDATA..)AddinName
```

The PackageGuid parameter is the same identifier as 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.

The PackageID parameter is provided to aid fast retrieval of Package details from Enterprise Architect, should this be required.

Syntax

Function MDG_Connect (Repository As EA.Repository, PackageID as Long, PackageGuid As String) As Long

The MDG_Connect function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageID	Long Direction: IN Description: The PackageID of the Enterprise Architect Package the user has requested to have connected to an external project.
PackageGuid	String Direction: IN Description: The unique ID identifying the project provided by the Add-In when a connection to a project branch of an Enterprise Architect model was first established.

Return Value

- Return a non-zero to indicate that a connection has been made
- Return a zero to indicate that the user has not nominated a project and connection should not proceed

MDG_Disconnect

Add-Ins can use MDG_Disconnect to respond to user requests to disconnect the model branch from an external project. This function is called when the user attempts to disconnect an associated external project. The Add-In is required to delete the details of the connection.

Syntax

Function MDG_Disconnect (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_Disconnect function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

Return Value

- Return a non-zero to indicate that a disconnection has occurred, enabling Enterprise Architect to update the user interface
- Return a zero to indicate that the user has not disconnected from an external project

MDG_GetConnectedPackages

Add-Ins can use MDG_GetConnectedPackages to return a list of current connections between Enterprise Architect and an external application.

This function is called when the Add-In is first loaded, and is expected to return a list of the available connections to external projects for this Add-In.

Syntax

Function MDG_GetConnectedPackages (Repository As EA.Repository) As Variant

The MDG_GetConnectedPackages function syntax contains this parameter.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.

Return Value

Returns an array of GUID strings representing individual Enterprise Architect Packages.

MDG_GetProperty

MDG_GetProperty provides miscellaneous Add-In details to Enterprise Architect.

This function is called by Enterprise Architect to poll the Add-In for information relating to the `PropertyName`. This event should occur in as short a duration as possible, as Enterprise Architect does not cache the information provided by the function.

Values corresponding to these `PropertyNames` must be provided:

- `IconID` - Return the name of a DLL and a resource identifier in the format `#ResID`, where the resource ID indicates an icon
`c:\program files\myapp\myapp.dll#101`
- `Language` - Return the default language that Classes should be assigned when they are created in Enterprise Architect
- `HiddenMenus` - Return one or more values from the `MDGMenus` enumeration to hide menus that do not apply to your Add-In

```
if(PropertyName == "HiddenMenus")
    return mgBuildProject + mgRun;
```

Syntax

Function MDG_GetProperty (Repository As EA.Repository, PackageGuid As String, PropertyName As String) As Variant

The MDG_GetProperty function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently-open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
PropertyName	String Direction: IN Description: The name of the property that is used by Enterprise Architect. See the start of this topic for the possible values.

Return Value

See the start of this topic.

MDG_Merge

Add-Ins can use MDG_Merge to jointly handle changes to both the model branch and the code project that the model branch is connected to.

This event should be called whenever the user has asked to merge their model branch with its connected code project, or whenever the user has established a new connection to a code project.

The purpose of this event is to make the Add-In interact with the user to perform a merge between the model branch and the connected project.

Syntax

Function MDG_Merge (Repository As EA.Repository, PackageGuid As String, SynchObjects As Variant, SynchType As String, ExportObjects As Variant, ExportFiles As Variant, ImportFiles As Variant, IgnoreLocked As String, Language As String) As Long

The MDG_Merge function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
SynchObjects	Variant Direction: OUT Description: A string array containing a list of objects (Object ID format) to be jointly synchronized between the model branch and the project. See <i>Object ID Format</i> for the format of the Object IDs.
SynchType	String Direction: OUT Description: The value determining the user-selected type of synchronization to take place. See <i>Synchronize Type</i> for a list of valid values.
ExportObjects	Variant Direction: OUT Description: The string array containing the list of new model objects (in Object ID format) to be exported by Enterprise Architect to the code project.
ExportFiles	Variant Direction: OUT Description: A string array containing the list of files for each model object chosen

	<p>for export by the Add-In.</p> <p>Each entry in this array must have a corresponding entry in the ExportObjects parameter at the same array index, so ExportFiles(2) must contain the filename of the object by ExportObjects(2).</p>
ImportFiles	<p>Variant</p> <p>Direction: OUT</p> <p>Description: A string array containing the list of code files made available to the code project to be newly imported to the model.</p> <p>Enterprise Architect imports each file listed in this array for import into the connected model branch.</p>
IgnoreLocked	<p>String</p> <p>Direction: OUT</p> <p>Description: A value indicating whether to ignore any files locked by the code project (that is, 'True' or 'False').</p>
Language	<p>String</p> <p>Direction: OUT</p> <p>Description: The string value containing the name of the code language supported by the code project connected to the model branch.</p>

Object ID Format

Each of the Object IDs listed in the 'SynchObjects' string arrays should have this format:

`(@namespace)*(#class)*($attribute|%operation|:property)*`

Return Value

- Return a non-zero if the merge operation completed successfully
- Return a zero when the operation has been unsuccessful

Merge

A merge consists of three major operations:

- Export: where newly created model objects are exported into code and made available to the code project
- Import: where newly created code objects, Classes and such things are imported into the model
- Synchronize: where objects available both to the model and in code are jointly updated to reflect changes made in either the model, code project or both

Synchronize Type

The Synchronize operation can take place in one of four different ways. Each of these ways corresponds to a value returned by 'SynchType':

- None: ('SynchType' = 0) No synchronization is to be performed
- Forward: ('SynchType' = 1) Forward synchronization, between the model branch and the code project is to occur
- Reverse: ('SynchType' = 2) Reverse synchronization, between the code project and the model branch is to occur
- Both: ('SynchType' = 3) Reverse, then Forward synchronizations are to occur

MDG_NewClass

Add-Ins can use MDG_NewClass to alter details of a Class before it is created.

This method is called when Enterprise Architect generates a new Class, and requires information relating to assigning the language and file path. The file path should be passed back as a return value and the language should be passed back via the language parameter.

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 these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
CodeID	String Direction: IN Description: A string used to identify the code element before it is created.
Language	String Direction: OUT Description: A string used to identify the programming language for the new Class. The language must be supported by Enterprise Architect.

Return Value

Return a string containing the file path that should be assigned to the Class.

MDG_PostGenerate

Add-Ins can use MDG_PostGenerate to handle file changes caused by generation.

This event is called after Enterprise Architect has prepared text to replace the existing contents of a file. Responding to this event enables the Add-In to write to the linked application's user interface rather than modify the file directly.

When the contents of a file are changed, Enterprise Architect passes FileContents as a non-empty string. New files created as a result of code generation are also sent through this mechanism, so the Add-Ins can add new files to the linked project's file list.

When new files are created Enterprise Architect passes FileContents as an empty string. When a non-zero is returned by this function, the Add-In has successfully written the contents of the file. A zero value for the return indicates to Enterprise Architect that the file must be saved.

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 these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
FilePath	String Direction: IN Description: The path of the file Enterprise Architect intends to overwrite.
FileContents	String Direction: IN Description: A string containing the proposed contents of the file.

Return Value

The return value depends on the type of event that this function is responding to (see introduction). This function is required to handle two separate and distinct cases.

MDG_PostMerge

MDG_PostMerge is called by Enterprise Architect after a merge process has been completed.

File save checking should not be performed with this function, but should be handled by MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse.

Syntax

Function MDG_PostMerge (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PostMerge function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

Return Value

- Return a non-zero to indicate that the post-merge has been successful
- Return a zero if the post-merge process has failed

Enterprise Architect assumes a non-zero return if this method is not implemented.

MDG_PreGenerate

Add-Ins can use MDG_PreGenerate to deal with unsaved changes.

This function is called immediately before Enterprise Architect attempts to generate files from the model. A possible use of this function would be to prompt the user to save unsaved source files.

Return Value

- Return a zero to abort generation
- Return any other value to enable the generation to continue

Syntax

Function MDG_PreGenerate (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreGenerate function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

MDG_PreMerge

MDG_PreMerge is called after a merge process has been initiated by the user and before Enterprise Architect performs the merge process.

This event is called after a user has performed their interactions with the merge screen and has confirmed the merge with the OK button, but before Enterprise Architect performs the merge process using the data provided by the MDG_Merge call, before any changes have been made to the model or the connected project.

This event is made available to provide the Add-In with the opportunity to generally set internal Add-In flags to augment the MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse events.

File save checking should not be performed with this function, but should be handled by MDG_PreGenerate, MDG_PostGenerate and MDG_PreReverse.

Syntax

Function MDG_PreMerge (Repository As EA.Repository, PackageGuid As String) As Long

The MDG_PreMerge function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

Return Value

- Return a zero to indicate that the merge process can not occur
- Return a non-zero if the merge process proceeds

If this method is not implemented then it is assumed that a merge process is used.

MDG_PreReverse

Add-Ins can use MDG_PreReverse to save file changes before they are imported into Enterprise Architect.

This function operates on a list of files that are about to be reverse-engineered into Enterprise Architect. If the user is working on unsaved versions of these files in an editor, you could either prompt the user or save automatically.

Syntax

Sub MDG_PreReverse (Repository As EA.Repository, PackageGuid As String, FilePaths As Variant)

The MDG_PreReverse function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
FilePaths	String array Direction: IN Description: An array of filepaths pointed to the files that are to be reverse engineered.

Return Value

None.

MDG_RunExe

Add-Ins can use MDG_RunExe to run the target application.

This function is called when the user selects the 'Execute > Run > Start > Run' ribbon option.

Respond to this event by launching the compiled application.

Syntax

Sub MDG_RunExe (Repository As EA.Repository, PackageGuid As String)

The MDG_RunExe function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.

Return Value

None.

MDG_View

Add-Ins can use MDG_View to display user specified code elements.

This function is called by Enterprise Architect when the user asks to view a particular code element. The Add-In can then present that element in its own way, usually in a code editor.

Syntax

Function MDG_View (Repository As EA.Repository, PackageGuid As String, CodeID as String) As Long

The MDG_View function syntax contains these parameters.

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
PackageGuid	String Direction: IN Description: The GUID identifying the Enterprise Architect Package sub-tree that is controlled by the Add-In.
CodeID	String Direction: IN Description: Identifies the code element in this format: <code><type>ElementPart<type>ElementPart...</code> where each element is preceded with a token identifying its type: @ - namespace # - Class \$ - attribute % - operation 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 this format: <code>@Name1#Class1%m_Name</code>

Return Value

- Return a non-zero value to indicate that the Add-In has processed the request
- Return a zero value for Enterprise Architect to employ the standard viewing process, which is to launch the associated source file

Add-In Workflow Événements

Enterprise Architect provides this set of four additional events that are sent only to workflow Add-Ins.

To use these the Workflow Add-In must be initialized with `EA_Connect` set to type: "Workflow". For more details see the `EA_Connect` Help topic.

Workflow Add-In Events

Event
<code>EA_AllowPropertyUpdate</code> This event is sent to workflow Add-Ins after a user has changed a built-in property value.
<code>EA_AllowTagUpdate</code> This event is sent to workflow Add-Ins after a user has changed a Tagged Value.
<code>EA_CanEditProperty</code> This event is sent to workflow Add-Ins when a property is being displayed that allows the property to block all edits.
<code>EA_CanEditTag</code> This event is sent to workflow Add-Ins when a Tagged Value is being displayed that allows the property to block all edits.

EA_AllowPropertyUpdate

This event is sent to workflow Add-Ins after a user has changed a built-in property value.

Syntax

Function EA_AllowPropertyUpdate (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the requested property update: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• Stereotype: A string value corresponding to Element.Stereotype• Property: The name of the property field to enable or disable• OldValue: The previous value of the property• NewValue: The new value of the property

Return Value

- Return False to prevent this change to the described property
- Return True to allow this change

EA_AllowTagUpdate

This event is sent to Workflow Add-Ins after a user has changed a Tagged Value.

Syntax

Function EA_AllowTagUpdate (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the requested Tagged Value update: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• Stereotype: A string value corresponding to Element.Stereotype• TagName: The name of the Tagged Value field to enable or disable• OldValue: The previous value of the tag• NewValue: The new value of the tag

Return Value

- Return False to prevent this change to the described Tagged Value
- Return True to allow this change

EA_CanEditProperty

This event is sent to Workflow Add-Ins when a property is being displayed that allows the property to block all edits.

Syntax

Function EA_CanEditProperty (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the property: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• Stereotype: A string value corresponding to Element.Stereotype• PropertyName: The name of the property field to enable or disable

Return Value

- Return False to prevent all edits to the described property
- Return True to allow changes

EA_CanEditTag

This event is sent to Workflow Add-Ins when a Tagged Value is being displayed that allows the property to block all edits.

Syntax

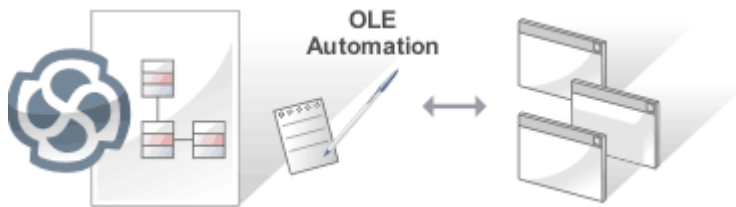
Function EA_CanEditTag (Repository As EA.Repository, Info As EA.EventProperties) As Boolean

Parameter	Type
Repository	EA.Repository Direction: IN Description: An EA.Repository object representing the currently open Enterprise Architect model. Poll its members to retrieve model data and user interface status information.
Info	EA.EventProperties Direction: IN Description: Contains these EventProperty objects describing the Tagged Value: <ul style="list-style-type: none">• Type: A string value corresponding to Element.Type• Stereotype: A string value corresponding to Element.Stereotype• TagName: The name of the tag to enable or disable

Return Value

- Return False to prevent all edits to the described Tagged Value
- Return True to allow changes

Enterprise Architect Modèle d'Objet



Le Enterprise Architect Modèle d'Objet donne au scripteur ou au programmeur accès aux objets sous-jacents que vous pouvez utiliser pour interroger ou manipuler le référentiel. Le Modèle Object est accessible soit à partir d'environnements de scripts internes ou externes, soit via Add-Ins . Il s'agit d'une fonctionnalité pratique qui garantit qu'un programmeur est isolé de la base de données sous-jacente où le référentiel est stocké, le protégeant ainsi des modifications de la structure ou du contenu de la base de données. Les objets sont regroupés dans Paquetages et contiennent un ensemble utile, complet et bien documenté de propriétés et de méthodes qui sont intuitives à utiliser et permettent d'accéder aux éléments, fonctionnalités , diagrammes et aux métadonnées du projet.

L'automatisation permet à d'autres applications d'accéder aux informations d'un modèle Enterprise Architect à l'aide de Windows OLE Automation (ActiveX). En général, cela implique l'écriture de scripts sur des clients tels que MS Word™ ou Visual Basic, ou l'utilisation de scripts créés dans Enterprise Architect à l'aide de la fenêtre Scriptant .

L'interface d'automatisation permet d'accéder aux éléments internes des modèles Enterprise Architect . Voici quelques exemples de ce que vous pouvez faire à l'aide de l'interface d'automatisation :

- Effectuer des tâches répétitives, telles que mettre à jour le numéro de version de tous les éléments d'un modèle
- Générer du code à partir d'un diagramme Statemachine
- Produire des rapports personnalisés
- Effectuer des requêtes ad hoc

Fonctionnalités

Fonctionnalité	Description
Connexion à l'interface d'automatisation	Tous les environnements de développement capables de générer des clients COM ActiveX doivent pouvoir se connecter à l'interface d'automatisation Enterprise Architect . Ce guide fournit des instructions détaillées sur la connexion à l'interface à l'aide de Microsoft Visual Basic 6.0, Borland Delphi 7.0, Microsoft C# et Java. Il contient également des étapes plus détaillées sur la configuration de Visual Basic ; les principes sont applicables à d'autres langages.
Exemples et Conseils	Des instructions sur la façon d'utiliser l'interface d'automatisation sont fournies au moyen d'exemples de code. Consultez les pointeurs vers les exemples et les autres ressources disponibles. Consultez également la section Référence détaillée.
Appel d'exécutables depuis Enterprise Architect	Enterprise Architect peut être configuré pour appeler une application externe. Vous pouvez transmettre des paramètres sur la position actuelle sélectionnée dans la fenêtre Navigateur à l'application appelée. Pour obtenir des instructions, consultez la rubrique <i>Appeler depuis Enterprise Architect</i> . Une méthode plus sophistiquée consiste à créer Add-Ins , qui sont abordés dans une section distincte.

Utilisation de l'interface d'automatisation

Cette section fournit des instructions sur la façon de se connecter et d'utiliser l'interface d'automatisation, notamment :

- Connexion à l'interface
- Définition des références dans Visual Basic
- Exemples et Conseils

Se connecter à l'interface

Tous les environnements de développement capables de générer des clients ActiveX Com peuvent se connecter à l'interface d'automatisation Enterprise Architect .

À titre d'exemple, ces sections décrivent comment se connecter à l'aide de plusieurs de ces outils. La procédure peut varier légèrement selon les différentes versions de ces produits.

Microsoft Visual Basic 6.0

Cette procédure prend en charge la syntaxe et les frameworks de la version 6.0. Les versions plus récentes ont le même framework que les autres langages .Net avec seulement des différences de syntaxe, et utilisent donc un processus similaire à celui décrit pour Microsoft C# , plus loin dans cette rubrique.

Étape	Action
1	Créer un nouveau projet.
2	Sélectionnez l'option de menu « Projet Références ».
3	Sélectionnez Enterprise Architect Modèle d'Objet 2.0 dans la liste. Si cela n'apparaît pas, accédez à la ligne de commande et réenregistrez Enterprise Architect en utilisant : EA.exe /unregister alors EA.exe /register
4	Consultez la documentation générale de la bibliothèque sur l'utilisation des classes. Cet exemple crée et ouvre un objet de référentiel : Sous-ShowRepository public() Dim MyRep en tant que nouveau EA.Repository MonRep.OpenFile "c:\eatest.eap" Fin de sous-titre

Borland Delphi 7.0

Note que les versions récentes de Delphi sont développées par Embarcadero.

Étape	Action
1	Créer un nouveau projet.
2	Sélectionnez l'option de menu « Projet Importer Bibliothèque Type ».
3	Sélectionnez Enterprise Architect Modèle d'Objet 2.0 dans la liste. Si cela n'apparaît pas, accédez à la ligne de commande et réenregistrez Enterprise Architect en utilisant : EA.exe /unregister alors

	EA.exe /register
4	Cliquez sur le bouton Créer une unité.
5	Inclure EA_TLB dans la clause Uses de Project1.
6	<p>Consultez la documentation générale de la bibliothèque sur l'utilisation des classes. Cet exemple crée et ouvre un objet de référentiel :</p> <pre> procédure TForm1.Button1Click(Sender: TObject); var r: TRépositaire; b : booléen ; commencer r:= TRepository.Create(nil); b:= r.OpenFile('c:\eatest.eap'); fin; </pre>

Microsoft C#

Étape	Action
1	Sélectionnez l'option de menu « Projet Visual Studio Ajouter une référence ».
2	Cliquez sur l'onglet « Parcourir ».
3	<p>Accédez au dossier dans lequel vous avez installé Enterprise Architect ; généralement :</p> <p>Fichiers de programme/ Sparx Systems /EA</p> <p>Sélectionner</p> <p>Interop.EA.dll</p>
4	<p>Consultez la documentation générale de la bibliothèque sur l'utilisation des classes. Cet exemple crée et ouvre un objet de référentiel :</p> <pre> bouton privé void1_Click(object expéditeur, System.EventArgs e) { EA.Repository r = nouveau EA.Repository (); r.OpenFile("c:\\eatest.eap"); } </pre>

Java

Étape	Action

1	<p>Copier le fichier: SSJavaCOM.dll</p> <p>à partir du sous-répertoire Java API de votre répertoire d'installation, généralement : Fichiers de programme/ Sparx Systems /EA dans n'importe quel emplacement dans le chemin Windows répertoire windows\system32.</p> <p>Note : l'API Java charge le dernier Enterprise Architect installé et n'est pas affectée lors de l'utilisation de la version 32 ou 64 de la DLL, à condition que la DLL SSJavaCOM puisse être trouvée par l'environnement d'exécution Java.</p>
2	<p>Copier le fichier eaapi.jar</p> <p>à partir du sous-répertoire Java API de votre répertoire d'installation, généralement : Fichiers de programme/ Sparx Systems /EA vers un emplacement dans le CLASSPATH Java ou là où le chargeur de classe Java peut le trouver au moment exécuter .</p>
3	<p>Toutes les classes décrites dans la documentation se trouvent dans le Paquetage org.sparx. Consultez la documentation générale de la bibliothèque pour connaître leur utilisation. Cet exemple crée et ouvre un objet de référentiel :</p> <pre>public void OpenRepository() { org.sparx.Repository r = nouveau org.sparx.Repository (); r.OpenFile("c:\\eatest.eap"); }</pre>

Définir des références dans Visual Basic

Il est possible d'utiliser l'interface ActiveX Enterprise Architect avec Visual Basic (VB). Son utilisation est garantie pour la version 6 de Visual Basic, mais peut varier légèrement avec des versions autres que la version 6.

Il est supposé que vous avez accédé à VB via une application Microsoft telle que VB 6.0, MS Word TM ou MS Access. Si le code n'est pas appelé depuis Word, la référence Word VB doit également être définie.

Lors de la création d'un nouveau projet VB, vous définissez une référence à une Bibliothèque Type Enterprise Architect et à une Bibliothèque Type Word.

Définir des références

Étape	Action
1	Sélectionnez l'option de menu « Outils Références ».
2	Sélectionnez la case à cocher « Enterprise Architect Modèle d'Objet 2.10 » dans la liste.
3	Faites de même pour VB ou VB Word : cochez la case « Bibliothèque Object Microsoft Word 10.0 ».
4	Cliquez sur le bouton OK .

Notes

- Si « Enterprise Architect Modèle d'Objet 2.10 » n'apparaît pas dans la liste, accédez à la ligne de commande et ressaisissez manuellement Enterprise Architect en utilisant :
 - (Pour désinscrire Enterprise Architect) ea.exe /unregister
 - (Pour enregistrer Enterprise Architect) ea.exe /register
- Les utilisateurs de Visual Basic 5/6 doivent également noter que le numéro de version de l'interface Enterprise Architect est stocké dans le fichier de projet VBP sous une forme similaire à celle-ci :
 Référence=*G{64FB2BF4-9EFA-11D2-8307-C45586000000}#2.2#0#..\..\..\Fichiers de programme\
 Sparx Systems \EA\EA.TLB# Modèle Object Enterprise Architect 2.02
 Si vous rencontrez des problèmes lors du passage d'une version d' Enterprise Architect à une autre, ouvrez le fichier VBP dans un éditeur de texte et supprimez cette ligne, puis ouvrez le projet dans Visual Basic et utilisez Project-References pour créer une nouvelle référence au modèle Object Enterprise Architect
 La référence aux objets dans Enterprise Architect et Word devrait maintenant être disponible dans le Navigateur Object , accessible depuis le menu principal en appuyant sur F2
 La liste déroulante en haut à gauche de la fenêtre devrait maintenant inclure Enterprise Architect et Word ; si MS-Project est installé, configurez-le également

Exemples et Conseils

Points à considérer

Sujet	Points
Exemples	<p>Les instructions d'utilisation de l'interface sont fournies sous forme d'exemples de code. Il existe plusieurs séries d'exemples :</p> <ul style="list-style-type: none"> • Des exemples VB 6 et C# sont disponibles dans le dossier Exemples de code sous votre installation Enterprise Architect (par défaut : C:\Program Files\ Sparx Systems \EA\Code Samples) • Enterprise Architect peut être configuré pour appeler une application externe • Plusieurs extraits de code VB.NET sont fournis dans la section de référence • Un exemple complet d'utilisation de Visual Basic pour créer une documentation MS Word TM est disponible sur Internet à l' adresse sparxsystems.com/resources/developers/autint_vb.html • Des échantillons supplémentaires sont disponibles sur le site Web Sparx Systems ; voir la rubrique <i>Ressources disponibles</i>
Conseils et astuces	<p>note également ces trucs et astuces :</p> <ul style="list-style-type: none"> • Une instance du processus Enterprise Architect (EA.exe) est exécutée lorsque vous initialisez un nouvel objet de référentiel - ce processus doit rester en cours d'exécution pour effectuer des tâches d'automatisation ; si la fenêtre principale est visible, vous pouvez la réduire en toute sécurité, mais elle doit rester en cours d'exécution • L'interface ActiveX Enterprise Architect est une interface fonctionnelle plutôt qu'une interface de données ; lorsque vous chargez des données via l'interface, il y a un délai notable pendant lequel les éléments de l'interface utilisateur Enterprise Architect (tels que Windows et les menus) sont chargés et la connexion à la base de données spécifiée est établie • Les collections utilisent un index de base zéro ; par exemple, Référentiel .Models(0) représente le premier modèle du référentiel • Pendant le développement de votre logiciel client, votre programme peut se terminer de manière inattendue et laisser EA.exe s'exécuter dans un état tel qu'il ne peut plus support d'autres appels d'interface ; si votre programme se termine de manière anormale, assurez-vous qu'Enterprise Architect ne reste pas exécuté en arrière-plan (voir l'onglet « Gestionnaire des tâches / Processus » Windows) • Un handle vers une instance en cours d'exécution d' Enterprise Architect peut être obtenu via l'utilisation d'un appel GetObject() (voir la page de référence pour l' object App) ; l'accès à votre modèle Enterprise Architect via l' object App permet d'interroger l'état actuel Interface Utilisateur , par exemple en utilisant GetContextItem() sur l' object Référentiel pour détecter la sélection actuelle par l'utilisateur, permettant un prototypage et des tests rapides
Enterprise Architect ne ferme pas	<p>Une fois tous les traitements effectués par un contrôleur d'automatisation terminés, il est recommandé d'appeler CloseFile () et Exit() sur l' object Référentiel , puis de définir toutes les références à l' object référentiel sur null.</p> <pre>référentiel.CloseFile (); référentiel.Exit(); repository = null;</pre>

	<p>Si votre contrôleur d'automatisation a été écrit à l'aide du framework .NET , Enterprise Architect ne se ferme pas même après avoir libéré toutes vos références. Pour forcer la libération des pointeurs COM, appelez les fonctions de gestion de la mémoire :</p> <pre>GC.Collect(); GC.WaitForPendingFinalizers();</pre> <p>Des problèmes supplémentaires surviennent lors du contrôle d'une instance en cours d'exécution d' Enterprise Architect qui charge Add-Ins . Consultez la rubrique <i>Astuces et pièges</i> pour plus de détails.</p>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Appel de Enterprise Architect

Enterprise Architect peut être configuré pour appeler une application externe. Vous pouvez transmettre des paramètres sur la position actuelle sélectionnée dans la fenêtre Navigateur à l'application appelée. Cela vous permet de :

- Ajouter une ligne de commande pour une application
- Définir les paramètres à transmettre à cette application

Les paramètres requis pour exécuter l'exécutable AutInt sont :

- Le paramètre de fichier Enterprise Architect \$f et
- Le PackageID actuel \$p

Par conséquent, les arguments devraient simplement contenir : \$f,\$p.

Une fois cette configuration effectuée, l'application peut être appelée à partir du ruban « Étendre » dans Enterprise Architect à l'aide de l'option « Étendre > <VotreApplication> ».

Accéder

Ruban	Démarrer > Apparence > Préférences > Autres options > Outils
-------	--------------------------------------------------------------

Paramètres pour transmettre des informations aux applications externes

Paramètre	Description
\$d	ID Diagramme Notes : ID permettant d'accéder diagramme associé.
\$D	Diagramme GUID Notes : GUID pour accéder au diagramme associé.
\$e	Liste séparée par des virgules des identifiants d'éléments Notes : Tous les éléments sélectionnés dans le diagramme actuel.
\$E	Liste séparée par des virgules des GUID d'éléments Notes : Tous les éléments sélectionnés dans le diagramme actuel.
\$f	Nom du projet Notes : Par exemple : C:\projects\EAexample.eap.
\$F	Appel d'application (Enterprise Architect) Notes : « Enterprise Architect ».
\$p	ID Paquetage actuel Notes : Par exemple : 144.

SP	GUID Paquetage Notes : GUID pour accéder à ce Paquetage .
----	--------------------------------------------------------------

Ressources disponibles

Ressources

Les ressources disponibles incluent :

Ressource	Lien de téléchargement
Add-In VB 6 pour générer la documentation MS Word.	sparxsystems.com/resources/developers/autint_vb.html
Add-In VB 6 pour afficher un contrôle graphique ActiveX personnalisé dans la fenêtre Enterprise Architect en tant que nouvelle vue.	sparxsystems.com/resources/developers/autint_vb_custom_view.html
Un framework Add-In de base écrit en C# . Utile comme point de départ pour la création de votre propre Add-In Enterprise Architect personnalisé.	sparxsystems.com/bin/CS_AddinFramework.zip
Une extension de l'exemple CS_AddinFramework montrant comment exporter Valeur Étiquetés vers un fichier .csv.	sparxsystems.com/bin/CS_AddinTaggedCSV.zip
Un squelette de Add-In de base écrit en Delphi.	sparxsystems.com/bin/DelphiDemo.zip
Un exemple simple Add-In écrit en C# .	sparxsystems.com/bin/CS_Sample.zip

Référence

This section provides detailed information on all the objects available in the object model provided by the Automation Interface, including:

Object Groups

Group
App Object
Enumerations
Repository Package
Element Package
Element Features Package
Connector Package
Diagram Package
Project Interface Package
Document Generator Interface Package
Mail Interface Package
Code Samples

Interface Overview

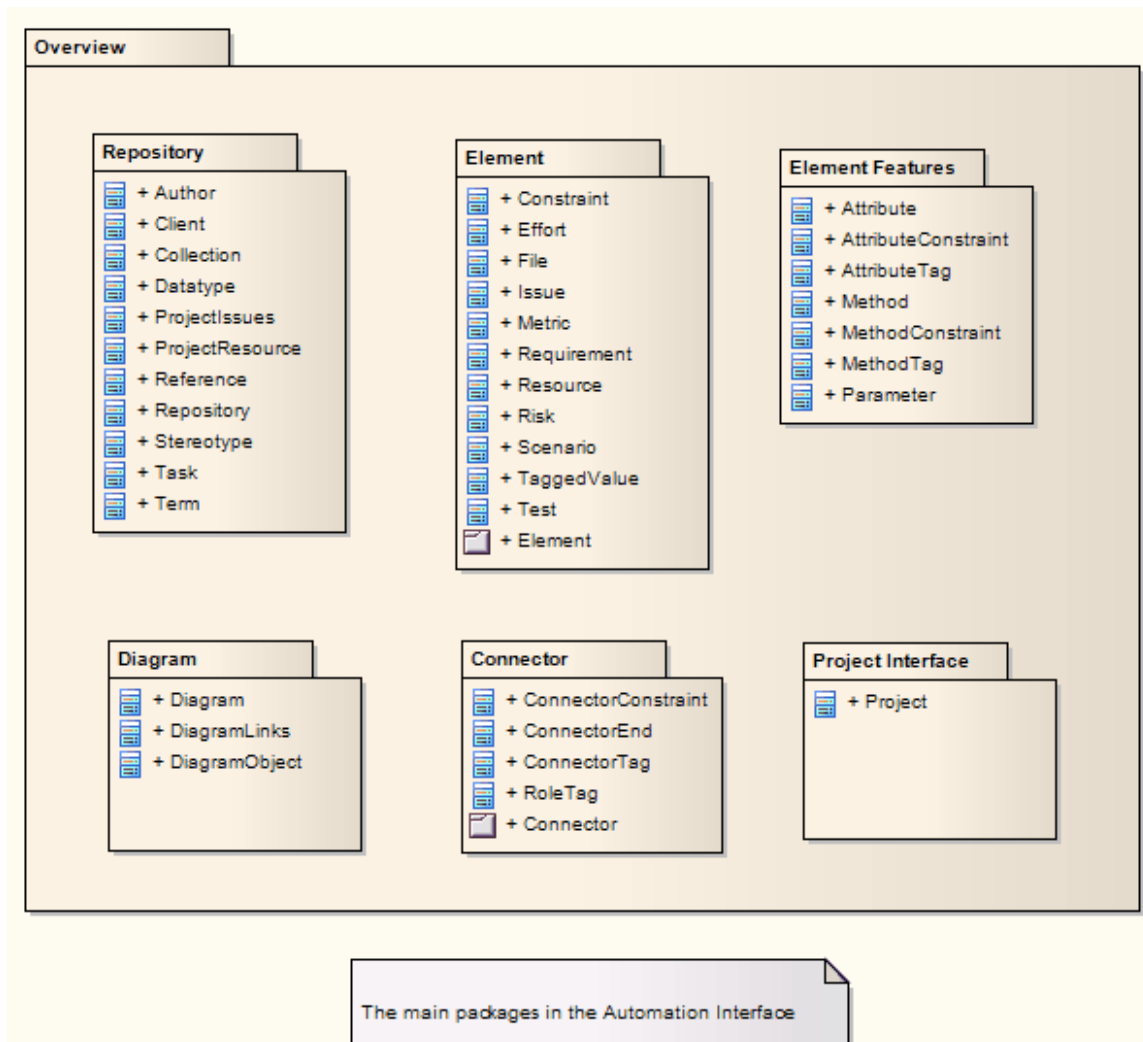
This section provides an overview of the main components of the Automation Interface.

Main Packages of Automation Interface

Package	Detail
Repository Package	Represents the model as a whole and provides entry to model Packages and collections.
Element Package	Identifies the basic structural units (such as Class, Use Case and Object).
Element Features Package	Identifies the attributes and operations defined on an element.
Diagram Package	Describes the visible drawings contained in the model.
Connector Package	Defines the relationships between elements.

Packages and Contents

This diagram illustrates the main interface Packages and their associated contents. Each UML element in this User Guide can be created by Automation and can be accessed either through the various collections that exist or, in some cases, directly.



The Repository Class is the starting point for all use of the Automation Interface. It contains the high level system objects and entry point into the model itself using the Models collection and the other system-level collections.

Object d'application

The App object represents a running instance of Enterprise Architect. Its object provides access to the Automation Interface.

Attributes

Attribute	Type
Project	Project Notes: Read only Provides a handle to the Project Interface.
Repository	Repository Notes: Read only Provides a handle to the Repository object.
Visible	Boolean Notes: Read/Write Whether or not the application is visible.

GetObject() Support

The App object is createable and a handle can be obtained by creating one. In addition, clients can use the equivalent of Visual Basic's GetObject() to obtain a reference to a currently running instance of Enterprise Architect.

Use this method to more quickly test changes to Add-Ins and external clients, as the Enterprise Architect application and data files do not have to be constantly re-loaded.

For example:

```
Dim App as EA.App
Set App = GetObject("EA.App")
MsgBox App.Repository.Models.Count
```

Another example, which uses the App object without saving it to a variable:

```
Dim Rep as EA.Repository
Set Rep = GetObject("EA.App").Repository
MsgBox Rep.ConnectionString
```

Enumerations

These enumerations are defined by the Automation Interface:

Automation Interface Enumerations

Enumeration	Link
Constant Layout Styles	Constant Layout Styles
Create Baseline Flag	Create Baseline Flag
Create Model Type	Create Model Type
Document Break	Document Break
Document Page Orientation	Document Page Orientation
Document Type	Document Type
Enterprise Architect Edition Types	Enterprise Architect Edition Types
Enumeration Relation Set Type	Enumeration Relation Set Type
Export Package XMI Flag	Export Package XMI Flag
Mail Interface Message Flag	Mail Interface Message Flag
MDG Menus	MDG Menus
Object Type	Object Type
PropType	PropType
Reload Type	Reload Type
Scenario Diagram Type	Scenario Diagram Type
Scenario Step Type	Scenario Step Type
Scenario Test Type	Scenario Test Type
XMI Type	XMI Type

ConstLayoutStyles

The enum values defined here are used exclusively for the 'Lay Out a Diagram' method. You use these values to define the layout options as provided by the 'Layout > Tools > Diagram Layout ' ribbon option.

Enum Values

Value	Meaning
IsCrossReduceAggressive	Perform aggressive Cross-reduction in the layout process (time consuming).
IsCycleRemoveDFS	Use the Depth First Cycle Removal algorithm.
IsCycleRemoveGreedy	Use the Greedy Cycle Removal algorithm.
IsDiagramDefault	Use existing layout options specified for this diagram.
IsInitializeDFSIn	Initialize the layout using the Depth First Search Inward algorithm.
IsInitializeNaive	Initialize the layout using the Naïve Initialize Indices algorithm.
IsInitializeDFSOut	Initialize the layout using the Depth First Search Outward 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.
IsLayoutDirectionDown	Direct connectors to point down.
IsLayoutDirectionLeft	Direct connectors to point left.
IsLayoutDirectionRight	Direct connectors to point right.
IsLayoutDirectionUp	Direct connectors to point up.
IsProgramDefault	Use factory default layout options as specified by Enterprise Architect.

CreateBaselineFlag

The CreateBaselineFlag enumeration is used in Baseline Management, when creating a Baseline.

Enum Values

Value	Meaning
cbSaveToStub	Baseline this Package with only immediate children (child Packages are included as stubs only).

CreateModelType

The CreateModelType enumeration is used in the CreateModel method on the Repository Class.

Enum Values

Value	Meaning
cmEAPFromBase	Create a copy of the EABase model file to the specified file path.
cmEAPFromSQLRepositor y	Create a .eap file shortcut to an SQL-based repository; requires user interaction to provide SQL connection details.

DocumentBreak

The DocumentBreak enumeration is used in the InsertBreak method on the DocumentGenerator Class.

Enum Values

Value	Meaning
breakPage	Insert a page break in the document.
breakSection	Insert a section break in the document.

DocumentPageOrientation

The DocumentPageOrientation enumeration is used in the SetPageOrientation method on the DocumentGenerator Class.

Enum Values

Value	Meaning
pagePortrait	Sets the current page orientation to Portrait.
pageLandscape	Sets the current page orientation to Landscape.

DocumentType

The DocumentType enumeration is used in the SaveDocument method on the DocumentGenerator Class.

Enum Values

Value	Meaning
dtRTF	Save the document file to disk as an RTF document.
dtHTML	Save the document file to disk as a HTML document.
dtPDF	Save the document file to disk as a PDF document.
dtDOCX	Save the document file to disk as a DOCX document.

EAEditionTypes

The `EAEditionTypes` enumeration identifies the current level of licensed functionality available.

```
EAEditionTypes theEdition = theRepository.GetEAEdition();  
if (theEdition == EAEditionTypes.piProfessional)  
...  
else if (theEdition == EAEditionTypes.piCorporate)  
...  
...
```

The enumeration defines these formal values:

- `piLite`
- `piProfessional`
- `piCorporate`
- `piBusiness`
- `piSystemEng`
- `piUltimate`

There is no separate value for the Trial Edition; the `Repository.GetEAEdition()` function returns the appropriate `EAEditionTypes` value for whichever edition the user has selected to trial.

EnumRelationSetType

This enumeration represents values returned from the GetRelationSet method of the Element object.

Enum Values

Value	Meaning
rsDependEnd	List of elements that depend on the current element.
rsDependStart	List of elements that the current element depends on.
rsGeneralizeEnd	List of elements that are generalized by the current element.
rsGeneralizeStart	List of elements that the current element generalizes.
rsParents	List of all parent elements of the current element.
rsRealizeEnd	List of elements that are realized by the current element.
rsRealizeStart	List of elements that the current element realizes.

ExportPackageXMIFlag

The ExportPackageXMIFlag enumeration is used in Package control, when exporting to XML.

Enum Values

Value	Meaning
epExcludeEAExtensions	Export this Package without any tool specific information.
epSaveToStub	Export this Package with only immediate children (child Packages are included as stubs only).

MDGMenus

Use this enumeration when providing the 'HiddenMenus' property to MDG_GetProperty.

These options are exclusive of one another and can be read or added to hide more than one menu.

Enum Values

Value	Meaning
mgBuildProject	'Hide Build Project' menu option.
mgMerge	'Hide Merge' menu option.
mgRun	'Hide Run' menu option.

MessageFlag

The MessageFlag enumeration is used in both the SendMailMessage and ComposeMailMessage methods of the MailInterface, to specify a flag to attach to the message.

Enum Values

Value	Meaning
mfNone	Do not flag the message.
mfComplete	Flag the message as 'Complete'.
mfPurple	Flag the message with a 'Purple' flag.
mfOrange	Flag the message with an 'Orange' flag.
mfGreen	Flag the message with a 'Green' flag.
mfYellow	Flag the message with a 'Yellow' flag.
mfBlue	Flag the message with a 'Blue' flag.
mfRed	Flag the message with a 'Red' flag.

ObjectType

The ObjectType enumeration identifies Enterprise Architect object types even when referenced through a Dispatch interface. For example:

```
var treeSelectedType = Repository.GetTreeSelectedItemType();
switch (treeSelectedType)
{
    case otElement :
    {
        // Code for when an element is selected
        var theElement as EA.Element;
        theElement = Repository.GetTreeSelectedObject();
        break;
    }
    case otPackage :
    {
        // Code for when a Package is selected
        var thePackage as EA.Package;
        thePackage = Repository.GetTreeSelectedObject();
        break;
    }
}
```

Valid Enumeration Values

otAttribute
otAttributeConstraint
otAttributeTag
otAuthor
otClient
otCollection
otConnector
otConnectorConstraint
otConnectorEnd
otConnectorTag
otConstraint
otCustomProperty
otDatatype
otDiagram
otDiagramLink
otDiagramObject
otEffort

otElement
otEventProperties
otEventProperty
otFile
otIssue
otMailInterface
otMethod
otMethodConstraint
otMethodTag
otMetric
otModel
otNone
otPackage
otParameter
otParamTag
otPartition
otProject
otProjectIssues
otProjectResource
otProperties
otProperty
otPropertyType
otReference
otRepository
otRequirement
otResource
otRisk
otRoleTag
otScenario
otScenarioExtension
otScenarioStep
otStereotype
otSwimlane
otSwimlaneDef
otSwimlanes
otTaggedValue
otTask
otTerm
otTest
otTransition

PropType

The PropType enumeration gives the automation programmer an indication of what sort of data is going to be stored by this property.

Enum Values

Value	Meaning
ptArray	An array containing values of any type.
ptBoolean	True or False.
ptEnum	A string being an entry in the semi-colon separated list specified in the validation field of the Property.
ptFloatingPoint	4 or 8 byte floating point value.
ptInteger	16 bit or 32 bit signed integer.
ptString	Unicode string.

ReloadType

The ReloadType enumeration represents values returned from the GetReloadItem and PeekReloadItem methods of the ModelWatcher Class. It has four possible values, which define the type of change that was made to a model.

Enum Values

Value	Meaning
rtElement	The Item parameter represents a particular element that must be reloaded.
rtEntireModel	Entire model must be reloaded to ensure that all changes are reloaded.
rtNone	No change in the model.
rtPackage	The Item parameter represents a particular Package that must be reloaded.

ScenarioDiagramType

The ScenarioDiagramType enumeration provides these enumeration values to the Project.GenerateDiagramFromScenario() method. They specify the type of diagram to generate.

Enum Values

Value	Meaning
sdActivity	Generate an Activity diagram.
sdActivityWithAction	Generate an Activity diagram with an Action.
sdActivityWithActionPin	Generate an Activity diagram with an ActionPin.
sdActivityWithActivityParameter	Generate an Activity diagram with an ActivityParameter.
sdRobustness	Generate a Robustness diagram.
sdRuleFlow	Generate a RuleFlow diagram.
sdSequence	Generate a Sequence diagram.
sdState	Generate a StateMachine diagram.

ScenarioStepType

The ScenarioStepType enumeration is used to identify the steps of a scenario, and the entity performing the step.

Enum Values

Value	Meaning
stActor	Identify that the step is an action performed by an actor.
stSystem	Identify that the step is an action performed by the system.

ScenarioTestType

The ScenarioTestType enumeration provides these enumeration values to the Project.GenerateTestFromScenario() method, to specify the type of test to generate.

Enum Values

Value	Meaning
stHorizontalTestSuite	Generate a horizontal Test Suite diagram.
stVerticalTestSuite	Generate a vertical Test Suite diagram.
stExternal	Generate an external Test Case element.
stInternal	Generate an internal test.

XMIType

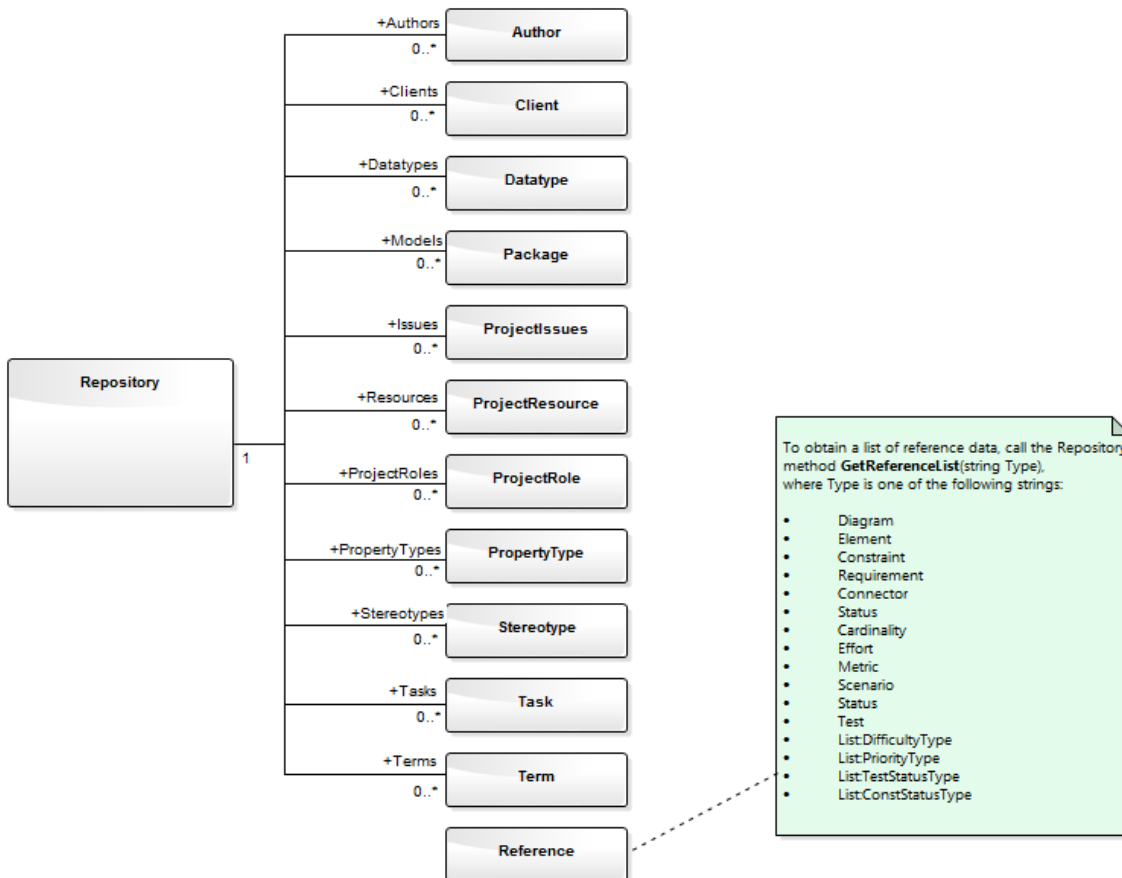
These enumeration values are used in the `Project.ExportPackageXMI()` and `Project.ExportPackageXMIEx()` methods, to specify the XMI export type.

- `xmiEADefault = 0`
- `xmiRoseDefault = 1`
- `xmiEA10 = 2`
- `xmiEA11 = 3`
- `xmiEA12 = 4`
- `xmiRose10 = 5`
- `xmiRose11 = 6`
- `xmiRose12 = 7`
- `xmiMOF13 = 8`
- `xmiMOF14 = 9`
- `xmiEA20 = 10`
- `xmiEA21 = 11`
- `xmiEA211 = 12`
- `xmiEA212 = 13`
- `xmiEA22 = 14`
- `xmiEA23 = 15`
- `xmiEA24 = 16`
- `xmiEA241 = 17`
- `xmiEA242 = 18`
- `xmiEcore = 19`
- `xmiBPMN20 = 20`
- `xmiXPDL22 = 21`
- `xmiEA251 = 22`
- `xmiARCGIS = 23`
- `xmiNative = 24`
- `xmiEA2511 = 25`
- `xmiNativeXEA = 26`

Repository Package

The Repository Package contains the high level system objects and the entry point into the model itself, using the Models collection and the other system level collections.

This diagram shows the collections of the Repository interface. Association Target roles correspond to member variable names in the Repository interface. The associated Classes represent the object type used in each collection.



Author Class

An Author object represents a named model author. Authors can be accessed using the Repository Authors collection.

Associated table in repository

t_authors

Author Attributes

Attribute	Remarks
Name	String Notes: Read/Write The Author name.
Notes	String Notes: Read/Write Notes about the author.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Roles	String Notes: Read/Write Roles the author might play in this project.

Author Methods

Method	Remarks
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update ()	Boolean Notes: Updates the current Author object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Client Class

A Client represents one or more people or organizations related to the project. Clients can be accessed using the Repository Clients collection.

Associated table in repository

t_clients

Client Attributes

Attribute	Remarks
EMail	String Notes: Read/Write The client's email address.
Fax	String Notes: Read/Write The client's fax number.
Mobile	String Notes: Read/Write The client's mobile phone number, if available.
Name	String Notes: Read/Write The client's name.
Notes	String Notes: Read/Write Notes about the client.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through the Dispatch interface.
Organization	String Notes: Read/Write The client's associated organization.
Phone1	String Notes: Read/Write The client's main phone number.

Phone2	String Notes: Read/Write The client's second phone number.
Roles	String Notes: Read/Write Roles this client might play in the project.

Client Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Client object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Collection Class

Collection 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 the 'AddNew' function is called, the item is not automatically added to the current collection. The typical steps are:

- Call AddNew to add a new item
- Modify the item as required
- Call Update on the item to save it to the database
- Call Refresh on the collection to include it in the current set

Delete is the same; until Refresh is called, the collection still contains a reference to the deleted item, which should not be called.

Each method can be used to iterate through the collection for languages that support this type of construct.

Collection Attributes

Attribute	Remarks
Count	Short Notes: Read only The number of objects referenced by this list.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Collection Methods

Method	Remarks
AddNew(string Name, string Type)	Object Notes: Adds a new item to the current collection. The interface is the same for all collections; you must provide a Name and Type argument. What these arguments are used for depends on the actual collection being accessed. For example, when adding a new element to the Elements collection, the Type string can be either a basic UML element type or a fully qualified element type (stereotype) defined by a profile, such as SysML::Requirement, differentiating it from a standard requirement. Also note that you must call Update() on the returned object to complete the AddNew function. If Update() is not called the object is left in an indeterminate state. When an error occurs an exception will be thrown, including when the user does not have Security permission to modify the specify type. Parameters:

	<ul style="list-style-type: none"> • Name: String • Type: String (up to 30 characters long)
Delete(short index)	<p>Void</p> <p>Notes: Deletes the item at the selected reference.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • index: Short
DeleteAt(short index, boolean Refresh)	<p>Void</p> <p>Notes: Deletes the item at the selected index. The second parameter is currently unused.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • index: Short • Refresh: Boolean
GetAt(short index)	<p>Object</p> <p>Notes: Retrieves the array object using a numerical index. If the index is out of bounds, an error occurs.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • index: Short
GetByName(string Name)	<p>Object</p> <p>Notes: Gets an item in the current collection by name. Supported for Model, Package, Element, Diagram and element TaggedValue collections.</p> <p>If the collection does not contain any items (or, for the Tagged Value collection, if the collection contains items but the method cannot locate an object with the specified name) the method returns a null value. For other collections, if the method is unable to find an object with the specified name, it raises an exception.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Refresh()	<p>Void</p> <p>Notes: Refreshes the collection by re-querying the model and reloading the collection. Should be called after adding a new item or after deleting an item.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current Collection object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

The AddNew Function

The AddNew() function is used widely across the API to add new objects to a Collection. In all cases you must provide a Name and Type argument, but what these arguments are used for depends on the actual collection being accessed. For example, when adding a new element to the Elements collection, the 'Type' string can be either a basic UML element type or a fully qualified element type (stereotype) defined by a profile, such as SysML::Requirement differentiated from a standard requirement.

AddNew Attribute Arguments

This table provides guidance in specifying the AddNew arguments for each of the object attributes.

Attribute	Arguments
AttributeConstraints	Name - The name of the constraint. Type - The constraint type
Attributes	Name - The name of the attribute. Type - The attribute type.
AttributesEx	Name - The name of the attribute. Type - The attribute type.
AttributeTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Authors	Name - The author name. Type - The author role.
Clients	Name - The client name. Type - The client role.
ConnectorConstraints	Name - The name of the constraint. Type - The constraint type.
ConnectorConveyedItems	Name - The GUID of an element. Type - <i>Not used</i> . Note: This does not return an object.
Connectors	Name - The name of the connector. Type - The connector type (for example 'Realization').
ConnectorTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Constraints	Name - The name of the constraint. Type - The constraint type.
ConstraintsEx	Name - The name of the constraint.

	Type - The constraint type.
CustomProperties	You cannot create these.
DataTypes	Name - The datatype name. Type - The datatype type.
DiagramLinks	Name - <i>Not used</i> . Type - The style string (such as 'l=200;r=400;t=200;b=600;') (You might prefer to leave the Type empty and use the Functions on this interface for size, colors and so on).
DiagramObjects	Name - This can either be an empty string, or it can specify the initial Left, Right, Top and Bottom values for the new DiagramObject. For example: <code>diagram.DiagramObjects.AddNew("l=200;r=400;t=200;b=600;", "")</code> Note: Top and Bottom values should be specified here as positive numbers, but will be set in the repository as negative values. Type - Unused.
Diagrams	Name - The name of the diagram. Type - This can be either a standard UML metaclass type (such as 'Class' or 'UseCase') or a fully-qualified metatype defined by an MDG Technology (such as 'BPMN2.0::BusinessProcess' or 'SysML1.4::Block').
Efforts	Name - The name of the effort. Type - The effort type.
Elements	Name - The name of the new element. If the repository has an auto-name counter defined for the element type being created, pass an empty string to use the auto-name counter instead. Type - Can be either a standard UML metaclass type (such as 'Class' or 'UseCase') or a fully-qualified metatype defined by an MDG Technology (such as 'BPMN2.0::BusinessProcess' or 'SysML1.4::Block').
Files	Name - The full pathname of the file. Type - The file type (such as 'Local File' or 'Web Address').
Issues	Name - The name of the issue. Type - The problem type, (such as 'Issue' or 'Defect')
MethodPostConditions	Name - The name of the constraint. Type - The constraint type
MethodPreconditions	Name - The name of the constraint. Type - The constraint type.
Methods	Name - The name of the method. Type - The return value of the method.
MethodsEx	Name - The name of the method.

	Type - The return value of the method.
MethodTags	Name - The fully-qualified name, or plain text. Type - The value of the Tagged Value.
Metrics	Name - The name of the metric. Type - The metric type.
Models	Name - The name of the model. Type - Unused.
Packages	Name - The name of the Package. Type - Unused.
Parameters	Name - The parameter name. Type - The parameter type.
ParamTags	Name - The fully-qualified name or plain text. Type - The value of the Tagged Value.
Partitions	Name - The partition name. Type - The partition note.
ProjectIssues	Name - The name of the issue. Type - The issue type (such as 'Request', 'Defect', or 'Release')
ProjectResources	Name - The resource name. Type - The resource role.
ProjectRole	Name - The role name. Type - <i>Not used</i> .
PropertyTypes	Name - The tag name. Type - The description (limited to 50 characters).
Requirements	Name - The name of the requirement. Type - The requirement type.
RequirementsEx	Name - The name of the requirement. Type - The requirement type.
Resources	Name - The resource name. Type - The resource role.
Risks	Name - The name of the risk. Type - The risk type.
ScenarioExtension	Name - The extension name. Type - The scenario type

ScenarioStep	Name - The step name. Type - The ScenarioStep type value.
Scenarios	Name - The name of the scenario. Type - The scenario type.
Stereotypes	Name - The stereotype name. Type - The element this applies to. Note: You can only support multiple elements from within a Profile.
Tasks	Name - The task name. Type - The task type.
TemplateBindings	Name - The formal name of the binding. Type - The actual name of the binding or element GUID.
TemplateParameters	Name - The parameter name. Type - The parameter type
Terms	Name - The term name. Type - The term type.
Tests	Name - The name of the test. Type - The test type.
Transitions	Name - The transition name. Type - The transition value.

Datatype Class

A Datatype is a named type that can be associated with attribute or method types. It typically is related to either code engineering or database modeling. Datatypes also indicate which language or database system they relate to. Datatypes can be accessed using the Repository Datatypes collection.

Associated table in repository

t_datatypes

Datatype Attributes

Attribute	Remarks
DatatypeID	Long Notes: Read/Write The instance ID for this datatype within the current model; this is system maintained.
DefaultLen	Long Notes: Read/Write The default length (DDL only).
DefaultPrec	Long Notes: Read/Write The default precision (DDL only).
DefaultScale	Long Notes: Read/Write The default scale (DDL only).
GenericType	String Notes: Read/Write The associated generic type for this data type.
HasLength	String Notes: Read/Write Indicates whether the datatype has a length component.
MaxLen	Long Notes: Read/Write The maximum length (DDL only).
MaxPrec	Long Notes: Read/Write

	The maximum precision (DDL only).
MaxScale	Long Notes: Read/Write The maximum scale (DDL only).
Name	String Notes: Read/Write The datatype name (such as integer). This appears in the related drop-down datatype lists where appropriate.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Product	String Notes: Read/Write The datatype product, such as Java, C++ or Oracle.
Size	Long Notes: Read/Write The datatype size.
Type	String Notes: Read/Write The type can be DDL for database datatypes or Code for language datatypes.
UserDefined	Long Notes: Read/Write Indicates if the datatype is a user defined type or system generated. Datatypes distributed with Enterprise Architect are all system generated. Datatypes created in the 'Datatype' dialog are marked 1 (True).

Datatype Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Datatype object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

EventProperties Class

An EventProperties object is passed to BroadcastFunctions to facilitate parameter passing.

EventProperties Attributes

Attribute	Remarks
Count	Long Notes: Read only The number of parameters being passed to this broadcast event.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EventProperties Methods

Method	Remarks
Get(object Index)	EventProperty Class Notes: Read only Returns an EventProperty in the list, raising an error if Index is out of range. Parameters: <ul style="list-style-type: none">• Index: Variant - can either be a number representing a zero-based index into the array, or a string representing the name of the EventProperty: for example, Props.Get(3) or Props.Get("ObjectID")

EventProperty Class

EventProperty objects are always part of an EventProperties collection, and are passed to Add-In methods responding to broadcast events.

EventProperty Attributes

Attribute	Remarks
Description	String Notes: An explanation of what this property represents.
Name	String Notes: A string distinguishing this property from others in the list.
ObjectType	ObjectType Notes: Distinguishes objects referenced through a Dispatch interface.
Value	Variant Notes: A string, number or object reference representing the property value.

ModelWatcher Class

The ModelWatcher object enables an automation client to track changes in a particular model.

ModelWatcher Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

ModelWatcher Methods

Methods	Remarks
GetReloadItem (object Item)	ReloadType Notes: The object that must be reloaded in order to see all changes is returned through the Item parameter. If there are no changes or the entire model must be reloaded, this value is returned as null (C#) or Nothing (VB). Calling this method clears the records so that the next time it is called the return values refer only to new changes. Returns a value from the ReloadType enumeration that specifies which type of change, if any, has occurred. Parameters: <ul style="list-style-type: none">Item: Object
PeekReloadItem	ReloadType Notes: This method behaves identically to 'GetReloadItem()' but does not clear the change record.

Notes

- After your model has been loaded, you only create the ModelWatcher once; if you reload the model, or load another model, the created ModelWatcher is still valid

Package Class

A Package object corresponds to a Package element in the Enterprise Architect Browser window. Packages can be accessed either through the Repository Models collection (a Model is a special form of Package) or through the Packages collection.

Note that a Package has an Element object as an attribute; this corresponds to an Enterprise Architect Package element in the t_object table and is used to associate additional information (such as scenarios and constraints) with the logical Package.

To set additional information for a Package, reference the Element object directly. Also note that if you add a Package to a diagram, you should add an instance of the element (not the Package itself) to the DiagramObject Class for a diagram.

Associated table in repository

t_package

Package Attributes

Attribute	Remarks
Alias	String Notes: Read only Alias
BatchLoad	Long Notes: Read/Write Flag to indicate that the Package is batch loaded during batch import from controlled Packages. Not currently used.
BatchSave	Long Notes: Read/Write Boolean value to indicate whether the Package is included in the batch XMI export list or not.
CodePath	String Notes: Read/Write The path where associated source code is found. Not currently used.
Connectors	Collection Notes: Read only The collection of connectors.
Created	Date Notes: Read/Write Date the Package was created.

Diagrams	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of diagrams contained in this Package.</p>
Element	<p>Element</p> <p>Notes: Read only</p> <p>The associated element object; use to get/set common information such as Stereotype, Complexity, Alias, Author, Constraints, Tagged Values and Scenarios.</p>
Elements	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of elements that belong to this Package.</p>
Flags	<p>String</p> <p>Notes: Read/Write</p> <p>Extended information about the Package.</p>
IsControlled	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the Package has been marked as Controlled.</p>
IsModel	<p>Boolean</p> <p>Notes: Read only</p> <p>Indicates if the Package is a model or a Package.</p>
IsNamespace	<p>Boolean</p> <p>Notes: Read/Write</p> <p>True indicates that 'Package is a Namespace root'.</p> <p>Use 0 and 1 to set False and True.</p>
IsProtected	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the Package has been marked as 'Protected'.</p>
IsVersionControlled	<p>Boolean</p> <p>Notes: Read only</p> <p>Indicates whether or not this Package is under Version Control.</p>
LastLoadDate	<p>Date</p> <p>Notes: Read/Write</p> <p>The date XML was last loaded for the Package.</p>
LastSaveDate	<p>Date</p> <p>Notes: Read/Write</p> <p>The date XML was last saved from the Package.</p>
LogXML	<p>Boolean</p>

	<p>Notes: Read/Write</p> <p>Indicates if XML export information is to be logged.</p>
Modified	<p>Date</p> <p>Notes: Read/Write</p> <p>Date the Package was last modified.</p>
Name	<p>String</p> <p>Notes: Read/Write</p> <p>The name of the Package.</p>
Notes	<p>String</p> <p>Notes: Read/Write</p> <p>Notes about this Package.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Owner	<p>String</p> <p>Notes: Read/Write.</p> <p>The Package owner when using controlled Packages.</p>
PackageGUID	<p>Variant</p> <p>Notes: Read only</p> <p>The global Package ID; valid across models.</p>
PackageID	<p>Long</p> <p>Notes: Read only</p> <p>The local Package ID number.</p> <p>Valid only in this model file.</p>
Packages	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of contained Packages that can be walked through.</p>
ParentID	<p>Long</p> <p>Notes: Read/Write</p> <p>The ID of the Package that is the parent of this one.</p> <p>0 indicates that this Package is a model (that is, it has no parent).</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the element in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p> <p>When setting this attribute, LastError (from the GetLastError method) will be non-empty on error.</p>

TreePos	Long Notes: Read/Write The relative position in the tree compared to other Packages (use to sort Packages).
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
UMLVersion	String Notes: Read/Write The UML version for XMI export purposes.
UseDTD	Boolean Notes: Read/Write Indicates if a DTD is to be used when exporting XML.
Version	String Notes: Read/Write The version of the Package.
XMLPath	String Notes: Read/Write The path to which the XML is saved when using controlled Packages.

Package Methods

Method	Remarks
ApplyGroupLock (string aGroupName)	Boolean Notes: Applies a group lock to the Package object, for the specified group, on behalf of the current user. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information. Parameters: <ul style="list-style-type: none"> aGroupName: String - The name of the security group for which to apply the lock
ApplyGroupLockRecursive (string aGroupName, boolean IncludeElements, boolean IncludeDiagrams, boolean IncludeSubPackages)	Boolean Notes: Applies a group lock to the Package object, object, and all of the Package, diagrams and elements contained within that Package, for the specified group, on behalf of the current user. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail. Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.

	<p>Parameters</p> <ul style="list-style-type: none"> • aGroupName: String - The name of the security group for which to apply the lock • IncludeElements: Boolean - Recursively apply group lock to child elements • IncludeDiagrams: Boolean - Recursively apply group lock to child diagrams • IncludeSubPackages: Boolean - Recursively apply group lock to child Packages
ApplyUserLock ()	<p>Boolean</p> <p>Notes: Applies a user lock to the Package object for the current user. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.</p>
ApplyUserLockRecursive (boolean IncludeElements, boolean IncludeDiagrams, boolean IncludeSubPackages)	<p>Boolean</p> <p>Notes: Applies user locks to the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • IncludeElements: Boolean - Recursively apply user lock to child elements • IncludeDiagrams: Boolean - Recursively apply user lock to child diagrams • IncludeSubPackages: Boolean - Recursively apply user lock to child Packages
Clone	<p>LDISPATCH</p> <p>Notes: Inserts a copy of the Package into the same parent as the original Package. Returns the newly-created Package.</p>
FindObject (string DottedID)	<p>LPDISPATCH</p> <p>Notes: Returns a Package, element, attribute or operation matching the parameter DottedID.</p> <p>If the DottedID is not found, an error is returned: <i>Can't find matching object.</i></p> <p>Parameters</p> <ul style="list-style-type: none"> • DottedID: String - Is in the form 'object.object.object' where object is replaced by the name of a Package, element attribute or operation; examples include MyNamespace.Class1, CStudent.m_Name, MathClass.DoubleIt(int)
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
GetTXAlias (string Code, long Flag)	<p>String</p> <p>Notes: Returns the Alias of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long

	<ul style="list-style-type: none"> - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXNote (string Code, long Flag)	<p>String</p> <p>Returns the Notes of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXAlias (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Alias of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated name of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Notes of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
GetTXName (string Code, long Flag)	<p>String</p> <p>Notes: Returns the name of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
ReleaseUserLock ()	<p>Boolean</p> <p>Notes: Releases user locks and group locks from the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail.</p> <p>Returns True if the operation is successful; returns False if the operation is</p>

	<p>unsuccessful. Use <code>GetLastError()</code> to retrieve error information.</p>
<p><code>ReleaseUserLockRecursive</code> (boolean <code>IncludeElements</code>, boolean <code>IncludeDiagrams</code>, boolean <code>IncludeSubPackages</code>)</p>	<p>Boolean</p> <p>Notes: Releases user locks from the Package object, and all of the Packages, diagrams and elements contained within that Package. User Security applies to the use of this function; if the user does not have permission to apply or release locks on elements, diagrams and Packages, the operation will fail.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use <code>GetLastError()</code> to retrieve error information.</p> <p>Parameters</p> <p><code>IncludeElements</code>: Boolean - Recursively release user locks from child elements</p> <p><code>IncludeDiagrams</code>: Boolean - Recursively release user locks from child diagrams</p> <p><code>IncludeSubPackages</code>: Boolean - Recursively release user locks from child Packages</p>
<p><code>SetReadOnly</code> (boolean <code>ReadOnly</code>, boolean <code>IncludeSubPkgs</code>)</p>	<p>Void</p> <p>Notes: Sets a Package Flag to mark a Package as <code>ReadOnly=1</code>.</p> <p>If Project Security is enabled, the user must have 'Configure Packages' permission to use this method.</p> <p>Throws an exception if the operation fails due to the user not having 'Configure Packages' permission; use <code>GetLastError()</code> to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • <code>ReadOnly</code>: Boolean - Sets or clears the Read Only flag on the Package(s); if: <ul style="list-style-type: none"> False, any Read Only flag is removed from the Package True, a Read Only flag is applied to the Package • <code>IncludeSubPkgs</code>: Boolean - Indicates whether to set/reset the Read Only flag on just the object Package, or on the object Package and all of the nested sub-Packages that it contains; if: <ul style="list-style-type: none"> False, only the flag on the object Package is set or cleared True, flags are set (or cleared, according to the <code>ReadOnly</code> parameter) for the object Package plus all of the nested sub-Packages that it contains <p>When working with Version Controlled Packages, the Read Only flag can be applied to Packages whether they are checked-in or checked-out.</p> <p>User Security applies to setting this flag - if you are prevented from editing the Package, you are also prevented from setting the flag.</p>
<p><code>Update ()</code></p>	<p>Boolean</p> <p>Notes: Updates the current Package object after modification or appending a new item.</p> <p>If False is returned, check the <code>GetLastError()</code> function for more information.</p> <p>Note that a Package object also has an element component that must be taken into account; the Package object contains information about the Package attributes such as hierarchy or contents.</p> <p>The element attribute contains information about, for example, Stereotypes, Constraints or Files - all the attributes of a typical element.</p>
<p><code>VersionControlAdd</code> (string <code>ConfigGuid</code>, string <code>XMLFile</code>, string <code>Comment</code>, boolean <code>KeepCheckedOut</code>)</p>	<p>Void</p> <p>Notes: Places the Package under Version Control, using the specified Version Control Configuration and the specified XMI filename.</p> <p>Throws an exception if the operation fails; use <code>GetLastError()</code> to retrieve error</p>

	<p>information.</p> <p>It is recommended that the Package be saved using Update() before calling VersionControlAdd(), so that any outstanding changes are not lost.</p> <p>Parameters</p> <ul style="list-style-type: none"> • ConfigGuid: String - Name corresponding to the Unique ID of the Version Control configuration to use • XMLFile: String - Name of the XML file to use for this Package; this filename is relative to the Working Copy folder specified for the Config • Comment: String - Log message that is added to the Version Controlled file's history (where applicable) • KeepCheckedOut: Boolean - Specify True to add to Version Control and keep the Package checked-out
VersionControlCheckin (string Comment)	<p>Void</p> <p>Notes: Perform checkin of the Version Controlled Package (also see VersionControlCheckinEx).</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the Version Controlled file's history (where applicable)
VersionControlCheckinEx (string Comment, boolean PreserveCrossPkgRefs)	<p>Void</p> <p>Notes: Perform check-in of the Version Controlled Package.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the Version Controlled file's history (where applicable) • PreserveCrossPkgRefs: Boolean - Flag to indicate whether to preserve or discard pre-existing Cross Package References when checking-in; this parameter overrides the setting in the 'Preferences' dialog, 'XML Specifications' page Unsatisfied cross-Package references are preserved or discarded according to this setting, without prompting the user; see <i>Learn more</i>
VersionControlCheckout (string Comment)	<p>Void</p> <p>Notes: Perform checkout of the Version Controlled Package.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Comment: String - Log message that is added to the Version Controlled file's history (where applicable) <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke the Repository.ScanXMIAndReconcile() method from time to time, following the re-importation of controlled Packages - for example, after using Package.VersionControlGetLatest() to update a number of Packages, or after performing a number of Package check-outs.</p>
VersionControlGetLatest	Void

(boolean ForceImport)	<p>Notes: Updates the local working copy of the Package file associated with the object Package, before re-importing the Package data from the Package file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ForceImport: Boolean - Used if the Package data in the model is found to be up-to-date with respect to the Version Controlled Package file; if: <ul style="list-style-type: none"> False, the Package data that exists in the model is accepted as being up-to-date and no attempt is made to re-import data from the Package file True, the system re-imports the Package from the Package file regardless <p>See also the menu option 'Version Control Get Latest'.</p> <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke the 'Repository.ScanXMIAndReconcile()' method from time to time, following the re-importation of controlled Packages - for example, after using 'Package.VersionControlGetLatest()' to update a number of Packages, or after performing a number of Package check-outs.</p>
VersionControlGetStatus ()	<p>Long</p> <p>Notes: Returns the Version Control status of the Package, as recorded in the current project database.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>Return value maps to this enumerated type:</p> <pre>enum EnumCheckOutStatus { csUncontrolled = 0, csCheckedIn, csCheckedOutToThisUser, csReadOnlyVersion, csCheckedOutToAnotherUser, csOfflineCheckedIn, csCheckedOutOfflineByUser, csCheckedOutOfflineByOther, csDeleted, };</pre> <ul style="list-style-type: none"> csUncontrolled - Either unable to communicate with the Version Control provider associated with the Package, or the Package file is unknown to the provider csCheckedIn - The Package is not checked-out to anybody in the current project database csCheckedOutToThisUser - The Package is marked as checked-out to the current user, in the current project database csReadOnlyVersion - The Package is marked as read-only; an earlier revision of the Package has been retrieved from Version Control csCheckedOutToAnotherUser - The Package is marked as checked-out in the current project database, by a user other than the current user csOfflineCheckedIn - The Package is not checked-out to anybody in the current project database; however, the Version Control configuration associated with the Package was unable to connect to the VC server csCheckedOutOfflineByUser - The Package was 'checked out' in this database,

	<p>by this user, whilst disconnected from Version Control</p> <ul style="list-style-type: none"> • csCheckedOutOfflineByOther - The Package was checked out in this project database, by another user, whilst disconnected from Version Control • csDeleted - The Package file has been deleted from Version Control
VersionControlPutLatest (string CheckInComment)	<p>Void</p> <p>Notes: Perform a checkin of the Version Controlled Package, whilst keeping the Package checked-out.</p> <p>Throws an exception if the operation fails; use GetLastError() to retrieve error information.</p> <p>When a Package that was previously marked as Checked Out Offline, is successfully 'Put' (checkedin) to Version Control, that Package's flags are updated to clear the Checked Out Offline indicator.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Comment: String - Log message added to the Version Controlled file's history (where applicable)
VersionControlRemove ()	<p>Void</p> <p>Notes: Removes Version Control from the Package.</p> <p>Throws an exception if the operation fails; use 'GetLastError()' to retrieve error information.</p>
VersionControlResynchPkgStatus (boolean ClearSettings)	<p>Notes: Synchronizes the Version Control status of the single object Package recorded in your current model with the Package status reported by your Version Control provider.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ClearSettings: Boolean - used if the Package file associated with the specified Package is reported by the Version Control provider as uncontrolled; if ClearSettings is: <ul style="list-style-type: none"> True, the Version Control settings are cleared from the Package False, the Version Control settings remain unchanged

ProjectIssues Class

A ProjectIssue is a system-level Issue that indicates a problem or risk associated with the system as a whole. ProjectIssues can be accessed using the Repository Issues collection.

Associated table in repository

t_issues

ProjectIssues Attributes

Attribute	Remarks
Category	String Notes: Read/Write The category this issue belongs to.
Date	Date Notes: Read/Write The date the issue item was created.
DateResolved	Date Notes: Read/Write The date the issue was resolved.
Name	String Notes: Read/Write The issue name (that is, the issue itself).
IssueID	Long Notes: Read only The ID of this issue.
Notes	String Notes: Read/Write The associated description of the issue.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Owner	String Notes: Read/Write The owner of the issue.

Priority	String Notes: Read/Write The issue priority - Low, Medium or High.
Resolution	String Notes: Read/Write A description of the resolution.
Resolver	String Notes: Read/Write The name of the person resolving the issue.
Severity	String Notes: Read/Write The issue severity - Low, Medium or High.
Status	String Notes: Read/Write The current status of the issue.

ProjectIssues Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Issue object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ProjectResource Class

A Project Resource is a named person who is available to work on the current project in any capacity. ProjectResources can be accessed using the Repository Resources collection.

Associated table in repository

t_resources

ProjectResource Attributes

Attribute	Remarks
Email	String Notes: The resource's email address.
Fax	String Notes: The resource's fax number.
Mobile	Variant Notes: The resource's mobile number, if available.
Name	String Notes: The name of the resource.
Notes	String Notes: A description of the resource, if appropriate.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Organization	Package Class : String Notes: The organization the resource is associated with.
Phone1	Variant Notes: The resource's main telephone number.
Phone2	Variant Notes: The resource's alternative telephone number.
Roles	String Notes: The roles this resource can play in the current project.

ProjectResource Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Resource object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ProjectRole Class

A ProjectRole object represents a named project role. ProjectRoles can be accessed using the Repository ProjectRole collection.

Associated table in repository

t_projectroles

ProjectRole Attributes

Attribute	Remarks
Description	String Notes: Read/Write The project role item description.
Notes	String Notes: Read/Write Notes about the project role item.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Role	String Notes: Read/Write The project role item name.

ProjectRole Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current ProjectRole object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

PropertyType Class

A PropertyType object represents a defined property that can be applied to UML elements as a Tagged Value. PropertyTypes can be accessed using the Repository PropertyTypes collection.

Each PropertyType corresponds to one of the predefined Tagged Values for the model.

Associated table in repository

t_propertytypes

PropertyType Attributes

Attribute	Remarks
Description	String Notes: Read/Write A short description of the property.
Detail	String Notes: Read/Write Configuration information for the property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Tag	String Notes: Read/Write The name of the property (Tag Name).

PropertyType Methods:

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current PropertyType object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Reference Class

This Interface provides access to the various lookup tables within Enterprise Architect. Use the Repository `GetReferenceList()` method to get a handle to a list.

`GetReferenceList` (string Type)

Notes: Uses the list type to get a pointer to a Reference List object.

Parameters:

Type: String - specifies the list type to get; valid list types are:

- Diagram
- Element
- Constraint
- Requirement
- Connector
- Status
- Cardinality
- Effort
- Metric
- Scenario
- Status
- Test
- List:DifficultyType
- List:PriorityType
- List:TestStatusType
- List:ConstStatusType

Example:

```
var statusList as EA.Reference;
statusList = Repository.GetReferenceList("Status");
Session.Output("Status Count: " + statusList.Count);
for (var i=0; i < statusList.Count; i++)
{
    Session.Output("#" + (i+1) + ": " + statusList.GetAt(i));
}
```

Reference Attributes

Attribute	Remarks
Count	Short Notes: A count of items in the list.
ObjectType	ObjectType

	Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: The list type (for example, DiagramTypes).

Reference Methods

Method	Remarks
GetAt(short Index)	String Notes: Get the item at the specified index. Parameters: <ul style="list-style-type: none">• Index: Short - The index of the item to retrieve from the list
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Refresh()	Short Notes: Refresh the current list and return the count of items.

Classe Référentiel

The Repository is the main container of all structures such as models, Packages and elements. You can begin accessing the model iteratively using the Models collection. The Repository also has some convenient methods to directly access the structures without having to locate them in the hierarchy first.

Associated table in repository

<none>

Repository Attributes

Attribute	Remarks
Authors	<p>Collection</p> <p>Notes: Read only</p> <p>This is the system Authors collection containing 0 or more Author objects, each of which can be associated with, for example, elements or diagrams as the item author or owner.</p> <p>Use AddNew(), Delete() and GetAt() to manage Authors.</p>
BatchAppend	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Set this property to True when your automation client has to rapidly insert many elements, operations, attributes and/or operation parameters.</p> <p>Set to False when work is complete.</p> <p>This can result in 10- to 20-fold improvement in adding new elements in bulk.</p>
Clients	<p>Collection</p> <p>Notes: Read only</p> <p>A list of Clients associated with the project. You can modify, delete and add new Client objects using this collection.</p>
ConnectionString	<p>String</p> <p>Notes: Read only</p> <p>The filename/connection string of the current Repository.</p> <p>For a connection string, the DBMS repository type is identified by "DBType=n;" where n is a number corresponding to the DBMS type, as shown:</p> <ul style="list-style-type: none"> 0 - MYSQL 1 - SQLSVR 3 - ORACLE 4 - POSTGRES 8 - ACCESS2007 9 - FIREBIRD 10 - SQLITE

CurrentSelection	Notes: Read only Provides information on what is selected, and in what location without making any requests to the database.
DataMinerManager	Data Miner object Notes: Returns a pointer to the EA.DataMinerManager interface.
Datatypes	Collection Notes: Read only The Datatypes collection. This contains a list of Datatype objects, each representing a data type definition for either data modeling or code generation purposes.
EAEdition	EAEditionTypes Notes: Read only Returns the current level of core licensed functionality available. This property returns Corporate when the edition is Unified or Ultimate. Use 'EAEditionEx' to identify which of these extended editions is available.
EAEditionEx	EAEditionTypes Notes: Read only Returns the current level of extended licensed functionality available (Unified or Ultimate).
EnableCache	Boolean Notes: Read/Write An optimization for pre-loading Package objects when dealing with large sets of automation objects.
EnableUIUpdates	Boolean Notes: Read/Write Set this property to False to improve the performance of changes to the model; for example, bulk addition of elements to a Package. To reveal changes to the user, call 'Repository.RefreshModelView()'.
FlagUpdate	Boolean Notes: Read/Write Instructs Enterprise Architect to update the Repository with the LastUpdate value.
InstanceGUID	String Notes: Read only The identifier string identifying the Enterprise Architect runtime session.
IsSecurityEnabled	Boolean Notes: Read only Indicates whether User Security is enabled for the current repository.
Issues	Collection Notes: Read only

	The System Issues list. Contains ProjectIssues objects, each detailing a particular issue as it relates to the project as a whole.
LastUpdate	String Notes: Read only The identifier string identifying the Enterprise Architect runtime session and the timestamp for when it was set.
LibraryVersion	Long Notes: Read only The build number of the Enterprise Architect runtime.
Models	Collection of type Package Notes: Read only Models are of type Package and belong to a collection of Packages. This is the top level entry point to an Enterprise Architect project file. Each model is a root node in the Browser window and can contain items such as Views and Packages. A model is a special form of a Package; it has a ParentID of 0. By iterating through all models, you can access all the elements within the project hierarchy. You can also use the AddNew() function to create a new model. A model can be deleted, but remember that everything contained in the model is deleted as well.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through the Dispatch interface.
ProjectGUID	String Notes: Read only Returns the unique ID for the project.
ProjectRoles	Collection Notes: Read only The system Roles collection containing 0 or more Role objects, each of which can be associated with, for example, elements or diagrams as the item author or owner. Use AddNew(), Delete() and GetAt() to manage Roles.
PropertyTypes	Collection Notes: Read only Collection of Property Types available to the Repository.
Resources	Collection Notes: Read only Contains available ProjectResource objects to assign to work items within the project. Use the 'Add New()', 'Modify()' and 'Delete()' functions to manage resources.
SearchWindow	Notes: Read only Returns a reference to the Enterprise Architect Search Window.

SecurityUser	Notes: Read only Provides information about the currently logged in security user.
Stereotypes	Collection Notes: Read only The Stereotype collection. A list of Stereotype objects that contain information on a stereotype and the elements it can be applied to.
SuppressEADialogs	Boolean Notes: Read/Write Set this property in the EA_OnPostNewElement broadcast event to control whether Enterprise Architect should suppress showing the default 'Properties' dialog to the user when an element is created.
SuppressSecurityDialog	Boolean Notes: Read/Write Suppress the login prompt dialog that appears by default when username and password parameters passed to OpenFile2 are invalid. For use by external automation clients only.
Tasks	Collection Notes: Read only A list of system tasks (to do list). Each entry is a Task Item; you can modify, delete and add new tasks.
Terms	Collection Notes: Read only The Project Glossary Terms. Each Term object is an entry in the Glossary. Add, modify and delete Terms to maintain the Glossary.

Repository Methods

Method	Remarks
ActivateDiagram (long DiagramID)	Notes: Activates an already open diagram (that is, makes it the active tab) in the main Enterprise Architect user interface. Parameters: <ul style="list-style-type: none"> DiagramID: Long - the ID of the diagram to make active
ActivatePerspective (string long)	Boolean Notes: Deprecated - no longer in use.
ActivateTab (string Name)	Notes: Activates an open Enterprise Architect tabbed view. Parameters: <ul style="list-style-type: none"> Name: String - the name of the view to activate
ActivateTechnology (string)	Notes: Activates an enabled MDG Technology.

TechnologyID)	<p>Parameters:</p> <ul style="list-style-type: none"> TechnologyID: String - the ID of the Technology to activate, as assigned in the MDG Technology Wizard
ActivateToolbox (string Toolbox, long Options)	<p>Boolean</p> <p>Notes: Activates a Toolbox page in the GUI. The returned value is reserved for future use.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Toolbox: String - the name of the Toolbox page to activate Options: Long - reserved for future use
AddDefinedSearches (string sXML)	<p>Notes: Used to enter a set of defined searches that last in Enterprise Architect for the life of the application; when Enterprise Architect loads again they must be inserted again by your Add-In.</p> <p>Parameters:</p> <ul style="list-style-type: none"> sXML: String - the XML of the defined searches; you can get this XML by performing an export of the searches from the 'Manage Searches' dialog in Enterprise Architect
AddDocumentationPath (string Name, string Path, long Type)	<p>Notes: Provides an Add-In with the ability to insert a book path into the Enterprise Architect installation directory, to display Learning Center pages on user-authored subjects (such as use of the Add-In).</p> <p>Parameters:</p> <ul style="list-style-type: none"> Name: String - the top-level (root) name for the Learning Center documentation hierarchy for the Add-In (for example, Enterprise Architect) Path: String - the directory path to the folder to contain the Learning Center documentation structure (for example, C:\Program Files (86)\Sparx Systems\EA\Books Type: Long - reserved for future use; set to 0
AddPerspective (string Perspective, long Options)	<p>Boolean</p> <p>Notes: Deprecated - no longer in use.</p>
AddPropertiesTab (string TabName, string PropXML)	<p>Notes: Create a Properties tab. Returns a PropertiesTab interface if a tab was created successfully, otherwise NULL.</p> <p>Parameters:</p> <ul style="list-style-type: none"> TabName: String - Name of the Properties tab PropXML: String - An XML string defining the values in the tab <p>Example XML string.</p> <pre><?xml version='1.0'?> <properties> <group name='theGroup1'> <property id='1' type='text' default="" readonly='false' > <name>TestText</name> <description>this has id=1</description> </property> <property id='2' type='combobox' default="" readonly='false' ></pre>


```
<name>TestCombo</name>
<value>Two</value>
<description>this has id=2</description>
<valuelist>
  <item>One</item>
  <item>Two</item>
  <item>Three</item>
</valuelist>
</property>
<property id='3' type='date' default='currentdate' showcheckbox='false'
readonly='false' >
  <name>TestDate</name>
  <value></value>
  <description>this has id=3</description>
</property>
<property id='4' type='checkbox' default='true' readonly='false' >
  <name>TestCheckbox</name>
  <description>this has id=4</description>
</property>
<property id='5' type='spin' default='1' min='0' max='100' readonly='false' >
  <name>TestSpin</name>
  <value>7</value>
  <description>this has id=5</description>
</property>
<property id='6' type='int' default='1' readonly='false' >
  <name>TestInt</name>
  <value>100</value>
  <description>this has id=6</description>
</property>
<property id='7' type='double' default='1' readonly='false' >
  <name>TestDouble</name>
  <value>3.333</value>
  <description>this has id=7</description>
</property>
<property id='8' type='memo' default="" readonly='false' >
  <name>TestMemo</name>
  <value></value>
  <description>this has id=8</description>
</property>
</group>
<group name='theGroup2'>
  <property id='22' type='text' default="" readonly='false' >
    <name>Test1</name>
    <value></value>
    <description>this has id=22</description>
```

	<pre> <valuelist> <item></item> </valuelist> </property> </group> </properties> </pre>
AddTab (string TabName, string ControlID)	<p>activeX custom control</p> <p>Notes: Adds an ActiveX custom control as a tabbed window. Enterprise Architect creates a control and, if successful, returns its Unknown pointer, which can be used by the caller to manipulate the control.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • TabName: String - used as the tab caption • ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AddWindow (string WindowName, string ControlID)	<p>activeX custom control</p> <p>Notes: Adds an ActiveX custom control as a window to the Add-Ins docked window. Enterprise Architect creates a control and, if successful, returns its Unknown pointer, which can be used by the caller to manipulate the control.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • WindowName: String - used as the window title • ControlID: String - the ProgID of the control; for example, "CS_AddinFramework.UserControl1"
AdviseConnectorChange (long ConnectorID)	<p>Notes: Provides an Add-In or automation client with the ability to advise the Enterprise Architect user interface that a particular connector has changed and, if it is visible in any open diagram, to reload and refresh that connector for the user.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ConnectorID: Long - the ID of the connector
AdviseElementChange (long ObjectID)	<p>Notes: Provides an Add-In or automation client with the ability to advise the Enterprise Architect user interface that a particular element has changed and, if it is visible in any open diagram, to reload and refresh that element for the user.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ObjectID: Long - the ID of the element
CallSBPI (string sbpiPrefix, string Method, string packedParameters)	<p>Notes: Returns a JSON string with the result from the external server.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • sbpiPrefix: String - Prefix value of the external server • Method: String - Name of the function to call on the external server • [Optional] packedParameters: String - For SBPI Integrations this must match the expected parameters for the specified method; for Custom Services this can pass generic data to the SBPI in any format, but it is suggested you use the packing methods to ensure a correct JSON string structure
ChangeLoginUser (string Name, string Password)	<p>Boolean</p> <p>Notes: Sets the currently logged on user to be the one specified by a name and password; this logs the user into the repository when security is enabled. If security is not enabled an exception (Security not enabled) is thrown.</p>

	<p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the user • Password: String - the password of the user
ClearAuditLogs (Object StartDateTime, Object EndDateTime)	<p>Boolean</p> <p>Notes: Clears all Audit Logs from the model. If StartDateTime and EndDateTime are not null then only log items that fall into this period are cleared. Returns True for success, False for failure.</p> <ul style="list-style-type: none"> • This method cannot be undone; it is strongly advised that you call 'SaveAuditLogs' first to backup the logs • This method might fail if the user logged into the model does not have the correct access permission <p>Parameters:</p> <ul style="list-style-type: none"> • StartDateTime: Variant (DateTime) - the earliest date and time of log entries to clear • EndDateTime: Variant (DateTime) - the latest date and time of log entries to clear
ClearOutput (string Name)	<p>Notes: Removes all the text from a tab in the System Output window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the tab to remove text from
CloseAddins ()	<p>Notes: Called by automation controllers to ensure that Add-Ins created in .NET do not linger after all controller references to Enterprise Architect have been cleared.</p>
CloseDiagram (long DiagramID)	<p>Notes: Closes a diagram in the current list of diagrams that Enterprise Architect has open.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to close
CloseFile ()	<p>Notes: Closes any open file.</p>
CodeMinerService()	<p>object de service Code Miner</p> <p>Notes : renvoie un pointeur vers l'interface EA.CMSservice.</p>
CreateDocumentGenerator()	<p>Document Generator</p> <p>Notes: Returns a pointer to the EA.DocumentGenerator interface.</p>
CreateModel (CreateModelType CreateType, string FilePath, long ParentWnd)	<p>Boolean</p> <p>Notes: Creates a new .eap model file based on the standard Enterprise Architect Base model, or a shortcut .eap based on a provided SQL connection. Returns True when the new file is created, otherwise returns False.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CreateType: CreateModelType - Specify whether to make a new copy of the EABase.eap model, or create a .eap file shortcut to a DBMS repository; the latter option requires a dialog to be opened for the user to provide SQL connection details • FilePath: String - Destination for new .eap file • ParentWnd: Long - Window handle to act as the parent for the 'SQL

	connection' dialog; only required when using cmEAPFromSQLRepository
CreateOutputTab (string Name)	Notes: Creates a tab in the System Output window. Parameters: <ul style="list-style-type: none"> Name: String - the name of the tab to create
DeletePerspective (string Perspective, long Options)	Boolean Notes: Deprecated - no longer in use.
DeleteTechnology (string ID)	Boolean Notes: Removes a specified MDG Technology resource from the repository. Returns True if the technology is successfully removed from the model. Returns False otherwise. <ul style="list-style-type: none"> This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies) Parameters: <ul style="list-style-type: none"> ID: String - the ID of the technology
EnsureOutputVisible (string Name)	Notes: Checks that a specified tab in the System Output window is visible to the user. The System Output window is made visible if it is hidden. Parameters: <ul style="list-style-type: none"> Name: String - the name of the tab to make visible
ExecutePackageBuildScript (long ScriptOptions, string PackageGuid)	Notes: Helps you to run the active Package build script based on your current selection in the Browser window. You can also run a script by passing in the Package GUID. Parameters: <ul style="list-style-type: none"> ScriptOptions: Long - the script type; can be any one of these numerical values: 1 = Build 2 = Test 3 = Run 4 = Create Workbench Instance 5 = Debug PackageGuid: String - the ID of the Package for which to run the script
Exit	Notes: Shuts down Enterprise Architect immediately. Used by .NET programmers where the garbage collector does not immediately release all referenced COM objects.
ExtractImagesFromNote (string Notes, string WriteImagePath, string RelativeImagePath)	String Notes: Writes any Image Manager links to the WriteImagePath directory. Returns a modified notes text, which contains links to the images using the RelativeImagePath parameter. Parameters: <ul style="list-style-type: none"> Notes: String - the notes of the selected Package, diagram or element WriteImagePath: String - the path where the image file links will be stored; this path must exist

	<ul style="list-style-type: none"> RelativeImagePath: String - the path to be inserted into the modified string indicating where the images can be found (for example, "..\images\")
ExtractSBPIParameter (string packedParameters, string name)	<p>Notes: Returns the value of the parameter name as a string.</p> <p>Parameters:</p> <ul style="list-style-type: none"> packedParameters: String - The JSON string to append the Name/Value to; cannot be empty name: String - The name of the parameter
GenerateMDGTechnology (string Filename)	<p>Boolean</p> <p>Notes: Generates an MDG Technology file using the settings in the given MTS file. The returned value indicates success or failure.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Filename: String - the name and path of the MTS file to use
GetActivePerspective ()	<p>String</p> <p>Notes: Deprecated - no longer in use.</p>
GetAttributeByGuid (string Guid)	<p>Attribute</p> <p>Notes: Returns a pointer to an attribute in the repository, located by its GUID. This is usually found using the AttributeGUID property of an attribute.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Guid: String - the GUID of the attribute to locate
GetAttributeByID (long AttributeID)	<p>Attribute</p> <p>Notes: Returns a pointer to an attribute in the repository, located by its ID. This is usually found using the AttributeID property of an attribute.</p> <p>Parameters:</p> <ul style="list-style-type: none"> AttributeID: Long - the ID of the attribute to locate
GetConnectorByGuid (string Guid)	<p>Connector</p> <p>Notes: Returns a pointer to a connector in the repository, located by its GUID. This is usually found using the ConnectorGUID property of a connector.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Guid: String - the GUID of the connector to locate
GetConnectorByID (long ConnectorID)	<p>Connector</p> <p>Notes: Searches the repository for a connector with a specific ID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> ConnectorID: Long - the ID of the connector to locate
GetContextItem (object Item)	<p>ObjectType</p> <p>Notes: Sets a pointer to an item in context within Enterprise Architect. Also returns the corresponding ObjectType.</p> <p>For additional information about ContextItems and the supported ObjectTypes see the 'GetContextItemType' method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> Item: Object - the item to point to

GetContextItemType ()	<p>ObjectType</p> <p>Notes: Returns the ObjectType of an item in context within Enterprise Architect. A ContextItem is defined as an item selected anywhere within the Enterprise Architect GUI including:</p> <ul style="list-style-type: none"> • An item selected in the Browser window • An item selected in an open diagram • An item selected in certain dialogs, such as the attribute 'Properties' dialog <p>The supported ObjectTypes can be any one of these values:</p> <ul style="list-style-type: none"> • otElement • otPackage • otDiagram • otAttribute • otMethod • otConnector
GetContextObject ()	<p>Object</p> <p>Notes: Returns the current context Object.</p>
GetCounts ()	<p>String</p> <p>Notes: Returns a set of counts from a number of tables within the base Enterprise Architect repository. These can be used to determine whether records have been added or deleted from the tables for which information is retrieved.</p>
GetCurrentDiagram ()	<p>Diagram</p> <p>Notes: Returns a selected diagram.</p>
GetCurrentLoginUser (boolean GetGuid)	<p>String</p> <p>Notes: If security is not enabled in the repository, an error is generated.</p> <p>If 'GetGuid' is True, a GUID generated by Enterprise Architect representing the user is returned; otherwise the text as entered in System Users/User Details/Login is returned.</p>
GetDiagramByGuid (string Guid)	<p>Diagram</p> <p>Notes: Returns a pointer to a diagram using the global reference ID (global ID). This is usually found using the diagram GUID property of an element, and stored for later use to open a diagram without using the collection GetAt() function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the diagram to locate
GetDiagramByID (long DiagramID)	<p>Diagram</p> <p>Notes: Gets a pointer to a diagram using an absolute reference number (local ID). This is usually found using the DiagramID property of an element, and stored for later use to open a diagram without using the collection GetAt() function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to locate
GetElementByGuid (string Guid)	<p>Element</p> <p>Notes: Returns a pointer to an element in the repository, using the element's GUID reference number (global ID). This is usually found using the ElementGUID</p>

	<p>property of an element, and stored for later use to open an element without using the collection 'GetAt ()' function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the element to locate
GetElementByID (long ElementID)	<p>Element</p> <p>Notes: Gets 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 to open an element without using the collection GetAt () function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementID: Long - the ID of the element to locate
GetElementsByQuery (string QueryName, string SearchTerm)	<p>Collection (of type Element)</p> <p>Notes: Helps you to run a search in Enterprise Architect, returning the result as a collection.</p> <p>For example: GetElementsByQuery('Simple','Class1'), where the results list elements with 'Class1' in the 'Name' and 'Notes' fields.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • QueryName: String - the name of the search to run, for example 'Simple' • SearchTerm: String - the term to search for
GetElementSet (string IDList, long Options)	<p>Collection (of type Element)</p> <p>Notes: Returns a set of elements as a collection based on a comma-separated list of ElementID values. By default, if no values are provided in the IDList parameter, all objects for the entire project are returned.</p> <p>Parameters</p> <ul style="list-style-type: none"> • IDList: String - a comma-separated list of ElementID values • Options: Long - modifies default behavior of this method <ol style="list-style-type: none"> 1. Returns empty collection when empty IDList parameter is given. 2. Use IDList string as an SQL query to populate this collection. 3. Use IDList string as a Package ID to populate the collection with elements under the given package. (One level only).
GetFieldFromFormat (string Format, string Text)	<p>String</p> <p>Notes: Converts a field from your preferred format to Enterprise Architect's internal format; returns the field in that format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Format: String - The format to convert the field from; valid formats are: <ul style="list-style-type: none"> - HTML - Full HTML - RTF - Rich Text Format - TXT - Plain text • Text: String - The field to be converted
GetFormatFromField (string Format, string Text)	<p>String</p> <p>Notes: After accessing a field that contains formatting, use this method to convert it to your preferred format; returns the field in the format specified.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Format: String - The format to convert the field to; valid formats are: <ul style="list-style-type: none"> - HTML - Full HTML - RTF - Rich Text Format

	<p>- TXT - Plain text</p> <ul style="list-style-type: none"> • Text: String - The field to be converted
GetFormattedName (string Guid, long FlagInclude, string Separator, long FlagFormat)	<p>String</p> <p>Notes: Provides special formatting for the name of the specified object; for example, the fully qualified name of a specific element or feature.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - The GUID of the object to be formatted • FlagInclude: Long - Items to be included in the formatted name: <ul style="list-style-type: none"> - fiFeature = &H01 - fiClass = &H02 - fiParents = &H04 - fiPackage = &H08 - fiRootNS = &H10 - fiHiddenNS = &H20 - fiDiagram = &H40 - fiElemAlias = &H80 • Separator: String - The string to use for separating each included item (such as Packages or elements) • FlagFormat: Long - Additional formatting options: <ul style="list-style-type: none"> - ffReplaceSpaces = &H01 - ffLowercase = &H02 - ffURLEncode = &H04 <p>Example:</p> <p>FormattedName = Repository.GetFormattedName (Element.ElementGUID, fiFeature Or fiClass Or fiParents Or fiPackage Or fiDiagram, ":", 0)</p>
GetGapAnalysisMatrix ()	<p>String</p> <p>Notes: Read Only</p> <p>Returns all Gap Analyses as an XML document.</p>
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
GetLocalPath (string Type, string Path)	<p>String</p> <p>Notes: Returns the expanded local file path for code generated from an element, with reference to the Type and Path defined in the 'Local Paths' dialog.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Type: String - the coding language for the element, such as Java, C or C++ • Path: String - the local path to be expanded; for example: %Desk%\Javacode\Motor.java <p>For example:</p> <p>Repository.GetLocalPath (Java, %Desk%\Javacode\Motor.java)</p> <p>This could return:</p> <p>C:\Users\fbloggs\Desktop\Javacode\Motor.java.</p>
GetMailInterface ()	<p>MailInterface</p> <p>Notes: Returns an instance of the EA.MailInterface; use this interface to automate the process of creating and sending Model Mail messages.</p>

GetMethodByGuid (string Guid)	<p>Method</p> <p>Notes: Returns a pointer to a method in the repository; this is usually found using the MethodGUID property of a method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the method to look for
GetMethodByID (long MethodID)	<p>Method</p> <p>Notes: Returns a pointer to a method in the repository; this is usually found using the MethodID property of a method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • MethodID: Long - the ID of the method to look for
GetPackageByGuid (string Guid)	<p>Package</p> <p>Notes: Returns a pointer to a Package in the repository using the Package's GUID reference number (global ID). This is usually found using the PackageGUID property of the Package.</p> <p>Each Package in the model also has an associated element with the same GUID, so if you have an element with Type="Package" then you can load the Package by calling:</p> <p style="padding-left: 40px;">GetPackageByGuid(Element.ElementGUID)</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Guid: String - the GUID of the Package to look for
GetPackageByID (long PackageID)	<p>Package</p> <p>Notes: 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 to open a Package without using the collection GetAt () function.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageID: Long - the ID of the Package to locate
GetProjectInterface ()	<p>Project</p> <p>Notes: Returns a pointer to the EA.Project interface (the XML-based automation server for Enterprise Architect). Use this interface to work with Enterprise Architect using XML, and also to access utility functions for loading diagrams, running reports and so on.</p>
GetPropertiesTab (string TabName)	<p>Notes: Finds an existing Properties tab.</p> <p>Returns a PropertiesTab interface if the tab exists, otherwise NULL.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • TabName: String - The name of the 'Properties' tab.
GetReferenceList (string Type)	<p>Reference</p> <p>Notes: Uses the list type to get a pointer to a Reference List object.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Type: String - specifies the list type to get; valid list types are: <ul style="list-style-type: none"> - Diagram - Element - Constraint - Requirement - Connector - Status

	<ul style="list-style-type: none"> - Cardinality - Effort - Metric - Scenario - Status - Test - List:DifficultyType - List:PriorityType - List:TestStatusType - List:ConstStatusType
GetRelationshipMatrix ()	<p>String</p> <p>Notes: Returns an XML document (as a string), containing definitions of all Relationship Matrix profiles saved in the current model.</p>
GetTechnologyVersion (string ID)	<p>String</p> <p>Notes: Returns the version of a specified MDG Technology resource.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ID: String - the specified technology ID
GetTreeSelectedElements ()	<p>Collection</p> <p>Notes: Returns the set of elements currently selected in the Browser window as a collection.</p>
GetTreeSelectedItem (object SelectedItem)	<p>ObjectType</p> <p>Notes: Gets an object variable and type corresponding to the currently selected item in the tree view.</p> <p>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.</p> <p>The object passed back through the parameter can be a Package, element, diagram, attribute or operation object.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • SelectedItem: Object - the object to get the variable and type for
GetTreeSelectedItemType ()	<p>ObjectType</p> <p>Notes: Returns the type of the object currently selected in the tree. One of:</p> <ul style="list-style-type: none"> • otDiagram • otElement • otPackage • otAttribute • otMethod
GetTreeSelectedObject ()	<p>Object</p> <p>Notes: The related method GetTreeSelectedItem () has an output parameter that is inaccessible by some scripting languages. As an alternative, this method provides the selected item through the return value.</p>
GetTreeSelectedPackage ()	<p>Package</p> <p>Notes: Returns the Package in which the currently selected tree view object is contained.</p>

HasPerspective (string Perspective)	String Notes: Deprecated - no longer in use.
HideAddinWindow ()	Notes: Hides the docked Add-In window.
ImportPackageAsMDGTechnology (string PackageGuid)	Notes : lorsque PackageGuid contient le GUID d'un paquetage <<mdg technology>>, un MDG Technologie sera créé à partir du contenu du paquetage et importé dans le modèle actuel. Renvoi : True en cas de succès Paramètres: <ul style="list-style-type: none"> PackageGuid : String - le GUID du Paquetage << mdg technology >>
ImportPackageBuildScripts (string PackageGuid, string BuildScriptXML)	Notes: Imports build scripts into a Package in Enterprise Architect. Parameters: <ul style="list-style-type: none"> PackageGuid: String - the GUID of the Package into which to import the build scripts BuildScriptXML: String - the build script XML data, which you can export from within Enterprise Architect
ImportRASAsset (string PackageGUID, string Protocol, string ServerName, string Model, string Storage, string RASGUID, string Password, string Version)	Notes: Imports the specified RAS asset. Returns True on success; check GetLastError on failure. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID of the Package to import the asset to Protocol: String - the protocol the server is using ServerName: String - the name of the RAS server Model: String - the name of the RAS model to use Storage: String - the storage name of the RAS asset RASGUID: String - the GUID of the RAS asset Password: String - the password to access the RAS asset Version: String - the version of the RAS asset to import
ImportTechnology (string Technology)	Boolean Notes: Installs a given MDG Technology resource into the repository. Returns True if the technology is successfully loaded into the model. Otherwise returns False. This applies to technologies imported into pre-7.0 versions of Enterprise Architect (imported technologies), not to technologies referenced in version 7.0 and later (referenced technologies). Parameters: <ul style="list-style-type: none"> Technology: String - the contents of the technology resource file
InsertSBPIPparameter (string packedParameters, string name, string value)	Notes: Returns a JSON string. Parameters: <ul style="list-style-type: none"> packedParameters: String - The JSON string to append the Name/Value to; cannot be empty name: String - The name of the parameter value: String - The value of the parameter
InvokeConstructPicker	

(string ElementFilter)	<p>String</p> <p>Notes: Invokes the 'Select <Item>' dialog with filters on the object type and, optionally, stereotype. Returns the ElementID of the selected object, or 0 if no object was selected when the dialog was closed.</p> <p>For example:</p> <pre>elementid=Repository.InvokeConstructPicker ("IncludedTypes=Class,Component;StereoType=foo,bar")</pre> <p>In this example, the 'Select <item>' dialog will allow the user to select any Class or Component element in the model that has a stereotype of 'foo' or 'bar'. The 'IncludedTypes' and 'StereoType' filters are separated by a semi-colon.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementFilter: String - specifies which elements or Packages are to be made available for selection, based on element types and stereotypes identified by the IncludedTypes and StereoType filters <ul style="list-style-type: none"> - IncludedTypes - (mandatory) comma separated list of element types that can be selected in the dialog; for example: Package,Class,Component - MultiSelect - (optional) when set to True ("MultiSelect=True;") allows the Construct picker to select multiple elements - Selection (optional) - list of comma-separated element GUIDs that will be selected by default - GetNext (optional) - returns the next ID in the list of selected elements, or 0 when no more are available; this option will not display a dialog and assumes the first call was made with MultiSelect=True; - StereoType - (optional) comma separated list of stereotypes that can be selected in this dialog <p>Do not use leading or trailing spaces between element type or stereotype values. Parameter values must be written with the correct case; element type names are also case sensitive.</p> <p>Example:</p> <pre>val = Repository.InvokeConstructPicker ("IncludedTypes=Class; MultiSelect=True;"); while(val != 0) { val = Repository.InvokeConstructPicker("GetNext=True;"); }</pre>
InvokeFileDialog (string FilterString, long Filterindex, long Flags)	<p>String</p> <p>Notes: Opens a standard 'Open File' dialog and returns a string containing the full path to the selected file on success. Returns an empty string if the dialog was canceled.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilterString: String - list of file type filters. • Filterindex: Long - one-based index of the filter to be used by default • Flags: Long - additional bit flags used to initialize the file dialog; see OPENFILENAME structure in MSDN documentation for accepted values
IsTabOpen (string TabName)	<p>String</p> <p>Notes: Checks whether a named Enterprise Architect tabbed view is open and active. This includes open diagram windows or custom controls added using</p>

	<p>'Repository.AddTab ()'.</p> <p>Returns:</p> <ul style="list-style-type: none"> • 2 to indicate that a tab is open and active (top-most) • 1 to indicate that it is open but not top-most, or • 0 to indicate that it is not visible at all <p>Parameters:</p> <ul style="list-style-type: none"> • TabName: String - the name of the tab to check for; TabName is case sensitive
<p>IsTechnologyEnabled (string ID)</p>	<p>Boolean</p> <p>Notes: Checks whether the specified string matches the ID of an enabled MDG Technology in Enterprise Architect.</p> <p>Returns True if the string matches the ID of an enabled Technology. Otherwise returns False.</p> <p>Parameters:</p> <p>ID: String - the technology ID to check for; built-in technology IDs include:</p> <ul style="list-style-type: none"> • ArcGIS ArcGIS • BABOK BABOK • BIZBOK BIZBOK Guide • BPSim BPSim • BRM Business Rule Model • CMMN Case Management Model & Notation • CODEENG Code Engineering • Database Modeling Database Modeling • DMN1.1 DMN1.1 • EAExtended Core Extensions • ERD Entity Relationship Diagram • GML GML • MYSQLTECH MySqlTech • EAReview Review • SIMF SIMF Technology • SOAML SOAML • SysML1.1 SysML1.1 • SysML1.2 SysML1.2 • SysML1.3 SysML1.3 • SysML1.4 SysML1.5 • UML2 Basic UML2 Technology • SYSENG System Engineering • 262139 MDG Technology Builder • TOGAF TOGAF • UAF UAF • UPDM2 UPDM 2.0 • Win32UI Win 32 User Interface Modeling • ZF Zachman Framework <p>Technically, any combination of technologies integrated with or added to Enterprise Architect - including user-developed technologies - could appear in this list. In practice you would only check for one or two technologies at a time.</p>

IsTechnologyLoaded (string ID)	<p>Boolean</p> <p>Notes: Checks whether a specified technology is loaded into the repository. Returns True if the MDG Technology resource is loaded into the repository. Otherwise returns False.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ID: String - the technology ID to check for
LoadAddins ()	<p>Notes: Loads all Add-Ins from a repository when Enterprise Architect is opened from automation.</p>
MarkupNotes (string Notes, string GlossaryType, string replacement)	<p>String</p> <p>Notes:</p> <p>Returns a string containing the translation of the term.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Notes: String - a value to perform a translation markup on • GlossaryType: String - a comma-separated list of glossary types; for example, 'tx-french,tx-global' • replacement: String - the value to replace the TERM when found; "#TERM#,"
OpenDiagram (long DiagramID)	<p>Notes: Provides a method for an automation client or Add-In to open a diagram. The diagram is added to the tabbed list of open diagrams in the main Enterprise Architect view.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to open
OpenFile (string Filename)	<p>Boolean</p> <p>Notes: This is the main point for opening an Enterprise Architect project file from an automation client, and working with the contained objects.</p> <p>If the required project is a DBMS or Cloud based repository, you will require a valid Enterprise Architect connection string. This can be obtained in one of two ways; both methods require you to first make and open a connection to the model in question with Enterprise Architect:</p> <ol style="list-style-type: none"> 1) Using the 'Save as Shortcut' menu item, create a shortcut .eap file containing the database connection string; you can call this shortcut file to access the repository. 2) Alternatively, you can right-click on the model's connection entry in the 'Open Project' screen and select 'Edit connection string', this connection string can then be used direct by OpenFile. <p>Parameters:</p> <ul style="list-style-type: none"> • Filename: String - the filename (or connection string) of the Enterprise Architect project to open
OpenFile2 (string FilePath, string Username, string Password)	<p>Boolean</p> <p>Notes: As for 'OpenFile ()' except this provides for the specification of a password.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - the file path of the Enterprise Architect project to open • Username: String - the user login ID • Password: String - the user password

OpenFileInEditor(string FilePath)	<p>Boolean</p> <p>Notes: Displays a document or source code file in the EA editor</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - the file path of the document or file to display in the editor
OpenFileInEditorAtLine(string FilePath, integer LineNumber)	<p>Boolean</p> <p>Notes: Displays a document or source code file in the EA editor</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - the file path of the document or file to display in the editor • LineNumber: Integer - the line number to highlight.
RefreshModelView (long PackageID)	<p>Notes: Reloads a Package or the entire model, updating the user interface.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageID: Long - the ID of the Package to reload: if 0, the entire model is reloaded; if a valid Package ID, only that Package is reloaded
RefreshOpenDiagrams (boolean FullReload)	<p>Notes: Reloads the diagram contents for all open diagrams from the repository.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FullReload: Boolean - if False only the contents of element compartments are reloaded; if True the full content of each diagram is reloaded
ReloadDiagram (long DiagramID)	<p>Notes: Reloads a specified diagram. This would commonly be used to refresh a visible diagram after code import/export or other batch process where the diagram requires complete refreshing.</p> <p>Calling this method within a call to <i>EA_OnNotifyContextItemModified</i> is not supported</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to be reloaded
ReloadPackage (long PackageID)	<p>Notes: Reloads a Package and its open child diagrams.</p> <p>Parameters:</p> <p>PackageID: Long - The ID of the Package to reload; if a valid Package ID, only that Package is reloaded.</p>
RemoveOutputTab (string Name)	<p>Notes: Removes a specified tab from the System Output window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the tab to be removed
RemoveWindow (string WindowName)	<p>Boolean</p> <p>Notes: Removes an Add-In window that matches the specified WindowName.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • WindowName: String - the name of the window to remove
RepositoryType ()	<p>String</p> <p>Notes: Returns the currently open database/repository type.</p> <p>Can return one of these values:</p> <ul style="list-style-type: none"> • JET (.EAP file, MS Access 97 to 2013 format) • FIREBIRD

	<ul style="list-style-type: none"> • ACCESS2007 (.accdb file, MS Access 2007+ format) • ASA (Sybase SQL Anywhere) • SQLSVR (Microsoft SQL Server) • MYSQL (MySQL) • ORACLE (Oracle) • POSTGRES (PostgreSQL)
RunModelSearch (string sQueryName, string sSearchTerm, string sSearchOptions, string sSearchData)	<p>Notes: Runs a search, displaying the results in Enterprise Architect's Model Search window.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • sQueryName: String - the name of the search to run, for example Simple • sSearchTerm: String - the term to search for • sSearchOptions: String - currently not being used • sSearchData: String - a list of results in the form of XML, which is appended onto the result list in Enterprise Architect - see the <i>XML Format</i> topic; this parameter is not mandatory so pass in an empty string to run the search as per normal
SaveAllDiagrams ()	<p>Notes: Saves all open diagrams.</p>
SaveAuditLogs (string FilePath, object StartDateTime, object EndDateTime)	<p>Boolean</p> <p>Notes: Saves the Audit Logs contained within a model to a specified file. If 'StartDateTime' and 'EndDateTime' are not null then only log items that fall into this period are saved.</p> <p>Returns True for success, False for failure.</p> <ul style="list-style-type: none"> • This might fail if the user logged into the model does not have the correct access permission <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - the file to save the Audit Logs to • StartDateTime: Variant (DateTime) - the earliest date and time of log entries to save • EndDateTime; Variant (DateTime) - the latest date and time of log entries to save
SaveDiagram (long DiagramID)	<p>Notes: Saves an open diagram; assumes the diagram is open in the main user interface Tab list.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramID: Long - the ID of the diagram to save
SaveDiagramAsUMLProfile (string DiagramGUID, string Filename)	<p>Boolean</p> <p>Notes: Saves a given diagram as a UML Profile, using the settings from the previous time that the specific diagram was saved manually.</p> <p>The returned value indicates success or failure.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID of the Profile diagram to save • Filename: String - the name and path of the file to create; if left blank, the method will use the filename from the previous time the specified diagram was saved

<p>SavePackageAsUMLProfile (string PackageGUID, string Filename)</p>	<p>Boolean</p> <p>Notes: Saves a given Package as a UML Profile, using the settings from the previous time that the specific Package was saved manually.</p> <p>The returned value indicates success or failure.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID of the Profile Package to save • Filename: String - the name and path of the file to create; if left blank, the method will use the filename from the previous time the specified Package was saved
<p>ScanXMIAndReconcile ()</p>	<p>Notes: Scans the Package XMI files associated with each of the project's controlled Packages and restores any diagram objects or cross-references that are detected as missing from the project.</p> <p>This function is useful in team environments where each user maintains their own private copy of the model database (that is, multiple private EAP files) and model updates are propagated through the use of controlled Packages; it provides no benefit when the model is hosted in a single shared database that is accessed by all team members.</p> <p>Each controlled Package is compared with its associated XMI file and, if the cross-reference information in the model does not match the XMI, Enterprise Architect updates the model with the information from the XMI and records the update in the System Output window.</p> <p>You can roll back such updates by right-clicking on the entry in the System Output window and selecting the 'Rollback Update' option (or 'Rollback Selected Updates' if multiple entries are selected).</p> <p>Closing the model clears the entries in the System Output window; an entry in this window is also cleared as and when you roll-back the update for it.</p> <p>This functionality is invoked automatically as part of the 'Get All Latest' operation.</p> <p>When working in an environment that uses a Private Model deployment and your model contains a significant number of cross-Package references, it is recommended that you invoke this function from time to time, following the re-importation of controlled Packages - for example, after using 'Get Latest' to update a number of Packages, or after performing a number of Package check-outs.</p> <p>As a general rule, avoid running this function while you have uncommitted changes in your model. Generally, you:</p> <ul style="list-style-type: none"> • Check-out a number of Packages • Invoke 'ScanXMIAndReconcile' • Make your modifications • Commit any outstanding changes before you check-out more Packages and run 'ScanXMIAndReconcile' again
<p>ShowAddinWindow (string TabName)</p>	<p>Boolean</p> <p>Notes: Shows the docked Add-In window on the specified page. Returns True if a tab of the specified name is now displayed.</p> <p>Parameters</p> <ul style="list-style-type: none"> • TabName: String - specifies the tab
<p>ShowDynamicHelp (string Topic)</p>	<p>Notes: Shows a Help topic as a view.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Topic: String - specifies the Help topic

ShowInProjectView (object Item)	<p>Notes: Selects a specified object in the Browser window.</p> <p>Accepted object types are Package, Element, Diagram, Attribute, and Method; an exception is thrown if the object is of an invalid type.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Item: Object - the object to highlight
ShowWindow (long Show)	<p>Notes: Shows or hides the Enterprise Architect User Interface.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Show: Long
ShutdownEA (long Flags)	<p>Notes: Closes all open Views and exits Enterprise Architect.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Flags: long - if set to 1 then all pending changes will be saved before closing. If set to 0 then all changes will be lost.
SQLQuery (string SQL)	<p>String</p> <p>Notes: Enables execution of a SQL select statement against the current repository. Returns an XML formatted string value of the resulting record set.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • SQL: String - contains the SQL Select statement
SynchProfile (string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes Tagged Values and constraints of a UML Profile item using the 'Synch Profiled Elements' dialog.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Profile: String - the name of the profile that contains the stereotype • Stereotype: String - the name of the profile stereotype for which the default tags and constraints are to be synchronized
VCRPS	<p>Type VersionControlResynchPkgStatuses (boolean ClearSettings)</p> <p>Notes: Synchronizes the Version Control status of each Version Controlled Package within the current model with the status reported by your Version Control provider.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ClearSettings: Boolean <ul style="list-style-type: none"> - if True, clear the Version Control settings from Packages that are reported by the Version Control provider as uncontrolled - if False, leave the Version Control settings unchanged for Packages reported as uncontrolled
WriteOutput (string Name, string Output, long ID)	<p>Notes: Writes text to a specified tab in the System Output window, and associates the text with an ID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - specifies the tab on which to display the text • Output: String - specifies the text to display • ID: Long - specifies a numeric ID value to associate with this output item for further handling by Add-Ins; can be set to 0 if no handling is required

SecurityUser Class

A SecurityUser object represents a named security user.

Associated table in repository

None.

SecurityUser Attributes

Attribute	Remarks
Department	String Notes: Read only Returns the current user's department.
FirstName	String Notes: Read only Returns the current user's first name.
FullName	String Notes: Read only Returns the current user's full name.
Login	String Notes: Read only Returns the current user's login name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Surname	String Notes: Read only Returns the current user's surname.

SecurityUser Methods

Method	Remarks
IsMemberOf (string GroupId)	Boolean Returns True if the user is part of the specified security group.

	<p>Parameter:</p> <ul style="list-style-type: none">• GroupId: String - Name of the security group to check.
--	-----------------------------------------------------------------------------------------------------------------------------------

Stereotype 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 repository

t_stereotypes

Stereotype Attributes

Attribute	Description
AppliesTo	String Notes: Read/Write A reference to the stereotype Base Class; that is, which element it applies to.
MetafileLoadPath	String Notes: Read/Write The path to an associated metafile. The Automation Interface does not yet support loading metafiles. To do this you must use the 'Stereotype' tab of the 'UML Types' dialog in Enterprise Architect.
Notes	String Notes: Read/Write. Notes about the stereotype.
Name	String Notes: Read/Write The stereotype name, which appears in the Stereotype drop list for elements that match the AppliesTo attribute.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
StereotypeGUID	String Notes: Read/Write A unique identifier for stereotype, generally set and maintained by Enterprise Architect.
Style	String Notes: Read/Write An additional style specifier for the stereotype.
VisualType	String

	<p>Notes: Read/Write</p> <p>Indicates an inbuilt visual style associated with a stereotype.</p> <p>Not currently implemented.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------

Stereotype Methods

Method	Description
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current stereotype object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Task Class

A Task is an entry in the System Task list. Tasks can be accessed using the Repository Tasks collection.

Associated table in repository

t_tasks

Task Attributes

Attribute	Remarks
ActualTime	Long Notes: Read/Write The time already expended on the task, in hours, days or other units.
AssignedTo	String Notes: Read/Write The person this task is assigned to; that is, the responsible resource.
EndDate	Date Notes: Read/Write The date the task is scheduled to finish.
History	String Notes: Read/Write A memo field to hold, for example, task history or notes.
Name	Variant Notes: Read/Write The task name.
Notes	Variant Notes: Read/Write A description of the task.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Owner	String Notes: Read/Write The task owner.
Percent	Long

	Notes: Read/Write The percentage completion of the task.
Phase	String Notes: Read/Write The phase of the project the task relates to.
Priority	String Notes: Read/Write The priority of this task.
StartDate	Date Notes: Read/Write The date the task is to start.
Status	Variant Notes: Read/Write The current status of the task.
TaskID	Long Notes: Read only The local ID of the task.
TotalTime	Long Notes: Read/Write The total expected time the task might run, in hours, days or some other unit.
Type	String Notes: Read/Write Sets or returns a string representing the type.

Task Methods

Method	Type
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Task object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Term Class

A Term object represents one entry in the system glossary. Terms can be accessed using the Repository Terms collection.

Associated table in repository

t_glossary

Term Attributes

Attribute	Remarks
Meaning	String Notes: Read/Write The description of the term; its meaning.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Term	String Notes: Read/Write The glossary item name.
TermID	Long Notes: Read only A local ID number to identify the term in the model.
Type	String Notes: Read/Write The type this term applies to (for example, business or technical).

Term Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Refresh	Void Notes: Forces Enterprise Architect to reload the Glossary terms from the database.

	If an element is selected, it will have to be re-selected before the 'Note' fields and windows reflect the updated Glossary terms.
Update()	Boolean Notes: Updates the current Term object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Properties Tab Package

The Properties Tab Package contains:

- A function to retrieve a pointer to the interface
- Functions to create or find a Properties tab
- Utility functions for modifying Properties values

You can get a pointer to this interface using the methods `Repository.AddPropertiesTab` and `Repository.GetPropertiesTab`.

PropertiesTab Class

PropertiesTab Attributes

Attribute	Remarks
-----------	---------

PropertiesTab Methods

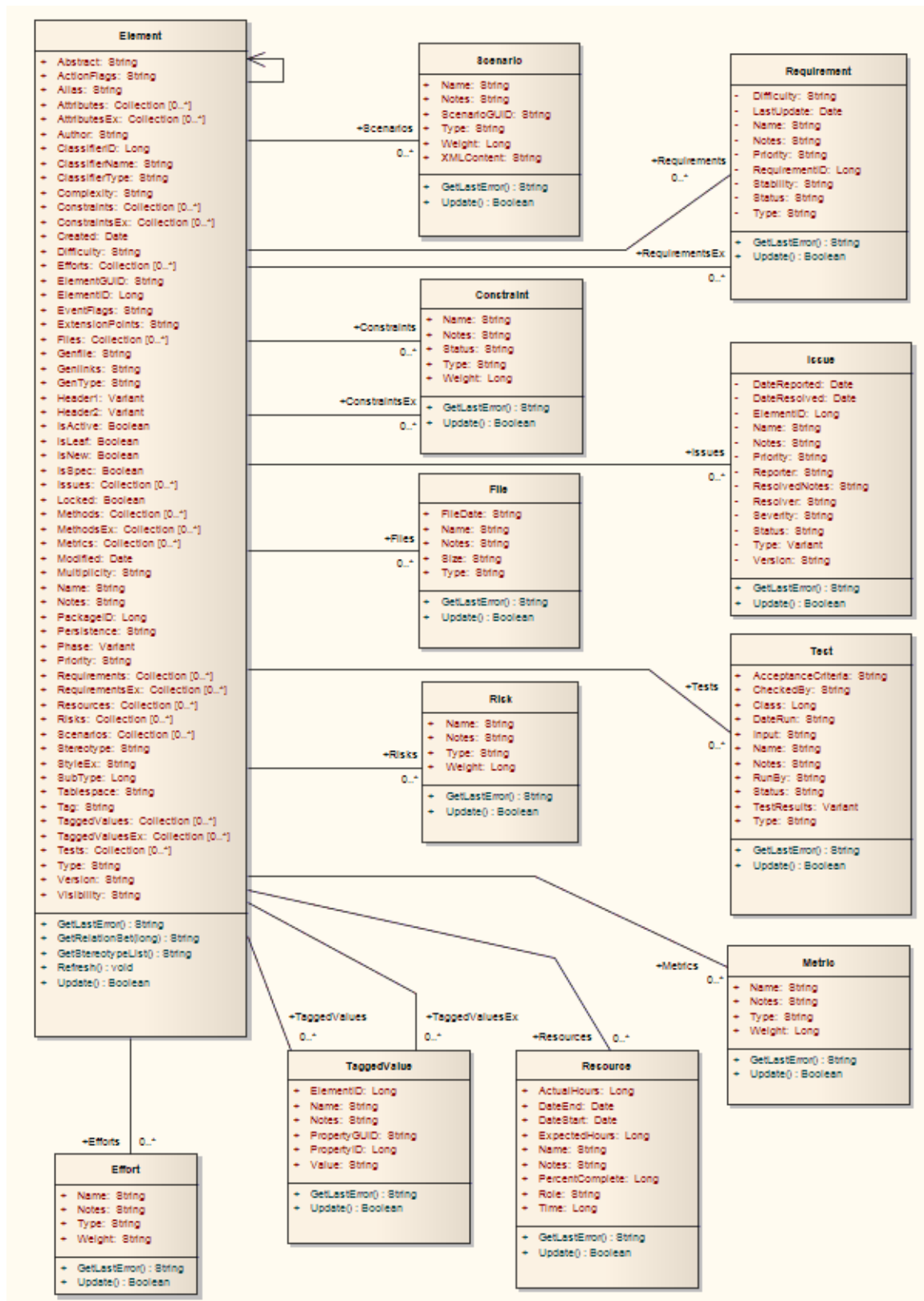
Method	Remarks
AddPropertiesTab (string TabName, string PropXML)	<p>Adds a Properties tab. Returns TRUE if the tab was added.</p> <p>Parameters:</p> <ul style="list-style-type: none"> TabName: String - The name of the Properties tab PropXML: String - An XML string defining the values in the tab
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
GetPropertiesTab (string TabName)	<p>Notes: Locates a Properties tab. Returns TRUE if the tab is found.</p> <p>Parameters:</p> <ul style="list-style-type: none"> TabName: String - The name of the Properties tab
GetPropertiesXML ()	<p>Notes: Returns the XML string of the properties.</p>
GetProperty (long PropID)	<p>Notes: Returns a string of the Property value.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PropID: long - The ID value of the property
RemovePropertiesTab ()	<p>Notes: Removes a Properties tab. Returns TRUE if the tab is removed.</p>
SetPropertiesXML (string PropXML)	<p>Notes: Sets the Properties values in the tab. Returns TRUE if the properties were set successfully.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PropXML: String - An XML string defining the values in the tab
SetProperty (long PropID, string Value)	<p>Notes: Returns TRUE if the value was set successfully.</p> <p>Parameters:</p> <ul style="list-style-type: none"> PropID: long - The ID value of the property to set Value: String - The value to set the property to

Element Package

The Element Package contains information about an element and its associated extended properties such as testing and project management information. An element is the basic item in an Enterprise Architect model. Classes, Use Cases and Components are all different types of UML element.

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 (for example, Scenarios and Tests). It also includes a full description of the element object (the basic model structural unit).

Example



Constraint Class

A Constraint is a condition imposed on an element. Constraints are accessed through the Element Constraints collection.

Associated table in repository

t_objectconstraints

Constraint Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the constraint (that is, the constraint).
Notes	String Notes: Read/Write Notes about the constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only The ElementID of the element to which this constraint applies.
Status	String Notes: Read/Write The current status of the constraint.
Type	String Notes: Read/Write The constraint type.
Weight	Long Notes: Read/Write A weighting factor.

Constraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Constraint object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Effort Class

An Effort is a named item with a weighting that can be associated with an element for purposes of building metrics about the model. Efforts are accessed through the Element Efforts collection.

Associated table in repository

t_objecteffort

Effort Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the effort.
Notes	String Notes: Read/Write Notes about the effort.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The effort type.
Weight	Long Notes: Read/Write A weighting factor.
Weight2	Float Notes: Read/Write A weighting factor.

Effort Methods

Method	Remarks
GetLastError()	String

	Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Effort object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Classe d'élément

An Element is the main modeling unit, corresponding to (for example) a Class, Use Case, Node or Component. You create new elements by adding to the Package Elements collection. Once you have created an element, you can add it to the DiagramObject Class of a diagram to include it in the diagram.

Elements have a collection of connectors. Each entry in this collection indicates a relationship to another element.

There are also some extended collections for managing addition information about the element, including properties such as Tagged Values, Issues, Constraints and Requirements.

Associated table in repository

t_object

Element Attributes

Attribute	Remarks
Abstract	String Notes: Read/Write Indicates if the element is Abstract (1) or Concrete (0).
ActionFlags	String Notes: Read/Write A structure to hold flags concerned with Action semantics.
Alias	String Notes: Read/Write An optional alias for this element.
AssociationClassConnector ID	Long Notes: Read only If the element is an AssociationClass, AssociationClassConnectorID contains the Connector ID of the respective Association connector.
Attributes	Collection Notes: Read only A collection of attribute objects for the current element; use the AddNew and Delete functions to manage attributes.
AttributesEx	Collection Notes: Read only A collection of attribute objects belonging to the current element and its parent elements.
Author	String Notes: Read/Write

	The element author.
BaseClasses	Collection Notes: Read only A list of Base Classes for this element, presented as a collection for convenience.
ClassifierID	Long Notes: Deprecated See ClassifierID
ClassifierID	Long Notes: Read/Write The ElementID of a Classifier associated with this element; that is, the base type. Only valid for instance type elements (such as Object or Sequence).
ClassifierName	String Notes: Read/Write Name of associated Classifier (if any).
ClassifierType	String Notes: Read only Type of associated Classifier.
Complexity	String Notes: Read/Write A complexity value indicating how complex the element is; used for metric reporting and estimation. Valid values are: 1 for Easy, 2 for Medium, 3 for Difficult.
CompositeDiagram	Diagram Notes: Read only If the element is Composite, returns its associated diagram; otherwise returns null.
Connectors	Collection Notes: Read only Returns a collection containing the connectors to other elements.
Constraints	Collection Notes: Read only A collection of Constraint objects.
ConstraintsEx	Collection Notes: Read only Collection of Constraint objects belonging to the current element and its parent elements.
Created	Date Notes: Read/Write

	The date the element was created.
CustomProperties	<p>Collection</p> <p>Notes: Read only</p> <p>List of advanced properties for an element.</p> <p>The collection of advanced properties differs depending on element type; for example, an Action and an Activity have different advanced properties.</p> <p>Currently only editable from the user interface.</p>
Diagrams	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of sub-diagrams (child diagrams) attached to this element as seen in the tree view.</p>
Difficulty	<p>String</p> <p>Notes: Read/Write</p> <p>A difficulty level associated with this element for estimation/metrics; only useable for Requirement, Change and Issue element types, otherwise ignored.</p> <p>Valid values are: Low, Medium, High.</p>
Efforts	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of Effort objects.</p>
ElementGUID	<p>String</p> <p>Notes: Read only</p> <p>A globally unique ID for this element; that is, unique across all model files.</p>
ElementID	<p>Long</p> <p>Notes: Read only</p> <p>The local ID of the element; valid for this file only.</p>
Elements	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of child elements (sub-elements) attached to this element as seen in the tree view.</p>
EmbeddedElements	<p>Collection</p> <p>Notes: Read only</p> <p>A list of elements that are embedded into this element, such as Ports, Parts, Pins and Parameter Sets.</p>
EventFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags to do with signals or events.</p>
ExtensionPoints	<p>String</p> <p>Notes: Read/Write</p>

	Optional extension points for a Use Case as a comma-separated list.
Files	Collection Notes: Read only A collection of File objects.
FQName	String Notes: Read only The fully-qualified name of the element, consisting of a dot-separated list of names including all parent elements and Packages up to the first namespace root that is encountered.
FQStereotype	String Notes: Read only The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.
GenFile	String Notes: Read/Write The file associated with this element for code generation and synchronization purposes; can include macro expansion tags for local conversion to full path.
Genlinks	String Notes: Read/Write Links to other Classes discovered at code reversing time; Parents and Implements connectors only.
GenType	String Notes: Read/Write The code generation type; for example, Java, C++, C#, VBNet, Visual Basic, Delphi.
Header1	Variant Notes: Read/Write A user defined string for inclusion as header in the source files generated.
Header2	Variant Notes: Read/Write Same as for Header1, but used in the CPP source file.
IsActive	Boolean Notes: Read/Write Boolean value indicating whether the element is active or not. 1 = True, 0 = False.
IsComposite	Boolean Notes: Read/Write Indicates whether the element is composite or not. 1 = True, 0 = False.

IsLeaf	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates whether or not the element is a leaf node (and therefore cannot be a parent for any other elements).</p> <p>1 = True, 0 = False.</p>
IsNew	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Boolean value indicating whether the element is new or not.</p> <p>1 = True, 0 = False.</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates whether or not the element is a root node (and therefore cannot be descended from another element).</p> <p>1 = True, 0 = False.</p>
IsSpec	<p>Boolean</p> <p>Notes: Read/Write; Note that this attribute is no longer used in UML 2.0 and later releases, and is provided only to support models maintained in releases of UML prior to 2.0.</p> <p>Boolean value indicating whether the element is a specification or not.</p> <p>1 = True, 0 = False.</p>
Issues	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Issue objects.</p>
Locked	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates if the element has been locked against further change.</p>
MetaType	<p>String</p> <p>Notes: Read only</p> <p>The element's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.</p>
Methods	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Method objects for current element.</p>
MethodsEx	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Method objects belonging to the current element and its parent elements.</p>
Metrics	<p>Collection</p> <p>Notes: Read only</p>

	Collection of Metric elements for current element.
MiscData	<p>String</p> <p>Notes: Read only</p> <p>This low-level property provides information about the contents of the PData x fields.</p> <p>These database fields are not documented, and developers must gain understanding of these fields through their own endeavors to use this property.</p> <p>MiscData is zero based, therefore:</p> <ul style="list-style-type: none"> • MiscData(0) corresponds to PData1 • MiscData(1) to PData2, and so on
Modified	<p>Date</p> <p>Notes: Read/Write</p> <p>The date the element was last modified.</p>
Multiplicity	<p>String</p> <p>Notes: Read/Write</p> <p>Multiplicity value for this element.</p>
Name	<p>String</p> <p>Notes: Read/Write</p> <p>The element name; should be unique within the current Package.</p>
Notes	<p>String</p> <p>Notes: Read/Write</p> <p>Further descriptive text about the element.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
PackageID	<p>Long</p> <p>Notes: Read/Write</p> <p>A local ID for the Package containing this element.</p>
ParentID	<p>Long</p> <p>Notes: Read/Write</p> <p>If this element is a child of another, used to set or retrieve the ElementID of the other element; if not, returns 0.</p>
Partitions	<p>Collection</p> <p>Notes: Read only</p> <p>List of logical partitions into which an element can be divided.</p> <p>Only valid for elements that support partitions, such as Activities and States.</p>
Persistence	<p>String</p> <p>Notes: Read/Write</p>

	The persistence associated with this element; can be Persistent or Transient.
Phase	String Notes: Read/Write The phase this element is scheduled to be constructed in; any string value.
Priority	String Notes: 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.
Properties	Properties Notes: Returns a list of specialized properties that apply to the element that might not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them.
PropertyType	Long Notes: Read/Write The ElementID of a Type associated with this element; only valid for Port and Part elements.
PropertyTypeName	String Notes: Read The name of a Type associated with this element; only valid for Port and Part elements.
Realizes	Collection Notes: Read only List of Interfaces realized by this element for convenience.
Requirements	Collection Notes: Read only Collection of Requirement objects.
RequirementsEx	Collection Notes: Read only Collection of Requirement objects belonging to the current element and its parent elements.
Resources	Collection Notes: Read only Collection of Resource objects for current element.
Risks	Collection Notes: Read only Collection of Risk objects.

RunState	<p>String</p> <p>Notes: Read/Write</p> <p>The object's runstate list as a string.</p> <p>The string consists of a set of statements in the form: string = '@VAR;Variable=<string>;Value=<string>;Op=<string>;@ENDVAR;'</p> <p>Where: Op = ['=', '>', '<', '>=', '<=', '!=', '<>']</p> <p>For example: A set of run states can be created by looping through a set of attributes and forming a concatenated string: eRunState = eRunState + "@VAR;Variable="+ attrib.name + ";Value=" + attrib.value + ";Op==;@ENDVAR;";</p>
Scenarios	<p>Collection</p> <p>Notes: Read only</p> <p>Collection of Scenario objects for current element.</p>
StateTransitions	<p>Collection</p> <p>Notes: Read only</p> <p>List of State Transitions that an element can support; applies in particular to Timing elements.</p>
Status	<p>String</p> <p>Notes: Read/Write</p> <p>Sets or gets the status, such as Proposed or Approved.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>The primary element stereotype; the first of the list of stereotypes you can access using the 'StereotypeEx' attribute.</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the element in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
StyleEx	<p>String</p> <p>Notes: Read/Write</p> <p>Advanced style settings; reserved for the use of Sparx Systems.</p>
Subtype	<p>Long</p>

	<p>Notes: Read/Write</p> <p>A numeric subtype that qualifies the Type of the main element</p> <ul style="list-style-type: none"> • For Event: 0 = Receiver, 1 = Sender • For Class: 1 = Parameterised, 2 = Instantiated, 3 = Both, 0 = Neither, 17 = Association Class <p>If 17, because an Association Class has been created through the user interface, MiscData(3) contains the ID of the related Association; as MiscData is read-only, you cannot create an Association Class through the Automation Interface.</p> <ul style="list-style-type: none"> • For Note: 1 = Note linked to connector, 2 = Constraint linked to connector • For StateNode: 100 = ActivityInitial, 101 = ActivityFinal • For Activity: 0 = Activity, 8 = composite Activity (also set to 8 for other composite elements such as Use Cases) • For Synchronization: 0 = Horizontal, 1 = Vertical <p>Note that there are many more Types than indicated in these examples.</p>
Tablespace	<p>String</p> <p>Notes: Read/Write</p> <p>Associated tablespace for a Table element.</p>
Tag	<p>String</p> <p>Notes: Read/Write</p> <p>Corresponds to the 'Keywords' field in the Enterprise Architect user interface.</p>
TaggedValues	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of TaggedValue objects.</p>
TaggedValuesEx	<p>Collection</p> <p>Notes: Read only</p> <p>Returns a collection of TaggedValue objects belonging to the current element and the elements specialized or realized by the current element.</p>
TemplateParameters	<p>Collection</p> <p>Notes: Read Only</p> <p>A collection of TemplateParameter objects.</p>
Tests	<p>Collection</p> <p>Notes: Read only</p> <p>A collection of Test objects for the current element.</p>
TreePos	<p>Long</p> <p>Notes: Read/Write</p> <p>Sets or gets the tree position.</p>
Type	<p>String</p> <p>Notes: Read/Write</p> <p>The element type (such as Class, Component).</p> <p>Note that Type is case sensitive inside Enterprise Architect and should be provided</p>

with an initial capital (proper case); valid types are:

- Action
- Activity
- ActivityPartition
- ActivityRegion
- Actor
- Artifact
- Association
- Boundary
- Change
- Class
- Collaboration
- Component
- Constraint
- Decision
- DeploymentSpecification
- DiagramFrame
- EmbeddedElement
- Entity
- EntryPoint
- Event
- ExceptionHandler
- ExitPoint
- ExpansionNode
- ExpansionRegion
- Feature
- GUIElement
- InteractionFragment
- InteractionOccurrence
- InteractionState
- Interface
- InterruptibleActivityRegion
- Issue
- Node
- Note
- Object
- Package
- Parameter
- Part
- Port
- ProvidedInterface
- Report
- RequiredInterface
- Requirement

	<ul style="list-style-type: none"> • Screen • Sequence • State • StateNode • Synchronization • Text • TimeLine • UMLDiagram • UseCase
TypeInfoProperties	<p>Notes: Read only</p> <p>Returns an interface pointer of TypeInfoProperties.</p>
Version	<p>String</p> <p>Notes: Read/Write</p> <p>The version of the element.</p>
Visibility	<p>String</p> <p>Notes: Read/Write</p> <p>The Scope of this element within the current Package.</p> <p>Valid values are: Public, Private, Protected or Package.</p>

Element Methods

Method	Remarks
ApplyGroupLock(string aGroupName)	<p>Boolean</p> <p>Notes: Applies a group lock to the element object, for the specified group, on behalf of the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock()	<p>Boolean</p> <p>Notes: Applies a user lock to the element object for the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use 'GetLastError()' to retrieve error information.</p>
Clone ()	<p>LDISPATCH</p> <p>Notes: Inserts a copy of the selected element under the same parent as the selected element.</p> <p>Returns the newly-created element.</p>
CreateAssociationClass(lon	

g ConnectorID)	<p>Boolean</p> <p>Notes: Makes this element an AssociationClass of the Association with the provided Connector ID; the return value indicates whether the function succeeded in converting the element to an AssociationClass.</p> <p>AssociationClasses are created only where:</p> <ul style="list-style-type: none"> • The current element is valid • The current element is a Class • The current element is not already an AssociationClass • The specified connector exists • The specified connector is an Association • The specified connector is not already in an AssociationClass pair • The current element is not at either end of the specified connector <p>Parameters:</p> <ul style="list-style-type: none"> • ConnectorID: Long - the Connector ID of an Association connector
DeleteLinkedDocument()	<p>Boolean</p> <p>Notes: Removes the Linked Document for the element. This method does not display a confirmatory prompt.</p> <p>Returns True if a document was deleted.</p>
GetBusinessRules()	<p>String</p> <p>Notes: Read Only.</p> <p>Returns all the Business Rules for the element.</p>
GetChart	<p>LDISPATCH</p> <p>Notes: For chart elements returns an interface to the chart</p>
GetDecisionTable()	<p>String</p> <p>Notes: Provides read-only access to a Decision Table XML string.</p> <p>Returns the XML data for the Decision Table as a string.</p>
GetElementGrid()	<p>String</p> <p>Notes: Returns an object of type ElementGrid (a Custom Table Artifact element).</p>
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
GetLinkedDocument()	<p>String</p> <p>Notes: Returns a string value containing the element's Linked Document contents, in Rich Text Format.</p> <p>If the element contains no Linked Document, an empty string is returned.</p>
GetRelationSet(EnumRelationSetType Type)	<p>String</p> <p>Notes: Returns a string containing a comma-separated list of ElementIDs of directly- and indirectly-related elements based on the given type.</p> <p>Recurses using the same relation type on all elements it finds, retrieving all dependencies and sub-dependencies of the current element; for example, Object1 depends on Object2, which depends on Object3, therefore this method returns</p>

	<p>Object2 and Object3.</p> <p>To obtain only the direct relationships of the element, use the Connector collection instead.</p>
GetStereotypeList()	<p>String</p> <p>Notes: Returns a comma-separated list of stereotypes allied to this element.</p>
GetTXAlias (string Code, long Flag)	<p>String</p> <p>Notes: Returns the Alias of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	<p>String</p> <p>Notes: Returns the name of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	<p>String</p> <p>Returns the Notes of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
HasStereotype(string Stereotype)	<p>Boolean</p> <p>Notes: Returns true if the current element has the specified stereotype applied to it. Accepts either qualified or unqualified stereotype names; for example, 'block' or 'SysML1.3::block'.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Stereotype: String - the name of the stereotype to search for
IsAssociationClass	<p>Boolean</p> <p>Notes: Returns whether or not the current element is an AssociationClass.</p>
LoadLinkedDocument(stri	<p>Boolean</p>

ng Filename)	<p>Notes: Loads the document from the specified file into the element's Linked Document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the file from which to load the document; both RTF and DOCX input formats are supported
Refresh()	<p>Void</p> <p>Notes: Refreshes the element features in the Browser window.</p> <p>Usually called after adding or deleting attributes or methods, when the user interface is required to be updated as well.</p>
ReleaseUserLock()	<p>Boolean</p> <p>Notes: Releases a user lock or group lock on the element object.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
SaveLinkedDocument(string Filename)	<p>Boolean</p> <p>Notes: Saves the Linked Document for this element to the specified file. Returns False if the element does not have a Linked document or fails to save the file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the file to save to disk The output format will be determined by the file's extension - currently rtf, docx and pdf are supported; if an invalid extension is used, it will write the file in RTF format regardless of the extension
SetAppearance(long Scope, long Item, long Value)	<p>Void</p> <p>Notes: Sets the visual appearance of the element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Scope: Long - Scope of appearance set to modify 1 - Base (Default appearance across entire model) To set appearance for the element (diagram object) in a selected diagram only, see <i>Setting The Style</i> in the <i>DiagramObject Class</i> topic • Item: Long - Appearance feature to modify 0 - Background color 1 - Font Color 2 - Border Color 3 - Border Width • Value: Long - Value to set appearance to
SetCompositeDiagram()	<p>Boolean</p> <p>Notes: Sets the composite diagram of the element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GUID: String - the GUID of the composite diagram; a blank GUID will remove the link to the composite diagram
SetCreated(Date NewVal)	<p>Void</p> <p>Notes: Deprecated</p> <p>This method is no longer supported.</p>
SetModified(Date NewVal)	<p>Void</p>

	<p>Notes: Deprecated</p> <p>This method is no longer supported.</p>
SetTXAlias (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Alias of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated name of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Notes of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
SynchConstraints(string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes the constraints of a UML Profile item for this element, only if the specified stereotype has been applied.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Profile: String - Name of the profile that contains the stereotype • Stereotype: String - Name of the profile stereotype for which the default constraints are to be synchronized
SynchTaggedValues(string Profile, string Stereotype)	<p>Boolean</p> <p>Notes: Synchronizes the Tagged Values of a UML Profile item for this element, only if the specified stereotype has been applied.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Profile: String - Name of the profile that contains the stereotype • Stereotype: String - Name of the profile stereotype for which the default tags are to be synchronized
UnlinkFromAssociation	<p>Boolean</p> <p>Notes: Performs the opposite of CreateAssociationClass().</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current element object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Classe ElementGrid

The ElementGrid object represents a Custom Table, which is used to display custom data in tabular format on a diagram, the data being provided by the user rather than generated by the system.

The ElementGrid object is accessible from an Element object, using the GetElementGrid() method.

Associated table in repository

t_object

ElementGrid Methods

Method	Remarks
GetCell (int nrow, int ncell)	Variant Notes: The cell value is return as a variant value. Parameters: <ul style="list-style-type: none"> • nRow: Integer - the number of the row containing the cell • nCell: Integer - the number of the cell in the row (the column number)
GetColumnCount ()	Integer Notes: Returns the number of columns in the grid.
GetRowCount ()	Integer Notes: Returns the number of rows in the grid.
SetCell (int nRow, int nCell, variant sValue)	Boolean Notes: Sets a value in the specified cell. Parameters: <ul style="list-style-type: none"> • nRow: Integer - specifies the row into which to insert the value • nCell: Integer - specifies the cell (column number) into which to insert the value • sValue: Variant - specifies the value to set in the cell
SetGridSize (int nRows, int nColumns)	Boolean Notes: Sets the size of the grid in rows and columns. The size can be set and reset; any data outside the bounds of the new grid size will be lost on resize. Parameters: <ul style="list-style-type: none"> • nRows: Integer - the number of rows in the table grid • nColumns: Integer - the number of columns in the table grid

File Class

A File represents an associated file for an element. Files are accessed through the Element Files collection.

Associated table in repository

t_objectfiles

File Attributes

Attribute	Remarks
FileDate	String Notes: Read/Write The file date when the entry was created.
Name	String Notes: Read/Write The file name can be a logical file or a reference to a web address (using http://).
Notes	String Notes: Read/Write Notes about the file.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Size	String Notes: Read/Write The file size.
Type	String Notes: Read/Write The file type.

File Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in

	relation to this object.
Update()	Boolean Notes: Updates the current File object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Issue (Maintenance) Class

An Issue is either a Change or a Defect, is associated with the containing element, and is accessed through the Issues collection of an element.

Associated table in repository

t_objectproblems

Issue Attributes

Attribute	Remarks
DateReported	Date Notes: Read/Write The date the issue was reported.
DateResolved	Date Notes: Read/Write The date the issue was resolved.
ElementID	Long Notes: Read/Write The ID of the element associated with this issue.
Name	String Notes: Read/Write The Issue name; that is, the Issue itself.
Notes	String Notes: Read/Write The Issue description.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Priority	String Notes: Read/Write The priority of the Issue - Low, Medium or High.
Reporter	String Notes: Read/Write The user ID of the person reporting the issue.

Resolver	String Notes: Read/Write The user ID of the person resolving the issue.
ResolverNotes	String Notes: Read/Write Notes entered by the resolver about resolution of the Issue.
Severity	String Notes: Read/Write The Issue severity - Low, Medium or High.
Status	String Notes: Read/Write The current status of the issue.
Type	Variant Notes: Read/Write The Issue type - Defect, Change, Issue or Task.
Version	String Notes: Read/Write The version associated with the issue. Note that this method is only available through a Dispatch interface. Object ob = Issue; Print ob.Version;

Issue Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Issue object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Metric Class

A Metric is a named item with a weighting that can be associated with an element for purposes of building metrics about the model. Metrics are accessed through the Element Metrics collection.

Associated table in repository

t_objectmetrics

Metric Attributes

Attribute	Remarks
Name	String Notes: Read/Write The name of the metric.
Notes	String Notes: Read/Write Notes about this metric.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The metric type.
Weight	Long Notes: Read/Write A user-defined weighting for estimation or metric purposes.

Metric Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Metric object after modification or appending a new

	<p>item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
--	-----------------------------------------------------------------------------------------------------

Requirement Class

An Element Requirement object holds information about the requirements of an element in the context of the model. Requirements can be accessed using the Element Requirements collection.

Associated table in repository

t_objectrequires

Requirement Attributes

Attribute	Remarks
Difficulty	String Notes: Read/Write The estimated difficulty of implementing the requirement.
LastUpdate	Date Notes: Read/Write The date the requirement was last updated.
Name	String Notes: Read/Write The requirement itself.
Notes	String Notes: Read/Write Further notes on the requirement.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only The ElementID of the element to which this requirement applies.
Priority	String Notes: Read/Write The assigned priority of the requirement.
RequirementID	Long Notes: Read only A local ID for this requirement.

Stability	String Notes: Read/Write The estimated stability of the requirement. This is an indication of the probability of the requirement - or understanding of the requirement - changing. High stability indicates a low probability of the requirement changing.
Status	String Notes: Read/Write The current status of the requirement.
Type	String Notes: Read/Write The requirement type.

Requirement Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current Requirement object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Resource 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 repository

t_objectresources

Resource Attributes

Attribute	Description
ActualHours	Long Notes: Read/Write The time already expended on the task, in hours, days or other units.
DateEnd	Date Notes: Read/Write The expected end date.
DateStart	Date Notes: Read/Write The date to start work.
ExpectedHours	Long Notes: Read/Write The total expected time the task might run, in hours, days or other units.
History	String Notes: Read/Write Gets or sets history text.
Name	String Notes: Read/Write The name of the resource (for example, a person's name).
Notes	String Notes: Read/Write Descriptive notes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

PercentComplete	Long Notes: Read/Write The current percent complete figure.
Role	String Notes: Read/Write The role the resource plays in implementing the element.
Time	Long Notes: Read/Write The time expected to complete the task; a numeric indicating the number of days.

Resource Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.
Update()	Boolean Notes: Update the current Resource object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Risk Class

A Risk object represents a named risk associated with an element. It is used for project management purposes. Risks can be accessed through the Element Risks collection.

Associated table in repository

t_objectrisks

Risk Attributes

Attribute	Description
Name	String Notes: Read/Write The name of the risk.
Notes	String Notes: Read/Write Further notes describing the risk.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The risk type associated with this element.
Weight	Long Notes: Read/Write A weighting for estimation or metric purposes.

Risk Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean

	<p>Notes: Update the current Risk object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.</p>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Scenario Class

A Scenario corresponds to a Collaboration or Use Case instance. Each Scenario is a path of execution through the logic of a Use Case. Scenarios can be added to using the Element Scenarios collection.

Associated table in repository

t_objectscenarios

Scenario Attributes

Attribute	Description
Name	String Notes: Read/Write The Scenario name.
Notes	String Notes: Read/Write A description of the Scenario, usually containing the steps to execute the scenario.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ScenarioGUID	String Notes: Read/Write A unique ID for the Scenario, used to identify the Scenario unambiguously within a model.
Steps	Collection of ScenarioStep Class Notes: Read only A collection of step objects for this Scenario. Use the 'AddNew' and 'Delete' functions to manage steps. 'AddNew' passes the step name and 'I' as the type for an actor step.
Type	String Notes: Read/Write The scenario type (for example, Basic Path).
Weight	Long Notes: Read/Write Currently used to position scenarios in the scenario list (that is, List Position).
XMLContent	String

	<p>Notes: Read/Write</p> <p>A structured field that can contain scenario details in XML format. It is recommended that you use the 'Steps' collection to read or modify this field.</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Scenario Methods

Method	Description
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
Update()	<p>Boolean</p> <p>Notes: Update the current Scenario object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

ScenarioExtension Class

ScenarioExtension Attributes

Attribute	Description
ExtensionGUID	String Notes: Read/Write A unique GUID for this Extension.
Join	String Notes: Read/Write The GUID of the step where this Extension rejoins the Scenario.
JoiningStep	ScenarioStep Notes: Read only The actual step where this Extension rejoins the Scenario, if any.
Level	String Notes: Read only The number of this Extension as shown in the scenario editor. This is derived from the value of Pos for this object and the owning step.
Name	String Notes: Read/Write The Extension name. This should match the name of the linked scenario.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Pos	Long Notes: Read/Write The position of the Extension in the Extensions list.
Scenario	Scenario Notes: Read only The scenario that is executed as an alternative path for this Extension.

ScenarioStep Classe

ScenarioStep Attributes

Attribute	Description
Extensions	Collection of ScenarioExtension Notes: Read only A collection of ScenarioExtension objects that specify how the scenario is extended from this step. The arguments to 'AddNew' should match the name and GUID of the alternative scenario being linked to.
Level	String Notes: Read only The number of this Step as shown in the scenario editor. This is derived from the value of Pos.
Link	String Notes: Read/Write The GUID of a Use Case that is relevant to this step.
LinkedElement	Element Notes: Read only The actual element specified by Link, if any.
Name	String Notes: Read/Write The step name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Pos	Long Notes: Read/Write The position of the 'Step' in the 'Scenario Step' list.
Results	String Notes: Read/Write Any results that are given from this step.
State	String Notes: Read/Write A description of the state the system enters when this Step is executed.
StepGUID	String

	Notes: Read/Write A unique GUID for this Step.
StepType	ScenarioStepType Notes: Read/Write Identifies whether this step is being performed by a user or the system.
Uses	String Notes: Read/Write The input and requirements that are relevant to this step.
UsesElementList	Collection of Element Notes: Read only Indicates that the Scenarios view 'Uses' field is a linked element list.

ScenarioStep Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current ScenarioStep object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Méthodes ScenarioExtension

Méthode	Description
Obtenir la dernière erreur ()	String Notes : renvoie une valeur string décrivant l'erreur la plus récente survenue par rapport à cet object .
Mise à jour()	Booléen Notes : met à jour l' object ScenarioExtension actuel après modification ou ajout d'un nouvel élément. Si False est renvoyé, vérifiez la fonction « GetLastError() » pour plus d'informations.

TaggedValue Class

A TaggedValue is a named property and value associated with an element. Tagged Values can be accessed through the TaggedValues collection.

Associated table in repository

t_objectproperties

TaggedValue Attributes

Attribute	Description
ElementID	Long Notes: Read/Write The local ID of the associated element.
FQName	String Notes: Read only The fully-qualified name of the tag.
Name	String Notes: Read/Write The name of the tag.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write The global ID of the tag.
PropertyID	Long Notes: Read only The local ID of the tag.
Value	String Notes: Read/Write

	<p>The value assigned to this tag.</p> <p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

TaggedValue Methods

Method	Description
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p> <p>Parameters:</p> <ul style="list-style-type: none"> propName: String - the name of the property for which the text is being returned
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p> <p>Parameters:</p> <ul style="list-style-type: none"> propName: String - the name of the property for which the text is being set propValue: the value of the property
Update()	<p>Boolean</p> <p>Notes: Updates the current TaggedValue object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Test 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 repository

t_objecttests

Test Attributes

Attribute	Description
AcceptanceCriteria	String Notes: Read/Write The acceptance criteria for successful execution.
CheckedBy	String Notes: Read/Write User ID of the person confirming the results.
Class	Long Notes: Read/Write The test Class: 1 = Unit Test 2 = Integration Test 3 = System Test 4 = Acceptance Test 5 = Scenario Test 6 = Inspection Test
DateRun	Date Notes: Read/Write The date the test was last run.
Input	String Notes: Read/Write Input data for the test.
Name	String Notes: Read/Write The test name.
Notes	String Notes: Read/Write

	Detailed notes about test to be carried out.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
RunBy	String Notes: Read/Write The user ID of the person conducting the test.
Status	String Notes: Read/Write The current status of the test.
TestResults	Variant Notes: Read/Write Results of test.
Type	String Notes: Read/Write The test type, such as Load or Regression.

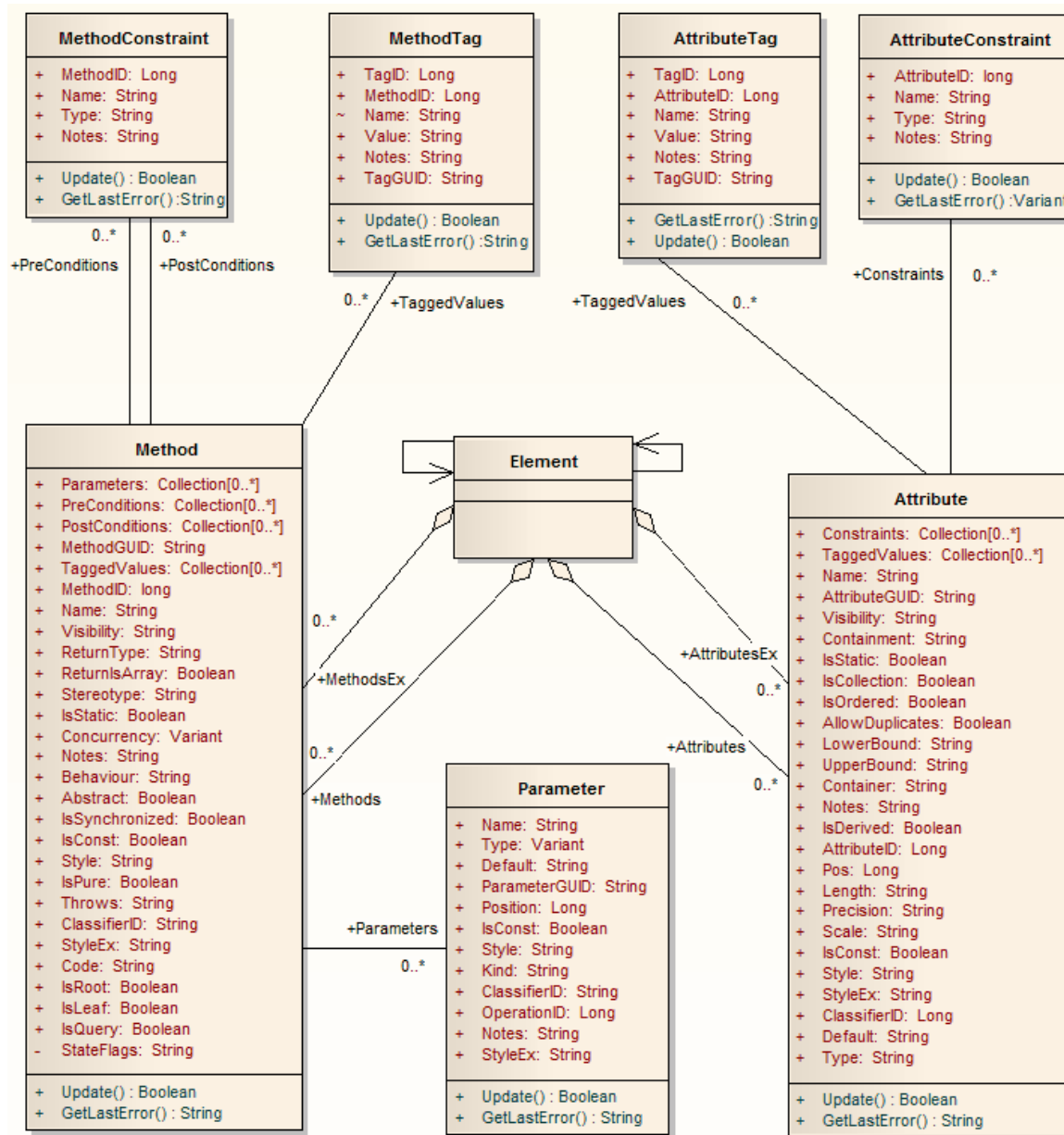
Test Methods

Method	Description
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Test object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Element Features Package

The ElementFeatures Package contains descriptions of the model interfaces that enable access to operations and attributes, and their associated Tagged Values and constraints.

This diagram illustrates the components associated with element features. These include attributes and methods, and their associated constraints and Tagged Values. It also includes the Parameter object that defines the arguments associated with an operation (Method).



Attribute 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 repository

t_attribute

Attribute Attributes

Attribute	Remarks
Alias	String Notes: Read/Write Contains the (optional) 'Alias' property for this attribute. This can be used interchangeably with the Style attribute.
AllowDuplicates	Boolean Notes: Read/Write Indicates if duplicates are allowed in the collection. If the attribute represents a database column this, when set, represents the 'Not Null' option.
AttributeGUID	String Notes: Read only A globally unique ID for the current attribute. This attribute is system generated.
AttributeID	Long Notes: Read only The local ID number of the attribute.
ClassifierID	Long Notes: Read/Write The classifier ID, if appropriate, indicating the base type associated with the attribute, if not a primitive type.
Constraints	Collection Notes: Read only A collection of AttributeConstraint objects, used to access and manage constraints associated with this attribute.
Container	String Notes: Read/Write The container type.

Containment	String Notes: Read/Write The type of containment - Not Specified, By Reference or By Value.
Default	String Notes: Read/Write The initial value assigned to this attribute.
FQStereotype	String Notes: Read Only The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.
IsCollection	Boolean Notes: Read/Write Indicates if the current feature is a collection or not. If the attribute represents a database column this, when set, represents a Foreign Key.
IsConst	Boolean Notes: Read/Write A flag indicating if the attribute is Const or not.
IsDerived	Boolean Notes: Read/Write Indicates if the attribute is derived (that is, a calculated value).
IsID	Boolean Notes: Read/Write Indicates if the attribute uniquely identifies an instance of the containing Class, or not.
IsOrdered	Boolean Notes: Read/Write Indicates if a collection is ordered or not. If the attribute represents a database column this, when set, represents a Primary Key.
IsStatic	Boolean Notes: Read/Write Indicates if the current attribute is a static feature or not. If the attribute represents a database column this, when set, represents the 'Unique' option.
Length	String Notes: Read/Write The attribute length, where applicable.
LowerBound	String Notes: Read/Write A value for the collection lower boundary.

Name	String Notes: Read/Write The attribute name.
Notes	String Notes: Read/Write Further notes on this attribute.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ParentID	Long Notes: Read only Returns the ElementID of the element that this attribute is a part of.
Pos	Long Notes: Read/Write The position of the attribute in the Class attribute list.
Precision	String Notes: Read/Write The precision value.
RedefinedProperty	String Notes: Read/Write Corresponds to the 'Redefined Property' field on the 'Detail' page of the attribute 'Properties' dialog, or the UML <i>redefinedProperty</i> attribute. Contains a comma separated list of GUIDs.
Scale	String Notes: Read/Write The scale value.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this attribute. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.
StereotypeEx	String Notes: Read/Write Provides all the applied stereotypes of the attribute, in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names. When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.

Style	String Notes: Read/Write Contains the (optional) Alias property for this attribute. This can be used interchangeably with the Alias attribute.
StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
SubsettedProperty	String Notes: Read/Write Corresponds to the 'Subsetted Property' field on the 'Detail' page of the attribute 'Properties' dialog, or the UML <i>subsettedProperty</i> attribute. Contains a comma separated list of GUIDs.
TaggedValues	Collection of type AttributeTag Notes: Read only A collection of AttributeTag objects, used to access and manage Tagged Values associated with this attribute.
TaggedValuesEx	Collection of type TaggedValue Notes: Read only A collection of TaggedValue objects belonging to the current attribute and the TaggedValuesEx property of its classifier.
Type	String Notes: Read/Write The attribute type (by name; also see <i>ClassifierID</i>).
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
UpperBound	String Notes: Read/Write A value for the collection upper boundary.
Visibility	String Notes: Read/Write Identifies the scope of the attribute - Private, Protected, Public or Package.

Attribute Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in

	relation to this object.
GetTXAlias (string Code, long Flag)	<p>String</p> <p>Notes: Returns the Alias of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	<p>String</p> <p>Notes: Returns the name of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	<p>String</p> <p>Returns the Notes of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXAlias (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Alias of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated name of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Notes of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog)

	<ul style="list-style-type: none">• Translation: String - The translated Notes
Update()	<p>Boolean</p> <p>Notes: Updates the current attribute object after modifying or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

AttributeConstraint Class

An AttributeConstraint is a constraint associated with the current Attribute.

Associated table in repository

t_attributeconstraints

AttributeConstraint Attributes

Attribute	Remarks
AttributeID	Long Notes: Read/Write The ID of the attribute this constraint applies to.
Name	String Notes: Read/Write The name of the constraint.
Notes	String Notes: Read/Write Descriptive notes about the constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The type of constraint.

AttributeConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current AttributeConstraint object after modification or

	<p>appending a new item.</p>
--	------------------------------

If False is returned, check the 'GetLastError()' function for more information.

AttributeTag Class

An AttributeTag represents a Tagged Value associated with an attribute.

Associated table in repository

t_attributetag

AttributeTag Attributes:

Attribute	Remarks
AttributeID	Long Notes: Read/Write The local ID of the attribute associated with this Tagged Value.
FQName	String Notes: Read only The fully-qualified name of the tag.
Name	String Notes: Read/Write The name of the tag.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A globally unique ID for this Tagged Value.
TagID	Long Notes: Read only The local ID to identify the Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

AttributeTag Methods:

Method	Remarks
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p>
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p> <p>This function is rarely used as an exception is thrown when an error occurs.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current AttributeTag object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

CustomProperties Collection

The CustomProperties collection contains 0 or more CustomProperties associated with the current element. These properties provide advanced UML configuration options, and must not be added to or deleted. The value of each property can be set.

CustomProperty

Attribute	Remarks
Name	String Notes: Read only The CustomProperty name.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Value	String Notes: Read/Write The value associated with this CustomProperty. This can be: <ul style="list-style-type: none">• A string• The Boolean values True or False, or• An enumeration value from a defined list The UML 2.5 specification in general provides information on the kinds of enumeration relevant here.

Notes

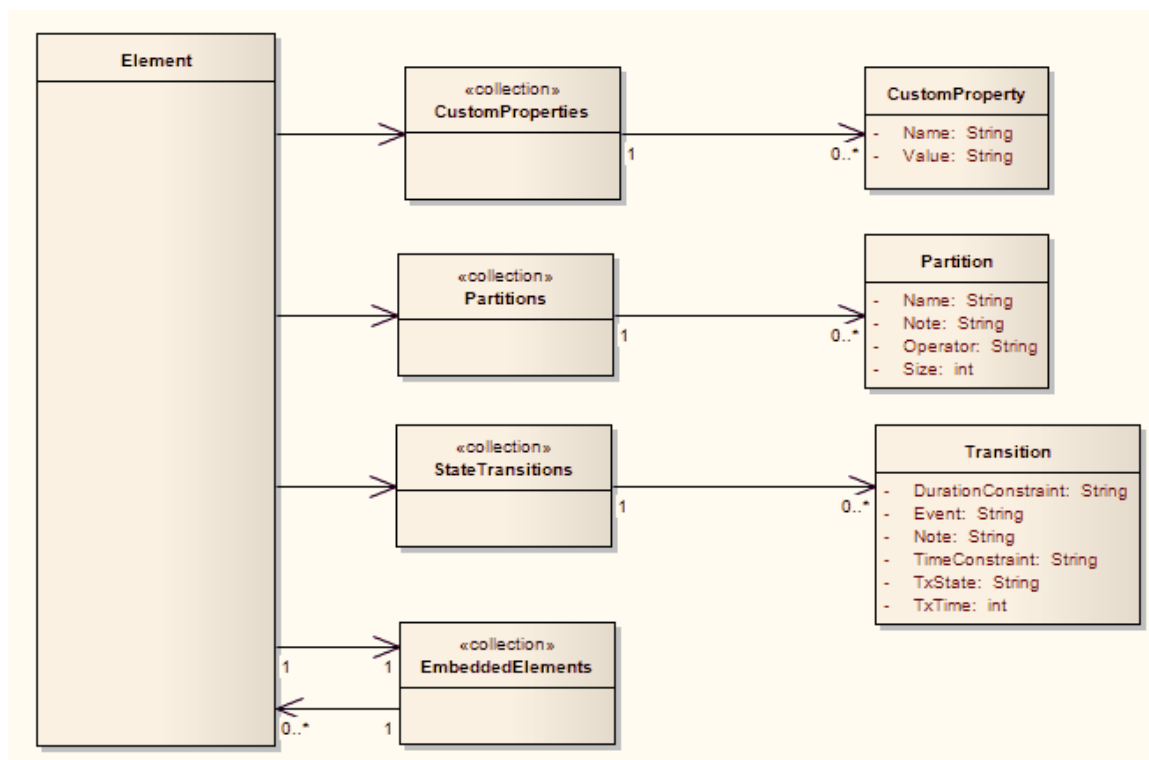
- The number and type of properties vary depending on the actual element

EmbeddedElements Collection

In UML 2.5 an element can have one or more embedded elements such as Ports, Pins, Parameters or ObjectNodes. These are attached to the boundary of the host element and cannot be moved off the element. They are owned by their host element. This collection gives easy access to the set of elements embedded on the surface of an element. Note that some embedded elements can have their own embedded element collection (for example, Ports can have Interfaces embedded on them).

The EmbeddedElements collection contains Element objects.

Example



Method 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 repository

t_operation

Method Attributes

Attribute	Remarks
Abstract	Boolean Notes: Read/Write A flag indicating if the method is abstract (1) or not (0).
Behavior	String Notes: Read/Write Some further explanatory behavior notes (for example, pseudocode). In earlier releases of Enterprise Architect this attribute had the UK/Australian spelling 'Behaviour'; this is still present for backwards compatibility, but please now use the 'Behavior' attribute for consistency.
ClassifierID	String Notes: Read/Write The Classifier ID that applies to the ReturnType.
Code	String Notes: Read/Write An optional field to hold the method code (used for the 'Initial Code' field).
Concurrency	Variant Notes: Read/Write Indicates the concurrency type of the method.
FQStereotype	String Notes: Read Only The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.
IsConst	Boolean Notes: Read/Write A flag indicating that the method is Const.

IsLeaf	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate if the method is a Leaf (cannot be overridden).</p>
IsPure	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the method is defined as 'Pure' in C++.</p>
IsQuery	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate if the method is a query (that is, does not alter Class variables).</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate if the method is Root.</p>
IsStatic	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate a static method.</p>
IsSynchronized	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating a Synchronized method call.</p>
MethodGUID	<p>String</p> <p>Notes: Read/Write</p> <p>A globally unique ID for the current method. This is system generated.</p>
MethodID	<p>Long</p> <p>Notes: Read only</p> <p>A local ID for the current method, only valid within this .eap file.</p>
Name	<p>String</p> <p>Notes: Read/Write</p> <p>The method name.</p>
Notes	<p>String</p> <p>Notes: Read/Write</p> <p>Descriptive notes on the method.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Parameters	<p>Collection Class</p> <p>Notes: Read only</p> <p>The Parameters collection for the current method, used to add and access parameter objects for the current method.</p>

ParentID	<p>Long</p> <p>Notes: Read only</p> <p>Returns the ElementID of the element that this method belongs to.</p>
Pos	<p>Long</p> <p>Notes: Read/Write</p> <p>Specifies the position of the method within the set of operations defined for a Class.</p>
PostConditions	<p>Collection Class</p> <p>Notes: Read only</p> <p>The PostConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'post'.</p>
PreConditions	<p>Collection Class</p> <p>Notes: Read only</p> <p>The PreConditions (constraints) as they apply to this method. This returns a MethodConstraint object of type 'pre'.</p>
ReturnIsArray	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag to indicate that the return value is an array.</p>
ReturnType	<p>String</p> <p>Notes: Read/Write</p> <p>The return type for the method; this can be a primitive data type or a Class or Interface type.</p>
StateFlags	<p>String</p> <p>Notes: Read/Write</p> <p>Some flags as applied to methods in State elements.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>The method stereotype (optional).</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the method in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
Style	<p>String</p> <p>Notes: Read/Write</p> <p>Contains the Alias property for this method.</p>

StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
TaggedValues	Collection Class of type MethodTag Class Notes: Read only The TaggedValues collection for the current method. This accesses a list of MethodTag objects.
Throws	String Notes: Read/Write Exception information. Valid input for setting the Throws is: <ul style="list-style-type: none"> • GUID String - the GUID of an element in the model or a comma-separated list of element GUIDS • <none> - removes the existing Throws set
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
Visibility	String Notes: Read/Write The method scope - Public, Protected, Private or Package.

Method Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed - 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of

	<p>the 'Manage Model Options' dialog)</p> <ul style="list-style-type: none"> • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	<p>String</p> <p>Returns the Notes of the element for a given language.</p> <p>Parameters</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
SetTXName (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated name of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXAlias (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Alias of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXNote (string Code, string Translation)	<p>String</p> <p>Notes - Set the translated Notes of the element for a given language.</p> <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
Update()	<p>Boolean</p> <p>Notes: Update the current method object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

MethodConstraint Class

A MethodConstraint is a condition imposed on a method. It is accessed through either the Method PreConditions or Method PostConditions collection.

Associated table in repository

t_operationpres and t_operationposts

MethodConstraint Attributes

Attribute	Remarks
MethodID	Long Notes: Read/Write The local ID of the associated method.
Name	String Notes: Read/Write The name of the constraint.
Notes	String Notes: Read/Write Descriptive notes about this constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The constraint type.

MethodConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.

Update()	<p>Boolean</p> <p>Notes: Update the current MethodConstraint object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

MethodTag Class

A MethodTag is a Tagged Value associated with a method.

Associated table in repository

t_operationtag

MethodTag Attributes:

Attribute	Remarks
FQName	String Notes: Read only The fully-qualified name of the tag.
MethodID	Long Notes: Read/Write The ID of the associated method.
Name	String Notes: Read/Write The tag or name of the property.
Notes	String Notes: Read/Write Further descriptive notes about this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A unique GUID for this Tagged Value.
TagID	Long Notes: Read only A unique ID for this Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

MethodTag Methods:

Method	Remarks
GetAttribute(string propName)	<p>String</p> <p>Notes: Returns the text of a single named property within a structured Tagged Value.</p>
GetLastError()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p> <p>This function is rarely used as an exception is thrown when an error occurs.</p>
HasAttributes()	<p>Boolean</p> <p>Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.</p>
SetAttribute(string propName, string propValue)	<p>Boolean</p> <p>Notes: Sets the text of a single named property within a structured Tagged Value.</p>
Update()	<p>Boolean</p> <p>Notes: Updates the current MethodTag object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>

Parameter Class

A Parameter object represents a method argument and is accessed through the Method Parameters collection.

Associated table in repository

t_operationparams

Parameter Attributes

Attribute	Remarks
Alias	String Notes: Read/Write An optional alias for this parameter.
ClassifierID	String Notes: Read/Write A ClassifierID for the parameter, if known.
Default	String Notes: Read/Write A default value for this parameter.
IsConst	Boolean Notes: Read/Write A flag indicating that the parameter is Const (cannot be altered).
Kind	String Notes: Read/Write The parameter kind - in, inout, out, or return.
Name	String Notes: Read/Write The parameter name; this must be unique for a single method.
Notes	String Notes: Read/Write Descriptive notes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
OperationID	Long

	<p>Notes: Read only</p> <p>The ID of the method associated with this parameter.</p>
ParameterGUID	<p>String</p> <p>Notes: Read/Write</p> <p>A system generated, globally unique ID for the current Parameter.</p>
Position	<p>Long</p> <p>Notes: Read/Write</p> <p>The position of the parameter in the argument list.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>The first stereotype of the parameter.</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the parameter in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p> <p>When setting this attribute, LastError (for the GetLastError method) will be non-empty if an error occurs.</p>
Style	<p>String</p> <p>Notes: Read/Write</p> <p>Some style information.</p>
StyleEx	<p>String</p> <p>Notes: Read/Write</p> <p>Advanced style settings, reserved for the use of Sparx Systems.</p>
TaggedValues	<p>Collection Class of type ParamTag Class</p> <p>Notes: Read/Write</p> <p>The GUID of the parameter with which this ParamTag is associated.</p>
Type	<p>Variant</p> <p>Notes: Read/Write</p> <p>The parameter type; can be a primitive type or a defined classifier.</p>
TypeInfoProperties	<p>Notes: Read only</p> <p>Returns an interface pointer of TypeInfoProperties.</p>

Parameter Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current Parameter object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ParamTag Class

A ParamTag is a Tagged Value associated with a method parameter.

Associated table in repository

t_taggedvalue

ParamTag Attributes

Attribute	Remarks
ElementGUID	String Notes: Read/Write The GUID of the parameter with which this ParamTag is associated.
FQName	String Notes: Read only The fully qualified name of the tag.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write A system generated GUID to identify the Tagged Value.
Tag	String Notes: Read/Write The actual tag name.
Value	String Notes: Read/Write The value associated with this tag.

ParamTag Methods

Method	Remarks
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a structured Tagged

	Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a structured Tagged Value.
Update()	Boolean Notes: Updates the current ParamTag object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Partitions Collection

A collection of internal element partitions (regions). This is commonly seen in Activity, State, Boundary, Diagram Frame and similar elements. Not all elements support partitions.

This collection contains a set of Partition elements. The set is read/write: information is not saved until the host element is saved, so ensure that you call the Element.Save method after making changes to a Partition.

Partition Attributes

Attribute	Remarks
Name	String Notes: Read/Write The partition name; this can represent a condition or constraint in some cases.
Note	String Notes: Read/Write A free text note associated with this partition.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Operator	String Notes: Read/Write An optional operator value that specifies the partition type.
Size	String Notes: Read/Write The vertical or horizontal width of the partition in pixels.

Properties Class

Properties

Properties Attributes

Attribute	Remarks
Count	Long Notes: The number of properties that are available for this object.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Properties Methods

Property

Method	Remarks
Item(object Index)	Property Notes: Returns a property either by name or by a zero-based integer offset into the list of properties. Parameter: <ul style="list-style-type: none"> Index: Variant - either a string representing the property name or an integer representing the zero-based offset into the property list

Property Attributes

Attribute	Remarks
Name	String Notes: Read only The name of the property. The object to which the properties list applies can have an automation property with the same name, in which case the data accessed through Value is identical to that obtained through the automation property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Type	<p>PropType</p> <p>Notes: Read only</p> <p>Provides an indication of what sort of data is going to be stored by this property. This restriction can be further defined by the Validation attribute.</p>
Validation	<p>String</p> <p>Notes: Read only</p> <p>An optional string that is used to validate any data that is passed to the Value attribute. This string is used by the programmer at run time to provide an indication of what is expected, and by Enterprise Architect to ensure that the submitted data is appropriate.</p>
Value	<p>Variant</p> <p>Notes: Read/write</p> <p>The value of the property as defined in the other fields.</p>

TemplateParameter Class

A TemplateParameter for a template signature specifies a formal parameter that will be substituted by an actual parameter (or the default) in a TemplateBinding relationship on a Class element.

Associated table in repository

t_xref

TemplateParameter Attributes

Attribute	Remarks
Constraint	String Notes: Read/Write The name of the Classifier that acts as the constraint value.
Default	String Notes: Read/Write The name of the Classifier that acts as the default value.
Name	String Notes: Read/Write The name of the Template Parameter.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a Dispatch interface.
TemplateParameterID	String Notes: Read Only The Enterprise Architect Globally Unique ID (GUID) of the current Template Parameter, in the XrefID column of t_xref.
Type	String Notes: Read/Write The Template Parameter type.

TemplateParameter Methods

Method	Remarks

GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Updates the current TemplateParameter object after modifying or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Transitions Collection

The Transitions collection applies only to Timeline elements.

A Timeline element displays 0 or more state transitions at set times on its extent. This collection enables you to access the transition set. You can also access additional information by referring to the connectors associated with the Timeline, and by referencing messages passed between timelines. Note that any changes made to elements in this collection are only saved when the main element is saved.

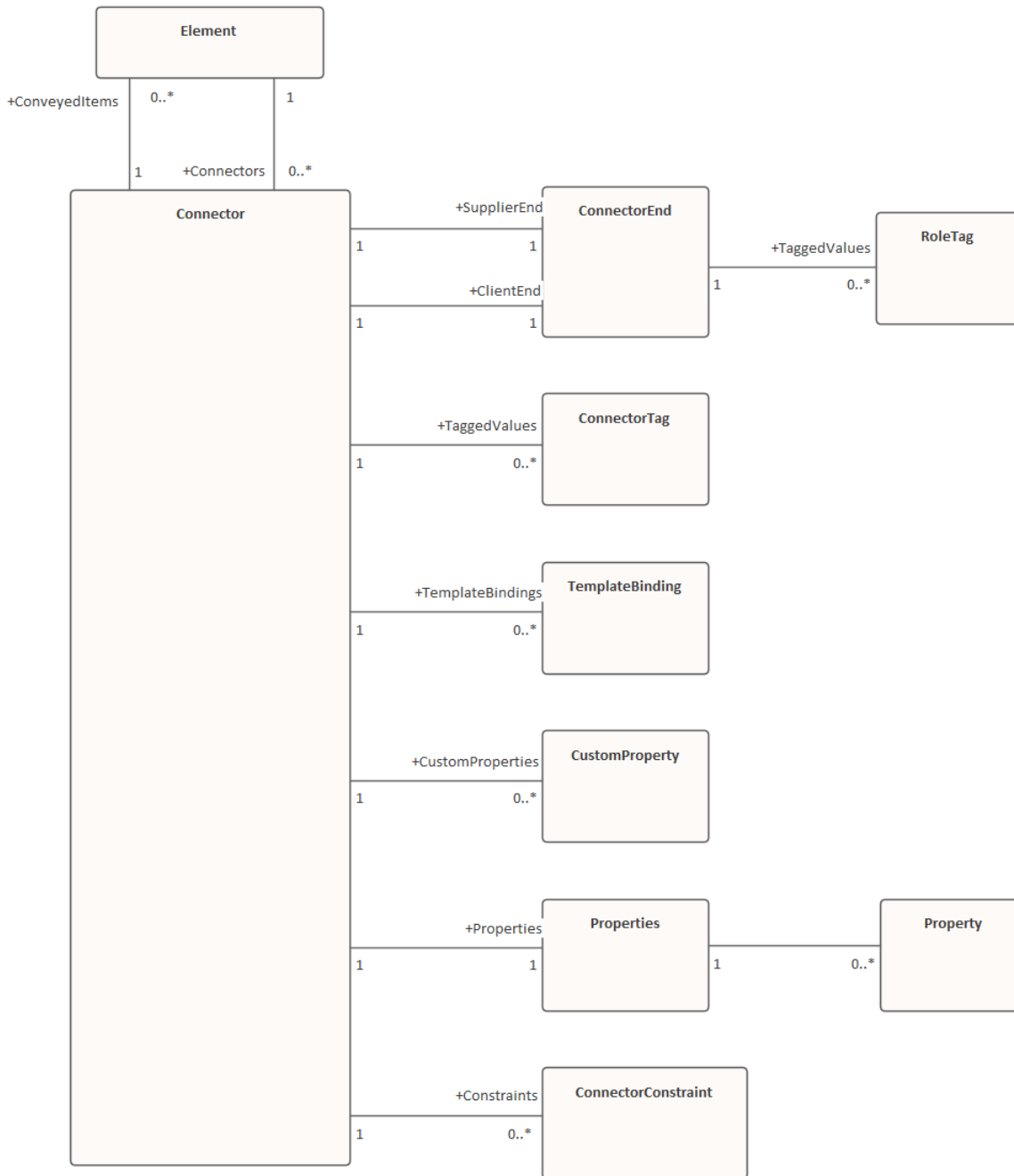
Transition Attributes

Attribute	Remarks
DurationConstraint	String Notes: Read/Write A constraint on the time duration of the transition.
Event	String Notes: Read/Write The event (optional) that initiated the transition.
Note	String Notes: Read/Write A free text note.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TimeConstraint	String Notes: Read/Write A constraint on when the transition has to be completed.
TxState	String Notes: Read/Write The state to transition to, as defined in the 'Timeline Properties' dialog.
TxTime	String Notes: Read/Write. The time that the transition occurs. The value depends on a range set in the diagram.

Connector Package

The Connector Package details how connectors between elements are accessed and managed.

This diagram shows the Connector Class, its collections, and its relationships to the Element Class. Association Target roles correspond to member variable names in the source interface. The associated Classes represent the object type used in each collection.



Connector Class

To represent the various kinds of connectors between UML elements, you use a Connector object. You can access this from either the Client or Supplier element, using the Connectors collection of that element. When creating a new connector you assign to it a valid type from this list:

- Aggregation
- Assembly
- Association
- Collaboration
- CommunicationPath
- Connector
- ControlFlow
- Delegate
- Dependency
- Deployment
- ERLink
- Generalization
- InformationFlow
- Instantiation
- InterruptFlow
- Manifest
- Nesting
- NoteLink
- ObjectFlow
- Package
- Realization
- Sequence
- StateFlow
- TemplateBinding
- UseCase

Associated table in repository

t_connector

Connector Attributes

Attribute	Remarks
Alias	String Notes: Read/Write

	An optional alias for this connector.
AssociationClass	Element Notes: Read Only Returns the Association Class element if the connector has one; otherwise NULL/.
ClientEnd	ConnectorEnd Notes: Read Only A pointer to the ConnectorEnd object representing the source end of the relationship.
ClientID	Long Notes: Read/Write The ElementID of the element at the source end of this connector.
Color	Long Notes: Read/Write Sets the color of the connector.
ConnectorGUID	String Notes: Read Only A system generated, globally unique ID for the current connector.
ConnectorID	Long Notes: Read Only A system generated local identifier for the current connector.
Constraints	Collection Notes: Read Only A collection of constraint objects.
ConveyedItems	Collection of type Element Notes: Read Only Returns a collection of elements that have been conveyed. To add another element to the conveyed Collection, use 'AddNew (ElementGUID,NULL)', where 'ElementGUID' is the GUID of the element to be added.
CustomProperties	Collection Notes: Read Only Returns a collection of advanced properties associated with an element in the form of CustomProperty objects.
DiagramID	Long Notes: Read/Write The DiagramID of the connector.
Direction	String

	<p>Notes: Read/Write</p> <p>The connector direction, which can be set to one of:</p> <ul style="list-style-type: none"> • Unspecified • Bi-Directional • Source -> Destination or • Destination -> Source <p>If the connector is non-navigable, set the 'sourceNavigability' and/or 'targetNavigability' attributes.</p>
EndPointX	<p>Long</p> <p>Notes: Read/Write</p> <p>The x-coordinate of the connector's end point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
EndPointY	<p>Long</p> <p>Notes: Read/Write</p> <p>The y-coordinate of the connector's end point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
EventFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags concerned with event signaling on messages.</p>
FQStereotype	<p>String</p> <p>Notes: Read Only</p> <p>The fully-qualified stereotype name in the format "Profile::Stereotype". One or more fully-qualified stereotype names can be assigned to StereotypeEx.</p>
ForeignKeyInformation	<p>String</p> <p>Notes: Read Only</p> <p>Returns the Foreign Key information.</p>
IsLeaf	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a leaf.</p>
IsRoot	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a root.</p>
IsSpec	<p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating that the connector is a specification.</p>
MessageArguments	<p>String</p> <p>Notes: Read Only</p>

	The connector Message arguments.
MetaType	String Notes: Read Only The connector's domain-specific meta type, as defined by an applied stereotype from an MDG Technology.
MiscData	String Notes: Read Only This low-level property returns an array providing information about the contents of the PData x fields. These database fields are not documented and developers must gain understanding of these fields through their own endeavors to use this property. MiscData is zero based, therefore: <ul style="list-style-type: none"> • MiscData(0) corresponds to PData1 • MiscData(1) corresponds to PData2, and so on
Name	String Notes: Read/Write The connector name.
Notes	String Notes: Read/Write Descriptive notes about the connector.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a Dispatch interface.
Properties	Properties Notes: Returns a list of specialized properties applicable to the connector that might not be available using the automation model. The properties are purposely undocumented because of their obscure nature and because they are subject to change as progressive enhancements are made to them.
ReturnValueAlias	String Notes: Shows the 'Return Value Alias' field of the operation.
RouteStyle	Long Notes: Read/Write The route style.
SequenceNo	Long Notes: Read/Write The SequenceNo of the connector.
StartPointX	Long Notes: Read/Write

	<p>The x-coordinate of the connector's start point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
StartPointY	<p>Long</p> <p>Notes: Read/Write</p> <p>The y-coordinate of the connector's start point.</p> <p>Connector end points are specified in Cartesian coordinates with the origin to the top left of the screen.</p>
StateFlags	<p>String</p> <p>Notes: Read/Write</p> <p>A structure to hold a variety of flags concerned with State signaling on messages; the list is delimited by semi-colons.</p>
Stereotype	<p>String</p> <p>Notes: Read/Write</p> <p>Sets or gets the stereotype for this connector end.</p>
StereotypeEx	<p>String</p> <p>Notes: Read/Write</p> <p>All the applied stereotypes of the connector in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully-qualified or simple names.</p>
StyleEx	<p>String</p> <p>Notes: Read/Write</p> <p>Advanced style settings; reserved for the use of Sparx Systems.</p>
Subtype	<p>String</p> <p>Notes: Read/Write</p> <p>A possible subtype to refine the meaning of the connector.</p>
SupplierEnd	<p>ConnectorEnd</p> <p>Notes: Read Only</p> <p>A pointer to the ConnectorEnd object representing the target end of the relationship.</p>
SupplierID	<p>Long</p> <p>Notes: Read/Write</p> <p>The ElementID of the element at the target end of this connector.</p>
TaggedValues	<p>Collection of type ConnectorTag</p> <p>Notes: Read Only</p> <p>The collection of ConnectorTag objects.</p>
TemplateBindings	<p>Collection of type TemplateBinding</p> <p>Notes: Read Only</p> <p>A collection of TemplateBinding objects.</p>

TransitionAction	String Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
TransitionEvent	String Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
TransitionGuard	String Notes: Read/Write See the <i>Transition</i> topic for appropriate values.
Type	String Notes: Read/Write The connector type; valid types are held in the t_connectortypes table in the .eap file.
TypeInfoProperties	Notes: Read only Returns an interface pointer of TypeInfoProperties.
VirtualInheritance	String Notes: Read/Write For Generalization, indicates if the inheritance is virtual.
Width	Long Notes: Read/Write Specifies the width of the connector.

Connector Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetTXAlias (string Code, long Flag)	String Notes: Returns the Alias of the element for a given language. Parameters <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Alias - 1 = Get the currently-stored translated Alias, and auto translate if the original Alias has changed

	- 2 = Always fetch the translated Alias from online
GetTXName (string Code, long Flag)	String Notes: Returns the name of the element for a given language. Parameters <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated name - 1 = Get the currently-stored translated name, and auto translate if the original name has changed - 2 = Always fetch the translated name from online
GetTXNote (string Code, long Flag)	String Returns the Notes of the element for a given language. Parameters <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Flag: Long <ul style="list-style-type: none"> - 0 = Get the currently-stored translated Notes - 1 = Get the currently-stored translated Notes, and auto translate if the original Notes have changed - 2 = Always fetch the translated Notes from online
IsConnectorValid()	Boolean Notes: Queries Enterprise Architect's internal relationship validation schema on the current connector. If False is returned, check the 'GetLastError()' function for more information.
SetTXAlias (string Code, string Translation)	String Notes - Set the translated Alias of the element for a given language. <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Alias
SetTXName (string Code, string Translation)	String Notes - Set the translated name of the element for a given language. <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated name
SetTXNote (string Code, string Translation)	String Notes - Set the translated Notes of the element for a given language. <ul style="list-style-type: none"> • Code: String - Two-letter language code (found on the 'Translations' page of the 'Manage Model Options' dialog) • Translation: String - The translated Notes
Update()	Boolean Notes: Updates the current ConnectorObject after modification or appending a new item.

	If False is returned, check the 'GetLastError()' function for more information.
--	---------------------------------------------------------------------------------

ConnectorConstraint Class

A ConnectorConstraint holds information about special conditions that apply to a connector. It is accessed through the Connector Constraints collection.

Associated table in repository

t_connectorconstraints

ConnectorConstraint Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write A local ID value (long) - system generated.
Name	String Notes: Read/Write The constraint name.
Notes	String Notes: Read/Write Notes about this constraint.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Type	String Notes: Read/Write The constraint type.

ConnectorConstraint Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current ConnectorConstraint object after modification or

	<p>appending a new item.</p>
--	------------------------------

If False is returned, check the 'GetLastError()' function for more information.

ConnectorEnd Class

A ConnectorEnd contains information about a single end of a connector. A ConnectorEnd is accessed from the connector as either the ClientEnd or SupplierEnd.

Associated table in repository

derived from t_connector

ConnectorEnd Attributes

Attribute	Remarks
Aggregation	Long Notes: Read/Write The type of Aggregation as it applies to this end; valid values are: 0 = None 1 = Shared 2 = Composite
Alias	String Notes: Read/Write An optional alias for this connector end.
AllowDuplicates	Boolean Notes: Read/Write For multiplicities greater than 1, indicates that duplicate entries are possible.
Cardinality	String Notes: Read/Write The cardinality associated with this end.
Constraint	String Notes: Read/Write A constraint that can be applied to this connector end.
Containment	String Notes: Read/Write The containment type applied to this connector end.
Derived	Boolean Notes: Read/Write Indicates that the value of this end is derived.
DerivedUnion	Boolean

	<p>Notes: Read/Write</p> <p>Indicates the value of this role derived from the union of all roles that subset this.</p>
End	<p>String</p> <p>Notes: Read only</p> <p>The end this ConnectorEnd object applies to - Client or Supplier.</p>
IsChangeable	<p>String</p> <p>Notes: Read/Write</p> <p>Flag indicating whether this end is changeable or not - 'frozen', 'addOnly' or none.</p>
IsNavigable	<p>Note: This property is not used</p> <p>Boolean</p> <p>Notes: Read/Write</p> <p>A flag indicating this end is navigable from the other end.</p>
Navigable	<p>String</p> <p>Notes: Read/Write</p> <p>Indicates whether this role of an association is navigable from the opposite classifier - Navigable, Non-Navigable or Unspecified.</p>
ObjectType	<p>ObjectType</p> <p>Notes: Read only</p> <p>Distinguishes objects referenced through a Dispatch interface.</p>
Ordering	<p>Long</p> <p>Notes: Read/Write</p> <p>Ordering for this connector end.</p>
OwnedByClassifier	<p>Boolean</p> <p>Notes: Read/Write</p> <p>Indicates that this Association end corresponds to an attribute on the opposite end of the Association.</p>
Qualifier	<p>String</p> <p>Notes: Read/Write</p> <p>A qualifier that can apply to the connector end.</p>
Role	<p>String</p> <p>Notes: Read/Write</p> <p>The connector end role.</p>
RoleNote	<p>String</p> <p>Notes: Read/Write</p> <p>Notes associated with the role of this connector end.</p>
RoleType	<p>String</p> <p>Notes: Read/Write</p>

	The role type applied to this end of the connector.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this connector end.
StereotypeEx	String Notes: Read/Write All the applied stereotypes of the connector end in a comma-separated list. Reading the value will provide the stereotype name only; assigning the value accepts either fully qualified or simple names.
TaggedValues	Collection of type RoleTag Notes: Read only A collection of RoleTag objects.
Visibility	String Notes: Read/Write The Scope associated with this connector end - Public, Private, Protected or Package.

ConnectorEnd Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current ConnectorEnd object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

ConnectorTag Class

A ConnectorTag is a Tagged Value for a connector and is accessed through the Connector TaggedValues collection.

Associated table in repository

t_connectortag

ConnectorTag Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write The local ID of the associated connector.
FQName	String Notes: Read only The fully qualified name of the tag.
Name	String Notes: Read/Write The tag or name.
Notes	String Notes: Read/Write Further descriptive notes on this tag. If 'Value' is set to '<memo>', then 'Notes' should contain the actual Tagged Value content.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
TagGUID	String Notes: Read/Write A globally unique ID for this Tagged Value.
TagID	Long Notes: Read only A local ID to identify the Tagged Value.
Value	String Notes: Read/Write The value assigned to this tag.

	<p>This field has a 255 character limit. If the value is greater than 255 characters long, set the value to "<memo>" and insert the body of text in the 'Notes' attribute.</p> <p>When reading existing Tagged Values, if 'Value' = "<memo>" then the developer should read the actual body of text from the 'Notes' attribute.</p>
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ConnectorTag Methods

Method	Remarks
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a Structured Tagged Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a Structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a Structured Tagged Value.
Update()	Boolean Notes: Update the current ConnectorTag object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

RoleTag Class

The RoleTag interface provides access to an Association's Role Tagged Values. Each connector end has a RoleTag collection that can be accessed to add, delete and access the RoleTags.

You might use this in creating code that resembles this fragment for accessing a RoleTag in VB.NET (where con is a Connector Object):

```
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
```

Associated table in repository

t_taggedvalue

RoleTag Attributes

Attribute	Description
BaseClass	String Notes: Read/Write Indicates the role end; set to ASSOCIATION_SOURCE or ASSOCIATION_TARGET.
ElementGUID	String Notes: Read/Write The GUID of the connector with which this role tag is associated.
FQName	String Notes: Read only The fully qualified name of the tag.
ObjectType	ObjectType

	Notes: Read only Distinguishes objects referenced through a Dispatch interface.
PropertyGUID	String Notes: Read/Write A system generated GUID to identify the Tagged Value.
Tag	String Notes: Read/Write The actual tag name.
Value	String Notes: Read/Write The value associated with this tag.

RoleTag Methods

Method	Description
GetAttribute(string propName)	String Notes: Returns the text of a single named property within a Structured Tagged Value.
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
HasAttributes()	Boolean Notes: Returns True if the Tagged Value is a Structured Tagged Value with one or more properties.
SetAttribute(string propName, string propValue)	Boolean Notes: Sets the text of a single named property within a Structured Tagged Value.
Update()	Boolean Notes: Update the RoleTag after changes or on initial creation. If False is returned, check the 'GetLastError()' function for more information.

TemplateBinding Class

A TemplateBinding defines the connector between a binding Class and a parameterized Class, and the binding expression on that connector.

TemplateBinding Attributes

Attribute	Remarks
ActualGUID	String Notes: Read/Write The GUID of the element classifier set as the Actual Template Binding parameter. If the Actual Template Binding parameter is set as a string expression only, this will be an empty string. Assigning a GUID value will automatically change the ActualName attribute after Update() has been called.
ActualName	String Notes: Read/Write The name of the Actual Template Binding parameter. Assigning a new value will clear any current ActualGUID value.
BindingExpression	String Notes: Read only The Binding Expression as shown in Enterprise Architect.
ConnectorGUID	String Notes: Read only The Globally Unique ID of the associated connector.
ConnectorType	String Notes: Read only The type of the associated connector.
FormalName	String Notes: Read/Write The name of the Formal Template Binding parameter.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch Interface.
Pos	String Notes: Read only The position of the Template Binding in the list (as on the 'Bindings' page of the connector 'Properties' dialog).

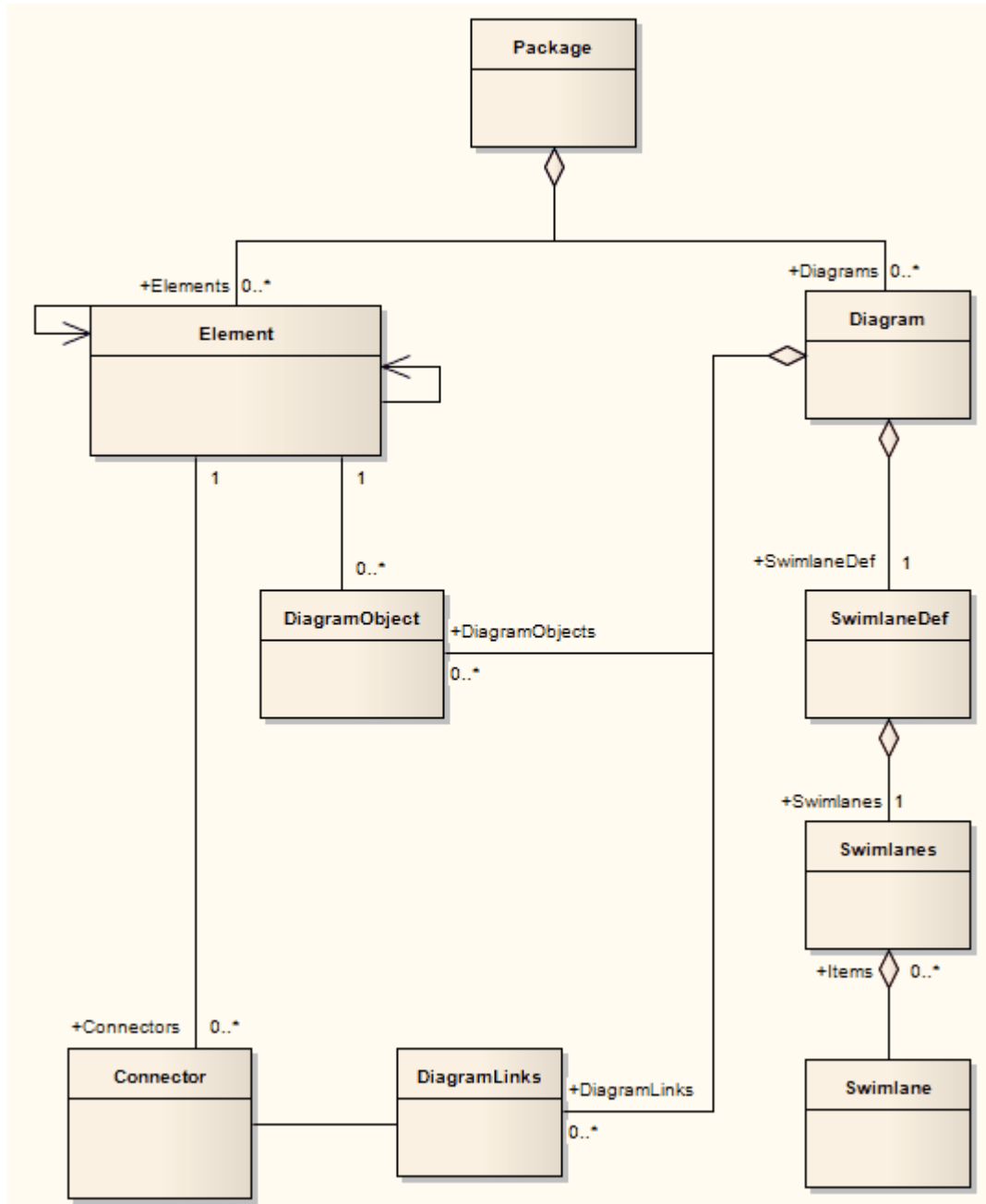
TemplateBindingID	String Notes: Read only The Globally Unique ID of the current Template Binding.
-------------------	---------------------------------------------------------------------------------------

TemplateBinding Methods

Method	Remarks
GetLastError()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
Update()	Boolean Notes: Update the current TemplateBinding object after modification or appending a new item. If False is returned, check the 'GetLastError()' function for more information.

Diagram Package

The Diagram Package has information on a diagram and on DiagramObject and DiagramLink, which are the instances of elements within a diagram.



Classe Diagramme

A Diagram corresponds to a single UML diagram. It is accessed through the Package Diagrams collection and in turn contains a collection of diagram objects and diagram connectors. Adding to the DiagramObject Class adds an existing element to the diagram. When adding a new diagram, you must set the diagram type to one of the valid types:

- Activity
- Analysis
- Component
- Custom
- Deployment
- Logical
- Sequence
- Statechart
- Use Case

For a Collaboration (Communication) diagram, use the Analysis type.

Associated table in repository

t_diagram

Diagram Attributes

Attribute	Remarks
Author	String Notes: Read/Write The name of the author.
CreatedDate	Date Notes: Read/Write The date the diagram was created.
cx	Long Notes: Read/Write The X dimension of the diagram (the default is 800).
cy	Long Notes: Read/Write The Y dimension of the diagram (the default is 1100).
DiagramGUID	Variant Notes: Read/Write A globally unique ID for this diagram.

DiagramID	Long Notes: Read only A local ID for the diagram.
DiagramLinks	Collection Notes: Read only A list of DiagramLink objects, each containing information about the display characteristics of a connector in a diagram.
DiagramObjects	Collection Notes: 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.
ExtendedStyle	String Notes: Read/Write An extended style attribute.
FilterElements	String Notes: Read/Write Applies a comma-separated list of object ids (from SelectedObjects) to the currently-applied diagram filter, overriding the filter. The effect persists until another filter is applied, or the diagram is closed.
HighlightImports	Boolean Notes: Read/Write A flag to indicate that elements from other Packages should be highlighted. Corresponds with the 'Show Namespace' option in the diagram 'Properties' dialog.
IsLocked	Boolean Notes: Read/Write A flag indicating whether this diagram is locked or not.
MetaType	String Notes: Read/Write The diagram's domain-specific meta type, as defined by an MDG Technology. When writing, the meta type must be fully qualified and from an existing profile.
ModifiedDate	Variant Notes: Read/Write The date the diagram was last modified.
Name	String Notes: Read/Write The diagram name.
Notes	String Notes: Read/Write Set or retrieve notes for this diagram.

ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Orientation	String Notes: Read/Write The page orientation: P for Portrait or L for Landscape.
PackageID	Long Notes: Read/Write The ID of the Package that this diagram belongs to.
PageHeight	Long Notes: Read The number of pages high the diagram is.
PageWidth	Long Notes: Read The number of pages wide the diagram is.
ParentID	Long Notes: Read/Write The optional ID of an element that 'owns' this diagram; for example, a Sequence diagram owned by a Use Case.
Scale	Long Notes: Read/Write The zoom scale (the default is 100).
SelectedConnector	Connector Notes: Read/Write The currently selected connector on this diagram. Null if there is no currently selected diagram.
SelectedObjects	Collection Notes: Read only Gets a collection representing the currently selected elements on the diagram. You can remove objects from this collection to deselect them, and add elements to the collection by passing the Object ID as a name to select them.
ShowDetails	Long Notes: Read/Write A flag to indicate that the Diagram Details text should be shown: 1 = Show, 0 = Hide.
ShowPackageContents	Boolean Notes: Read/Write A flag to indicate that the Package contents should be shown in the current

	diagram.
ShowPrivate	Boolean Notes: Read/Write A flag to show or hide Private features.
ShowProtected	Boolean Notes: Read/Write A flag to show or hide Protected features.
ShowPublic	Boolean Notes: Read/Write A flag to show or hide Public features.
Stereotype	String Notes: Read/Write Sets or gets the stereotype for this diagram.
StyleEx	String Notes: Read/Write Advanced style settings, reserved for the use of Sparx Systems.
Swimlanes	String Notes: Read/Write Information on swimlanes contained in the diagram. Please note that this property is superseded by SwimlaneDef.
SwimlaneDef	SwimlaneDef Notes: Read/Write Information on swimlanes contained in the diagram.
Type	String Notes: Read only The diagram type; see the t_diagramtypes table in the .cap file for more information.
Version	String Notes: Read/Write The version of the diagram.

Diagram Methods

Method	Details
ApplyGroupLock (string	Boolean

aGroupName)	<p>Notes: Applies a group lock to this diagram object, for the specified group, on behalf of the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p> <p>Parameter:</p> <ul style="list-style-type: none"> • aGroupName: String - the name of the user group for which to set the group lock
ApplyUserLock ()	<p>Boolean</p> <p>Notes: Applies a user lock to this diagram object, for the current user.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
FindElementInDiagram (long ElementID)	<p>Boolean</p> <p>Notes: This function activates the Diagram View and displays the diagram with the diagram object selected. If the diagram is too large to display all of it on the screen, the portion of the diagram containing the object is displayed with the object shown in the center of the screen. Diagram objects flagged as non-selected are shown but are not selected</p> <p>Returns True if the diagram object was found, the diagram displayed and the object selected (or at least displayed) in the view. Returns False if the diagram object was not found in the diagram and the diagram not displayed.</p> <p>Parameter</p> <ul style="list-style-type: none"> • ElementID: Long - the element ID of the diagram object to locate
GetDiagramObjectByID (long ID, string DUID)	<p>DiagramObject</p> <p>Notes: Returns the DiagramObject object, if it exists on the diagram.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ID: Long - the ElementID of the diagram object • DUID: String - the optional Diagram Unique ID of the diagram object
GetElementByGrid (string GridX, string GridY)	<p>Element</p> <p>Notes: Uses the Excel type of format to specify the column and row of a grid at which an element should be found: A 5, CB 1.</p> <p>Returns null if no element is at the specified position.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GridX: string - Column A to Z • GridY: string - Number of row
GetElementByName (string Name)	<p>Element</p> <p>Notes: Locates an element with the specified name.</p> <p>Returns null if no element is found with that name.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: string - Name of the element to find
GetObjectByGrid (string GridX, string GridY)	<p>DiagramObject</p> <p>Notes: Uses the Excel type of format to specify the column and row of a grid at which an object should be found: A 5, CB 1.</p> <p>Returns null if no element is at the specified position.</p>

	<p>Parameters:</p> <ul style="list-style-type: none"> • GridX: string - Column A to Z • GridY: string - Number of row
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
ReadStyle (string StyleName)	<p>String</p> <p>Notes: Returns the current value of the named diagram style.</p> <p>Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are: <ul style="list-style-type: none"> - Show Element Property String - Show Connector Property String - Show Feature Property String
ReleaseUserLock ()	<p>Boolean</p> <p>Notes: Releases a group lock or user lock on this diagram object.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful. Use GetLastError() to retrieve error information.</p>
ReorderMessages ()	<p>Void</p> <p>Notes: Resets the display order of Sequence and Collaboration messages. This is typically used after inserting or deleting messages in the diagram.</p>
SaveAsPDF (string FileName)	<p>Boolean</p> <p>Notes: Exports the diagram to a PDF document. Returns True on success.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - full path to file location
SaveImagePage(long x, long y, long sizeX, long sizeY, string filename, long flags)	<p>Boolean</p> <p>Notes: Saves a page of the diagram to disk.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful.</p> <p>Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • x: Long - the horizontal page • y: Long - the vertical page • sizeX: Long - currently unused; pass a value of 0 to ensure behavior does not change in a future build • sizeY: Long - currently unused; pass a value of 0 to ensure behavior does not change in a future build • filename: String - the filename and path to save the image • flags: Long - additional options, currently unused; pass a value of 0 to ensure behavior does not change in a future build <p>The image type is determined by the extension of the filename. Currently only .emf, .bmp and .png formats are supported.</p>

<p>ShowAsElementList (bool ShowAsList, bool Persist)</p>	<p>Boolean</p> <p>Notes: Toggles the diagram display between diagram format and Diagram List depending on the value of ShowAsList.</p> <p>If Persist is set, the display format is written to the database so the diagram always opens in that format (diagram or list). Otherwise, the display format falls back to the default (diagram) once the display is closed.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ShowAsList: Boolean - indicates diagram or Diagram List • Persist: Boolean - indicates set (maintain ShowAsList value) or not (revert to default)
<p>Update ()</p>	<p>Boolean</p> <p>Notes: Updates this diagram object after modification or appending a new item. If False is returned, use GetLastError() to retrieve error information.</p>
<p>VirtualizeConnector (int ConnectorID, int Action, int X, int Y)</p>	<p>Boolean</p> <p>Notes: Creates a virtual copy of the source or target element on a connector, and sets its location on the diagram as a waypoint on the connector. If the source element is being virtualized, the waypoint is created as the first on the connector, and if the target element is being virtualized, the waypoint is created as the last on the connector.</p> <p>If called again on the same connector, removes the virtual element. However, the waypoint remains in place.</p> <p>As waypoints and therefore virtual elements can only be created on connectors with the Custom line-style, if the connector does not have this line style the method sets it. So, after this method executes, an Update function should be called for the connector as well as for the diagram. All parameters are required for the function to complete successfully.</p> <p>Returns True if the operation is successful; returns False if the operation is unsuccessful.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ConnectorID - Integer: the ID of the connector on which to create the virtual element • Action - Integer: the element to be virtualized; 1 for the source element, 2 for the target element • X - Integer: the position on the X axis that the element's center point will be aligned with • Y - Integer: the position on the Y axis that the element's center point will be aligned with <p>For example, to virtualize the source element of the selected connector:</p> <pre>function main() { var diagram as EA.Diagram; var conn as EA.Connector; diagram = Repository.GetCurrentDiagram(); if(diagram != null) { var connector as EA.Connector. connector = diagram.SelectedConnector;</pre>

	<pre> diagram.VirtualizeConnector(connector.ConnectorID, 1, 100, 150); connector.Update(); diagram.Update(); Repository.ReloadDiagram(diagram.DiagramID); } else { Session.Output("Script requires a diagram to be visible"); } } main();</pre>
WriteStyle (string StyleName, string StyleValue)	<p>Void</p> <p>Notes: Sets the value of the named diagram style. Use GetLastError() to retrieve error information.</p> <p>Parameters:</p> <ul style="list-style-type: none">• StyleName: String - the name of the diagram style whose value is to be retrieved; valid StyleNames are:<ul style="list-style-type: none">- Show Element Property String- Show Connector Property String- Show Feature Property String• StyleValue: String - the value to be set in the named diagram style; valid values for the StyleNames listed are 0 and 1

DiagramLink Class

A DiagramLink is an object that holds display information on a connector between two elements in a specific diagram. It includes, for example, the custom points and display appearance. It can be accessed from the Diagram DiagramLinks collection.

Associated table in repository

t_diagramlinks

DiagramLink Attributes

Attribute	Remarks
ConnectorID	Long Notes: Read/Write The ID of the associated connector.
DiagramID	Long Notes: Read/Write The local ID for the associated diagram.
Geometry	String Notes: Read/Write The geometry associated with the current connector in this diagram.
HiddenLabels	Boolean Notes: Indicates if this connector's labels are hidden on the diagram.
InstanceID	Long Notes: Read only The connector identifier for the current model.
IsHidden	Boolean Notes: Read/Write Indicates if this item is hidden or not.
LineColor	Long Notes: Sets the line color of the connector. Set to -1 to reset to the default color in the model.
LineStyle	Long Notes: Sets the line style of the connector. 1 = Direct 2 = Auto Routing

	<p>3 = Custom Line 4 = Tree Vertical 5 = Tree Horizontal 6 = Lateral Vertical 7 = Lateral Horizontal 8 = Orthogonal Square 9 = Orthogonal Rounded</p>
LineWidth	<p>Long Notes: Sets the line width of the connector.</p>
ObjectType	<p>ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.</p>
Path	<p>String Notes: Read/Write The path of the connector in this diagram.</p>
SourceInstanceUID	<p>String Notes: Read only Returns the Unique Identifier of the source object.</p>
SuppressSegment	<p>Long Notes: Read/Write Returns the index of the line segment that has been suppressed. Returns 0 when no segments are suppressed.</p>
Style	<p>String Notes: Read/Write Additional style information; for example, color or thickness.</p>
TargetInstanceUID	<p>String Notes: Read only Returns the Unique Identifier of the target object.</p>

DiagramLink Methods

Method	Remarks
GetLastError()	<p>String Notes: Returns a string value describing the most recent error that occurred in relation to this object. This function is rarely used as an exception is thrown when an error occurs.</p>

Update()	<p>Boolean</p> <p>Notes: Update the current DiagramLink object after modification or appending a new item.</p> <p>If False is returned, check the 'GetLastError()' function for more information.</p>
----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Classe DiagramObject

The DiagramObject Class stores presentation information that indicates what is displayed in a diagram and how it is shown.

Associated Table in Repository

t_diagramobjects

DiagramObject Attributes

Attribute	Remarks
BackgroundColor	Long Notes: The background color of the object on the diagram. Set to -1 to re-set to the default color in the model.
BorderColor	Long Notes: The border line color of the object on the diagram. Set to -1 to re-set to the default color in the model.
BorderLineWidth	Long Notes: The border line width of the object on the diagram. Valid values are 1 (narrowest) to 5 (thickest); a default of 1 is applied if an invalid value is passed in.
Bottom	Long Notes: Read/Write The bottom edge position of the object on the diagram. Enterprise Architect uses a cartesian coordinate system, with {0,0} being the top-left corner of the diagram. For this reason, Y-axis values (Top and Bottom) should always be negative.
DiagramID	Long Notes: Read/Write The ID of the associated diagram.
ElementDisplayMode	Long Notes: Indicates how to adjust the element features if the element is resized. 1 = Resize to longest feature 2 = Wrap features 3 = Truncate features Defaults to 1 if an invalid value is supplied.
ElementID	Long Notes: Read/Write

	The ElementID of the object instance in this diagram.
FeatureStereotypesToHide	String Notes: Lists the stereotypes to hide on the object on the diagram.
FontBold	Boolean Notes: Get or Set the status of the object text font as Bold.
FontColor	Long Notes: The color of the font of the object text on the diagram.
FontItalic	Boolean Notes: Get or Set the status of the object text font as Italic.
FontName	String Notes: The name of the font used for the object text.
FontSize	String Notes: The size of the font used for the object text.
FontUnderline	Boolean Notes: Get or Set the status of the object text font as Underlined.
InstanceGUID	String Notes: The instance GUID for the object on the diagram (the DUID).
InstanceID	Long Notes: Read Holds the connector identifier for the current model.
IsSelectable	Boolean Notes: Indicates whether this object on the diagram can be selected.
Left	Long Notes: Read/Write The left edge position of the object on the diagram.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Right	Long Notes: Read/Write The right edge position of the object on the diagram.
Sequence	Long Notes: Read/Write The sequence position when loading the object into the diagram (this affects its Z

	order). The Z-order is one-based and the lowest value is in the foreground.
ShowComposedDiagram	Boolean Notes: Indicates whether the object's composite diagram should be displayed by default when the object is selected.
ShowConstraints	Boolean Notes: Show constraints for this object on the diagram.
ShowFormattedNotes	Boolean Notes: Show any formatting applied to the notes, for this object on the diagram. ShowNotes must be True for the formatted notes to be displayed.
ShowFullyQualifiedTags	Boolean Notes: Show fully qualified Tagged Values for this object on the diagram.
ShowInheritedAttributes	Boolean Notes: Show inherited attributes for this object on the diagram.
ShowInheritedConstraints	Boolean Notes: Show inherited constraints for this object on the diagram.
ShowInheritedOperations	Boolean Notes: Show inherited operations for this object on the diagram.
ShowInheritedResponsibilities	Boolean Notes: Show the inherited requirements within the Requirements compartment for this object on the diagram.
ShowInheritedTags	Boolean Notes: Show inherited Tagged Values for this object on the diagram.
ShowNotes	Boolean Note: Show the notes for this object on the diagram.
ShowPackageAttributes	Boolean Notes: Show Package attributes for this object on the diagram.
ShowPackageOperations	Boolean Notes: Show Package operations for this object on the diagram.
ShowPortType	Boolean Notes: Show the Port type.
ShowPrivateAttributes	Boolean Notes: Show private attributes for this object on the diagram.
ShowPrivateOperations	Boolean

	Notes: Show private operations for this object on the diagram.
ShowProtectedAttributes	Boolean Notes: Show protected attributes for this object on the diagram.
ShowProtectedOperations	Boolean Notes: Show protected operations for this object on the diagram.
ShowPublicAttributes	Boolean Notes: Show public attributes for this object on the diagram.
ShowPublicOperations	Boolean Notes: Show public operations for this object on the diagram.
ShowResponsibilities	Boolean Notes: Show the requirements compartment for this object on the diagram.
ShowRunstates	Boolean Notes: Show Runstates for this object on the diagram.
ShowStructuredCompartment ents	Boolean Note: Indicates whether to display the Structure Compartments for this object on the diagram.
ShowTags	Boolean Notes: Show Tagged Values for this object on the diagram.
Style	Variant Notes: Read/Write The style information for this object. Returns a semi-colon delimited string that defines the current style settings. Changing a value will completely overwrite the previously existing value, so caution is advised to avoid losing existing style information that you want to keep. See <i>Setting the Style</i> .
TextAlign	Long Notes: Indicates the alignment of text on a Text element on the diagram. 1 = Left aligned 2 = Center aligned 3 = Right aligned Defaults to 1 if an invalid value is supplied.
Top	Long Notes: Read/Write The top edge position of the object on the diagram. Enterprise Architect uses a cartesian coordinate system, with {0,0} being the top-left corner of the diagram. For this reason, Y-axis values (Top and Bottom) should always be negative.

DiagramObject Methods

Method	Remarks
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
MoveElementToGridPosition (GridX, GridY)	Notes: Currently not implemented.
ResetFont	Notes: Resets the font of the object text on the diagram back to the model default.
SetFontStyle (FontName, FontSize, Bold, Italic, Underline)	Boolean Notes: Sets the font of the object text on the diagram to the specified values.
SetStyleEx (string Parameter, string Value)	Void Notes: Sets an individual parameter of the Style string. Parameters: <ul style="list-style-type: none"> Parameter: String - the name of the style parameter to modify; for example: <ul style="list-style-type: none"> "BCol" = background color "BFol" = font color "LCol" = line color "LWth" = line width Value: String - the new value for the style parameter
Update ()	Boolean Notes: Updates the current DiagramObject after modification or appending a new item If False is returned, check the GetLastError function for more information.

Setting the Style

The Style attribute contains various settings that affect the appearance of a DiagramObject. However, it is not recommended to directly edit this attribute string. Instead, use either the SetStyleEx method or one of the individual DiagramObject attributes such as BackgroundColor, FontColor or BorderColor.

For example, the Style string might contain a series of values in a format such as:

```
BCol=n;BFol=n;LCol=n;LWth=n;
```

where:

- BCol = Background Color
- BFol = Font Color
- LCol = Line Color
- LWth = Line Width

The value assigned to each of the Style color properties is a decimal representation of the hex RGB value, where Red=FF, Green=FF00 and Blue=FF0000.

This code snippet shows how you might change the style settings for all of the objects in the current diagram, changing the background color to red (FF=255) and the font and line colors to yellow (FFFF=65535):

```
For Each aDiagObj In aDiag.DiagramObjects
    aDiagObj.BackgroundColor=255
    aDiagObj.FontColor=65535
    aDiagObj.BorderColor=65535
    aDiagObj.BorderLineWidth=1
    aDiagObj.Update
    aRepos.ReloadDiagram aDiagObj.DiagramID
```

Next

SwimlaneDef Class

A SwimlaneDef object makes available attributes relating to a single row or column in a list of swimlanes.

SwimlaneDef Attributes

Attribute	Description
Bold	Boolean Notes: Read/Write Show the title text in bold.
FontColor	Long Notes: Read/Write The RGB color used to draw the titles.
HideClassifier	Boolean Notes: Read/Write Removes any classifier from the title display.
HideNames	Boolean Notes: Read/Write Set to True to hide the swimlane titles.
LineColor	Long Notes: Read/Write The RGB color used to draw swimlane borders.
LineWidth	Long Notes: Read/Write The width, in pixels, of the line used to draw swimlanes. Valid values are 1, 2 or 3.
Locked	Boolean Notes: Read/Write If set to True, disables user modification of the swimlanes via the diagram.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Orientation	String Notes: Read/Write Indicates whether the swimlanes are vertical or horizontal.
ShowInTitleBar	Boolean Notes: Read/Write

	Enables vertical swimlane titles to be shown in the title bar.
Swimlanes	Swimlanes Notes: Read/Write A list of individual swimlanes.

Swimlanes Class

A Swimlanes object is attached to a diagram's SwimlaneDef object and provides a mechanism to access individual swimlanes.

Swimlanes Attributes

Attribute	Description
Count	Long Notes: Read/Write Gives the number of swimlanes.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Swimlanes Methods

Method	Description
Add(string Title, long Width)	Swimlane Notes: Adds a new swimlane to the end of the list, and returns a swimlane object representing the newly added entry. Parameters: <ul style="list-style-type: none"> Title: String - The title text that appears at the top of the swimlane; this can be the same as an existing swimlane title Width: Long - The width of the swimlane in pixels
Delete(object Index)	Void Notes: Deletes a selected swimlane. If the string matches more than one entry, only the first entry is deleted. Parameter: <ul style="list-style-type: none"> Index: Object - Either a string representing the title text or an integer representing the zero-based index of the swimlane to delete
DeleteAll()	Void Notes: Removes all swimlanes.
Insert(long Index, string Title, long Width)	Swimlane Notes: Inserts a swimlane at a specific position, and returns a swimlane object representing the newly added entry. Parameters: <ul style="list-style-type: none"> Index: Long - The zero-based index of the existing Swimlane before which this

	<p>new entry is inserted</p> <ul style="list-style-type: none">• Title: String - The title text that appears at the top of the swimlane; this can be the same as an existing swimlane title• Width: Long - The width of the swimlane in pixels
Items(object Index)	<p>Swimlane collection</p> <p>Notes: Accesses an individual swimlane.</p> <p>If the string matches more than one swimlane title, the first matching swimlane is returned.</p> <p>Parameter:</p> <ul style="list-style-type: none">• Index: Object - Either a string representing the title text or an integer representing the zero-based index of the swimlane to get

Swimlane Class

A Swimlane object makes available attributes relating to a single row or column in a list of swimlanes.

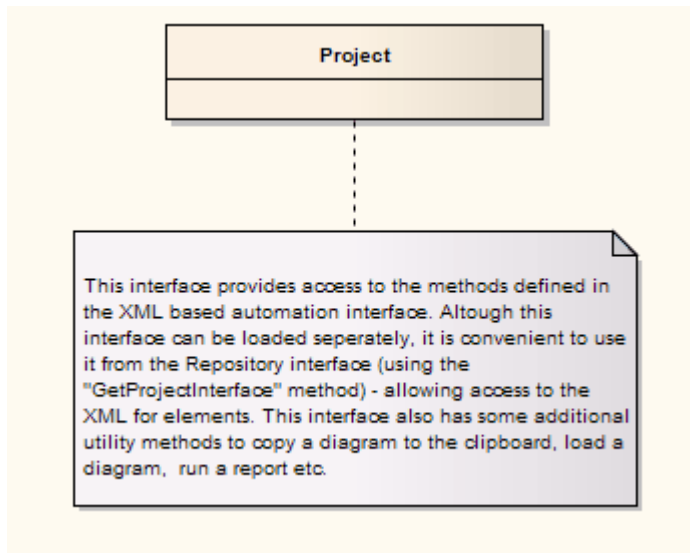
Swimlane Attributes

Attribute	Description
BackColor	Long Notes: Read/Write The RGB color that the swimlane is filled with.
ClassifiedGuid	String Notes: Read/Write The GUID of the classifier Class. This can be obtained from the corresponding element object via the ElementGUID property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Title	String Notes: Read/Write The text at the head of the swimlane.
Width	Long Notes: Read/Write The width of the swimlane, in pixels.

Project Interface Package

The Enterprise Architect.Project interface. This is the interface to Enterprise Architect elements; it also includes some utility functions. You can get a pointer to this interface using the Repository.GetProjectInterface method.

Example



Classe de projet

The Project interface can be accessed from the Repository using `GetProjectInterface()`. The returned interface provides access to the XML-based Enterprise Architect Automation Interface. Use this interface to get XML for the various internal elements and to run some utility functions to perform tasks such as load diagrams or run reports.

Project Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

Notes

- The Project methods listed here all require input GUIDs in XML format; use **GUIDtoXML** to change the Enterprise Architect GUID to an XML GUID

Project Methods

Method	Remarks
BuildExecutableStateMachine (string ElementGUID, string ExtraOptions)	Boolean Notes: Builds Executable StateMachine code for an <<executable statemachine>> Artifact element. Parameters: <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to generate ExtraOptions: String - enables extra options to be given to the command (currently unused)
CancelValidation ()	Void Notes: Cancels a validation process.
CanValidate ()	Boolean Notes: Returns a value to indicate that the Model Validation component is loaded.
CreateBaseline (string PackageGUID, string Version, string Notes)	Boolean Notes: Creates a Baseline of a specified Package. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to Baseline Version: String - the version of the Baseline Notes: String - any notes concerning the Baseline

<p>CreateBaselineEx (string PackageGUID, string Version, string Notes, EA.CreateBaselineFlag Flags)</p>	<p>Boolean</p> <p>Notes: Creates a Baseline of a specified Package, with a flag to exclude Package contents below the first level.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to be Baselined • Version: String - the version of the Baseline • Notes: String - any notes concerning the Baseline • Flags: EA.CreateBaselineFlag - whether or not to exclude the Package contents below the first level
<p>DefineRule (string CategoryID, EA.EnumMVErrorType ErrorType, string ErrorMessage)</p>	<p>String</p> <p>Notes: Defines the individual rules that can be performed during model validation. It must be called once for each rule from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a RuleId, which can be used for reference purposes when an individual rule is executed by Enterprise Architect during model validation.</p> <p>See the <i>Model Validation Example</i> for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryId: String - should be passed the return value from the DefineRuleCategory method • ErrorType: EA.EnumMVErrorType - depending on the severity of the error being validated, can be: <ul style="list-style-type: none"> - mvErrorCritical - mvError - mvWarning, or - mvInformation • ErrorMessage: String - can contain a default error string, although this is probably overridden by the PublishResult call
<p>DefineRuleCategory (string CategoryName)</p>	<p>String</p> <p>Notes: Defines a category of rules that can be performed during model validation (there is typically one category per Add-In). It must be called once from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a CategoryId that must to be passed to the DefineRule method.</p> <p>See the <i>Model Validation Example</i> for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryName: String - a text string that is visible in the 'Model Validation Configuration' dialog
<p>DeleteBaseline (string BaselineGUID)</p>	<p>Boolean</p> <p>Notes: Deletes a Baseline, identified by the BaselineGUID, from the repository. If the repository is configured to store Baselines in a Reusable Asset Service Registry, then it is not possible to delete the Baseline and a False value is returned.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • BaselineGUID: String - the GUID (in XML format) of the Baseline to delete
<p>DoBaselineCompare (string PackageGUID, string Baseline, string BaselineGUID)</p>	<p>String</p> <p>Notes: Performs a Baseline comparison using the supplied Package GUID and Baseline GUID (obtained in the result list from GetBaselines).</p>

ConnectString)	<p>Optionally you can include the connection string required to find the Baseline if it exists in a different model file.</p> <p>This method returns a log file of the status of all elements found and compared in the difference procedure. You can use this log information as input to DoBaselineMerge - automatically merging information from the Baseline.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to run the comparison on • Baseline: String - the GUID (in XML format) of the Baseline to run the comparison on • ConnectString: String - not currently used
DoBaselineMerge (string PackageGUID, string Baseline, string MergeInstructions, string ConnectString)	<p>String</p> <p>Notes: Performs a batch merge based on instructions contained in an XML file (MergeInstructions). You can supply an optional connection string if the Baseline is located in another model.</p> <p>In the MergeInstructions file, each MergeItem node supplies the GUID of a differenced item from the XML difference log. As the merge is uni-directional and actioned in only one possible way, no additional arguments are required. Enterprise Architect chooses the correct procedure based on the 'Difference' results.</p> <pre><Merge> <MergeItem guid="{XXXXXX}" /> <MergeItem guid="{XXXXXX}" /> </Merge></pre> <p>Alternatively, you can supply a single Mergeitem with a GUID of RestoreAll. In this case, Enterprise Architect batch-processes ALL differences.</p> <pre><Merge> <MergeItem guid="RestoreAll" changed="true" baselineOnly="true" modelOnly="true" moved="true" fullRestore="false" /> </Merge></pre> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to merge the Baseline into • Baseline: String - the GUID of the Baseline (in XML format) to merge into the Package • MergeInstructions: String - the file containing the GUID of each differenced item from the XML difference log returned by DoBaselineCompare() • ConnectString: String - not currently used
EnumDiagramElements (string DiagramGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of all elements in a diagram.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to get elements for
EnumDiagrams (string PackageGUID)	<p>protected abstract: String</p> <p>Notes: Gets an XML list of all diagrams in a specified Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to list diagrams for

EnumElements (string PackageGUID)	protected abstract: String Notes: Gets an XML list of elements in a specified Package. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to get a list of elements for
EnumLinks (string ElementGUID)	protected abstract: String Notes: Gets an XML list of connectors for a specified element. Parameters: <ul style="list-style-type: none"> ElementGUID: String - the GUID (in XML format) of the element to get all associated connectors for
EnumPackages (string PackageGUID)	protected abstract: String Notes: Gets an XML list of child Packages inside a parent Package. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the parent Package
EnumProjects ()	protected abstract: String Notes: Gets a list of projects in the current file; corresponds to Models in Repository.
EnumViewEx (string ProjectGUID)	protected abstract: String Notes: Gets a list of Views in the current project. Parameters: <ul style="list-style-type: none"> ProjectGUID: String - the GUID (in XML format) of the project to get views for
EnumViews ()	protected abstract: String Notes: Enumerates the Views for a project. Returned as an XML document.
Exit ()	protected abstract: String Notes: Exits the current instance of Enterprise Architect; this function is maintained for backward compatibility and should never be called. Enterprise Architect automatically exits when you are no longer using any of the provided objects.
ExportPackageXMI (string PackageGUID, enumXMIMType XMIMType, long DiagramXML, long DiagramImage, long FormatXML, long UseDTD, string FileName)	protected abstract: String Notes: Exports XMI for a specified Package. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to be exported XMIMType: EnumXMIMType - specifies the XMI type and version information; see <i>XMIMType Enum</i> for accepted values DiagramXML: Long - True if XML for diagrams is required; accepted values: 0 = Do not export diagrams 1 = Export diagrams 2 = Export diagrams along with alternative images DiagramImage: Long - the format for diagram images to be created at the same time; accepted values:

	<p>-1 = NONE 0 = EMF 1 = BMP 2 = GIF 3 = PNG 4 = JPG</p> <ul style="list-style-type: none"> • FormatXML: Long - True if XML output should be formatted prior to saving • UseDTD: Long - True if a DTD should be used • FileName: String - the filename to output to
<p>ExportPackageXMIEx (string PackageGUID, enumXMIMType XMIMType, long DiagramXML, long DiagramImage, long FormatXML, long UseDTD, string FileName, ea.ExportPackageXMIFlag Flags)</p>	<p>protected abstract: String</p> <p>Notes: Exports XMI for a specified Package, with a flag to determine whether the export includes Package content below the first level.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to be exported • XMIMType: EnumXMIMType - specifies the XMI type and version information; see <i>XMIMType Enum</i> for accepted values • DiagramXML: Long - true if XML for diagrams is required; accepted values: 0 = Do not export diagrams 1 = Export diagrams 2 = Export diagrams along with alternative images • DiagramImage: Long - the format for diagram images to be created at the same time; accepted values: -1 =NONE 0 =EMF 1 =BMP 2 =GIF 3 =PNG 4 =JPG • FormatXML: Long - True if XML output should be formatted prior to saving • UseDTD: Long - True if a DTD should be used. • FileName: String - the filename to output to • Flags: ea.ExportPackageXMIFlag - specify whether or not to include Package content below the first level (currently supported for xmiEADefault), whether or not to exclude tool-specific information from export
<p>ExportProjectXML (string DirectoryPath)</p>	<p>Boolean</p> <p>Notes: Exports the entire current project to Native XML files in the specified directory. The contents of the directory will be deleted prior to exporting the project data</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DirectoryPath: String - directory path to save the exported Native XML files
<p>ExportReferenceData (string FileName, string Tables)</p>	<p>Boolean</p> <p>Notes: Exports Reference Data.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the file to output the reference data to • Tables: String - the list of reference data tables to be output; the data table delimiter is ";" If the string is empty, Enterprise Architect will prompt with a dialog to select the tables to output

<p>GenerateBuildRunExecutableStateMachine (string ElementGUID, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Generates, builds and runs Executable StateMachine code for an <<executable statemachine>> Artifact element, which will start simulation of the StateMachine.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to generate • ExtraOptions: String - enables extra options to be given to the command (currently unused)
<p>GenerateClass (string ElementGUID, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Generates the code for a single Class.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to generate • ExtraOptions: String - enables extra options to be given to the command; currently unused
<p>GenerateDiagramFromScenario (string ElementGUID, EnumScenarioDiagramType DiagramType, long OverwriteExistingDiagram)</p>	<p>Boolean</p> <p>Notes: Generates various diagrams from the scenario specification of an element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element containing the scenario specification • DiagramType: EnumScenarioDiagramType - the type of diagram to generate; see ScenarioDiagramType Enum for accepted values • OverwriteExistingDiagram: Long - determines whether to overwrite the existing diagram or synchronize the existing elements with the scenario steps <ul style="list-style-type: none"> 0 = Delete the existing diagram and elements, and create a new diagram and elements 1 = Synchronize existing elements with the scenario steps and preserve the diagram layout 2 = Synchronize existing elements with the scenario steps and re-cast the diagram layout 3 = Do not generate a diagram if one already exists
<p>GenerateElementDDL (string ElementGUID, string FileName, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Generates DDL for an element using the options that are currently set on the Generate DDL screen.</p>
<p>GenerateExecutableStateMachine (string ElementGUID, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Generates Executable StateMachine code for an <<executable statemachine>> Artifact element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to generate • ExtraOptions: String - enables extra options to be given to the command (currently unused, pass an empty string to ensure current behavior in future versions)
<p>GeneratePackage (string PackageGUID, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Generates the code for all Classes within a Package.</p> <p>For example:</p>

	<pre>recurse=1;overwrite=1;dir=C:\</pre> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to generate code for • ExtraOptions: String - enables extra options to be given to the command; currently enables: <ul style="list-style-type: none"> - Generation of all sub-Packages (recurse) - Force overwrite of all files (overwrite) and - Specification to auto generate all paths (dir)
GeneratePackageDDL (string PackageGUID, string FileName, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates DDL for all elements in a Package using the options that are currently set on the Generate DDL screen.</p>
GenerateTestFromScenario (string ElementGUID, EnumScenarioTestType TestType)	<p>Boolean</p> <p>Notes: Generates a Vertical Test Suite, a Horizontal Test Suite, an Internal test or an External test from the scenario specification of an element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element containing the scenario specification • TestType: EnumScenarioTestType - the type of test to generate; see <i>ScenarioTestType Enum</i> for accepted values
GenerateWSDL(string WSDLComponentGUID, string Filename, string Encoding, string ExtraOptions)	<p>Boolean</p> <p>Notes: Generates WSDL for the specified WSDL stereotyped Component.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • WSDLComponentGUID: String - the GUID (in XML format) of the WSDL stereotyped Component • Filename: String - the target file path • Encoding: String - the XML encoding for the code page instruction • ExtraOptions: String - enables extra options to be given to the command; currently unused
GenerateXSD (string PackageGUID, string FileName, string Encoding, string Options)	<p>Boolean</p> <p>Notes: Creates an XML schema for a Package, specified by its GUID. Returns True on success.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package • FileName: String - the target filepath • Encoding: String - the XML encoding for the code page instruction • Options: String - enables extra options to be given to the command, in a comma-separated string; currently enables: <ul style="list-style-type: none"> - GenGlobalElement - turn the generation of global elements for all global ComplexTypes On or Off; for example: GenGlobalElement=1 - UseRelativePath - turns on or off the option to use a relative path in the XSD import or XSD include statement when referencing external Package, provided the schemaLocation tag is empty on the referenced Packages; for example: UseRelativePath=1

<p>GetAllDiagramImagesAndMap (string Directory)</p>	<p>Boolean</p> <p>Notes : Saves the image and image-map for every diagram in the model, in the specified directory location.</p> <p>The image files will be saved in PNG format and each will have the diagram GUID as the image name. The image-map files will be saved as .txt files and each will have the diagram GUID as the image map name.</p> <p>The 'Auto Create Diagram Image and Image Map' option must be selected in the model options for this function to save the images and image-maps.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Directory – the location of the directory into which the images and image-maps are to be saved
<p>GetBaselines (string PackageGUID, string ConnectString)</p>	<p>String</p> <p>Notes: Returns a list (in XML format) of Baselines associated with the supplied Package GUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to get Baselines for • ConnectString: String - not currently used
<p>GetDiagram (string DiagramGUID)</p>	<p>protected abstract: String</p> <p>Notes: Gets the diagram details, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to get details for
<p>GetDiagramImageAndMap (string DiagramGUID, string Directory)</p>	<p>Boolean</p> <p>Notes: Saves the image and image-map for the diagram with the specified GUID, in the specified directory location.</p> <p>The image will be saved in PNG format and will have the DiagramGUID as the image name. The image-map will be saved as a .txt file and will have the DiagramGUID as the image-map name.</p> <p>The 'Auto Create Diagram Image and Image Map' option must be selected in the model-specific options for this function to save the image and image-map.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID – the GUID of the diagram for which the image and image-map are to be saved • Directory – the directory into which the image and image-map are to be saved
<p>GetElement (string ElementGUID)</p>	<p>protected abstract: String</p> <p>Notes: Gets XML for the specified element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to retrieve XML for
<p>GetElementConstraints (string ElementGUID)</p>	<p>protected abstract: String</p> <p>Notes: Gets constraints for an element, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element

GetElementEffort (string ElementGUID)	protected abstract: String Notes: Gets efforts for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementFiles (string ElementGUID)	protected abstract: String Notes: Gets metrics for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementMetrics (string ElementGUID)	protected abstract: String Notes: Gets files for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementProblems (string ElementGUID)	protected abstract: String Notes: Gets a list of issues (problems) associated with an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementProperties (string ElementGUID)	protected abstract: String Notes: Gets Tagged Values for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementRequirements (string ElementGUID)	protected abstract: String Notes: Gets a list of requirements for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String -the GUID (in XML format) of the element
GetElementResources (string ElementGUID)	protected abstract: String Notes: Gets a list of resources for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementRisks (string ElementGUID)	protected abstract: String Notes: Gets a list of risks associated with an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementScenarios (string ElementGUID)	protected abstract: String Notes: Gets a list of scenarios for an element, in XML format. Parameters: <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
GetElementTests (string ElementGUID)	protected abstract: String Notes: Gets a list of tests for an element, in XML format.


	<p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element
<p>GetFileNameDialog (string Filename, string FilterString, long FilterIndex, long Flags, string InitialDirectory, long OpenOrSave)</p>	<p>String</p> <p>Notes: Opens a standard 'File Open' or 'Save As' dialog and returns a string containing the full path to the selected file on success. Returns an empty string if the dialog was canceled.</p> <p>For example:</p> <pre>Filename = "" FilterString = "CSV Files (*.csv) *.csv All Files (*.*) *.* " Filterindex = 1 Flags = &H2 'OFN_OVERWRITEPROMPT InitialDirectory = "" OpenOrSave = 1 filepath = Project.GetFileNameDialog (Filename, FilterString, Filterindex, Flags, InitialDirectory, OpenOrSave)</pre> <p>In this example, the 'Save As' dialog will prompt for a CSV file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Filename: String - default filename specified in the dialog • FilterString: String - delimited list of available file type filters • Filterindex: Long - one-based index of the filter to be used by default • Flags: Long - additional bit flags used to initialize the file dialog; see the OPENFILENAME structure in MSDN documentation for accepted values • InitialDirectory: String - directory path to open this dialog • OpenOrSave: Long - show dialog as an 'Open' or 'Save As' style dialog; accepted values: 0 = Open, 1 = Save As
<p>GetLastError ()</p>	<p>protected abstract: String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
<p>GetLink (string LinkGUID)</p>	<p>protected abstract: String</p> <p>Notes: Gets connector details, in XML format.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • LinkGUID: String - the GUID (in XML format) of the connector to get details of
<p>GUIDtoXML (string GUID)</p>	<p>String</p> <p>Notes: Changes an internal GUID to the form used in XML.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • GUID: String - the Enterprise Architect style GUID to convert to XML format
<p>ImportDirectory (string PackageGUID, string Language, string DirectoryPath, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Imports a source code directory into the model.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to reverse engineer code into • Language: String - specifies the language of the code to be imported

	<ul style="list-style-type: none"> • DirectoryPath: String - specifies the path where the code is found on the computer • ExtraOptions: String - enables extra options to be given to the command; currently enables import of source from all child directories (recurse) - for example: recurse=1
ImportFile (string PackageGUID, string Language, string FileName, string ExtraOptions)	<p>Boolean</p> <p>Notes: Imports an individual file or binary module into the model, in a Package per namespace style import.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to reverse engineer code into; this is expected to be a namespace root Package • Language: String - specifies the language of the code to be imported Use the value 'DNPE' to import a binary module; this imports a .NET assembly or Java .class file, but not a .jar file • Filename: String - specifies the path where the code or module is found on the computer • ExtraOptions: String - enables extra options to be given to the command; currently unused
ImportPackageXMI (string PackageGUID, string Filename, long ImportDiagrams, long StripGUID)	<p>String</p> <p>Notes: Imports an XMI file at a point in the tree. Returns an empty string if successful, or returns an error message on failure.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the target Package to import the XMI file into (or overwrite with the XMI file) • Filename or XMLText: String - the name of the XMI file; if the String is of type filename it is interpreted as a source file, otherwise the String is imported as XML text • ImportDiagrams: Long - 1 for importing diagrams and 0 to skip importing diagrams • StripGUID: Long <ul style="list-style-type: none"> - 1 to replace the element UniqueIDs on import; if stripped, then a copy of the Package could be imported into the same Enterprise Architect model as two different versions - 0 to retain the element UniqueIDs on import; a duplicate copy of the Package cannot be created in the same model of Enterprise Architect
ImportReferenceData (string FileName, string DataSets)	<p>Boolean</p> <p>Notes: Imports Reference Data.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the name of the reference data file to import from • DataSets: String - the list of reference data sets to import from; the data set delimiter is ";" If the string is empty, Enterprise Architect displays a dialog prompt to select the data sets to import
ImportVisualStudioSolution (string PackageGUID, string SolutionPath)	<p>Boolean</p> <p>Notes: Imports a Visual Studio Solution into the model.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: string - the GUID (in XML format) of the Package to reverse

	<p>engineer the solution into</p> <ul style="list-style-type: none"> • SolutionPath: string - specifies the path of the Visual Studio Solution file on the computer
LayoutDiagram (string DiagramGUID, long LayoutStyle)	<p>Boolean</p> <p>Notes: Deprecated. Use LayoutDiagramEx.</p> <p>Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to lay out • LayoutStyle: Long - always ignored
LayoutDiagramEx (string DiagramGUID, long LayoutStyle, long Iterations, long LayerSpacing, long ColumnSpacing, boolean SaveToDiagram)	<p>Boolean</p> <p>Notes: Calls the function to automatically layout a diagram in hierarchical fashion. It is only recommended for Class and Object diagrams.</p> <p>LayoutStyle accepts these options</p> <ul style="list-style-type: none"> • Default Options: <ul style="list-style-type: none"> - lsDiagramDefault - lsProgramDefault • Cycle Removal Options: <ul style="list-style-type: none"> - lsCycleRemoveGreedy - lsCycleRemoveDFS • Layering Options: <ul style="list-style-type: none"> - lsLayeringLongestPathSink - lsLayeringLongestPathSource - lsLayeringOptimalLinkLength • Initialize Options: <ul style="list-style-type: none"> - lsInitializeNaive - lsInitializeDFSOut - lsInitializeDFSIn • Crossing Reduction Option: <ul style="list-style-type: none"> - lsCrossReduceAggressive • Layout Options - Direction <ul style="list-style-type: none"> - lsLayoutDirectionUp - lsLayoutDirectionDown - lsLayoutDirectionLeft - lsLayoutDirectionRight <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to lay out • LayoutStyle: Long - the layout style • Iterations: Long - the number of layout iterations the Layout process should take to perform cross reduction (Default value = 4) • LayerSpacing: Long - the per-element layer spacing the Layout process should use (Default value = 20) • ColumnSpacing: Long - the per-element column spacing the Layout process should use (Default value = 20) • SaveToDiagram: Boolean - specifies whether or not Enterprise Architect

	should save the supplied layout options as default to the diagram in question
LoadControlledPackage (string PackageGUID)	String Notes: Loads a Package that has been marked and configured as controlled. The filename details are stored in the Package control data. Parameters: <ul style="list-style-type: none"> PackageGUID: String - the GUID (in XML format) of the Package to load
LoadDiagram (string DiagramGUID)	protected abstract: Boolean Notes: Loads a diagram by its GUID. Parameter: <ul style="list-style-type: none"> DiagramGUID: String - the GUID (in XML format) of the diagram to load; if you retrieve the GUID using the Diagram interface, use the GUIDtoXML function to convert it to XML format
LoadProject (string FileName)	protected abstract: Boolean Notes: Loads an Enterprise Architect project file. Do not use this method if you have accessed the Project interface from the Repository, which has already loaded a file. Parameters: <ul style="list-style-type: none"> FileName: String - the name of the project file to load
Migrate (string GUID, string SourceType, string DestinationType)	Void Notes: Migrates a model (or part of a model) from one BPMN, ArchiMate, UPDM or SysML format to an upgraded format. Parameters: <ul style="list-style-type: none"> GUID: String - the GUID of the Package or element for which the contents are to be migrated SourceType: String - the type of model to be upgraded; accepted values: <ul style="list-style-type: none"> BPMN BPMN1.1 UPDM SysML1.1 SysML1.2 SysML1.3 ArchiMate ArchiMate2 UPDM2 DestinationType: String - the type of model to upgrade to; accepted values: <ul style="list-style-type: none"> BPMN1.1 BPMN1.1::BPEL BPMN2.0 UPDM2 SysML1.2 SysML1.3 SysML1.4 ArchiMate2 ArchiMate3 UAF
MigrateToBPMN11 (string GUID, string Type)	Void Notes: Migrates every BPMN 1.0 construct in a Package or an element (including elements, attributes, diagrams and connectors) to BPMN 1.1.

	<p>Parameters</p> <ul style="list-style-type: none"> • GUID: String - the GUID of the Package or element for which the contents are to be migrated to BPMN 1.1 • Type: String - the type of upgrade, either just to BPMN 1.1 or to BPMN 1.1 and BPEL. Accepted values are: <ul style="list-style-type: none"> - BPMN = migrate to BPMN 1.1 - BPEL = migrate to BPMN 1.1 and update: <ul style="list-style-type: none"> - any diagram with stereotype BPMN to BPEL - any element with stereotype BusinessProcess to BPELProcess <p>Migrating to BPEL is possible in the Ultimate and Unified Editions of Enterprise Architect.</p>
<p>ProjectTransfer (string SourceFilePath, string TargetFilePath, string LogFilePath)</p>	<p>Boolean</p> <p>Notes: Transfers the project from a source .eap file or DBMS to a target .eap file, .eapx file, .feap file, .qea file or .qeax file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • SourceFilePath: String - the path of the source file to transfer • TargetFilePath: String - the path of the target file, including the file type extension; Enterprise Architect creates a new Base project in this location (using the TargetFilePath as its name) and then transfers the content of the source project into that file • LogFilePath: String - the path of the log file where the status of the transfer process is updated <p>In automation, the target file must not previously exist. Enterprise Architect creates a new, empty Base.* file using the specified target name and extension, and transfers the source project into it.</p>
<p>PublishResult (string CategoryID, EA.EnumMVErrorType ErrorType, string ErrorMessage)</p>	<p>String</p> <p>Notes: Returns the results of each rule that can be performed during model validation. It must be called once for each rule from the EA_OnInitializeUserRules broadcast handler.</p> <p>The return value is a RuleId, which can be used for reference purposes when an individual rule is executed by Enterprise Architect during model validation.</p> <p>See the Model Validation Example for a detailed example of the use of this method.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • CategoryId: String - should be passed the return value from the DefineRuleCategory method • ErrorType: EA.EnumMVErrorType - depending on the severity of the error being validated, can be: <ul style="list-style-type: none"> - mvErrorCritical - mvError - mvWarning, or - mvInformation • ErrorMessage: String - contains an error string
<p>PutDiagramImageOnClipboard (string DiagramGUID, long Type)</p>	<p>protected abstract: Boolean</p> <p>Notes: Copies an image of the specified diagram to the clipboard.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to copy • Type: Long - the file type <ul style="list-style-type: none"> - If Type = 0 then it is a metafile - If Type = 1 then it is a Device Independent Bitmap

<p>PutDiagramImageToFile (string Diagram GUID, string FileName, long Type)</p>	<p>protected abstract: Boolean Notes: Saves an image of the specified diagram to file. Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID (in XML format) of the diagram to save • FileName: String - the name of the file to save the diagram into • Type: Long - the file type <ul style="list-style-type: none"> - If type = 0 then it is a metafile - If type = 1 then it uses the file type from the name extension (that is, .bmp, .jpg, .gif, .png, .tga)
<p>ReloadProject ()</p>	<p>protected abstract: Boolean Notes: Reloads the current project. This is a convenient method to refresh the current loaded project (in case of outside changes to the .eap file).</p>
<p>RunExecutableStatemachin e (string ElementGUID, string ExtraOptions)</p>	<p>Boolean Notes: Runs Executable StateMachine code for an <<executable statemachine>> Artifact element, which will start simulation of the StateMachine Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to generate • ExtraOptions: String - enables extra options to be given to the command (currently unused)
<p>RunModelSearch (string Search, string SearchTerm, bool ShowInEA)</p>	<p>Void Notes: Invokes the Model Search component. Parameters:</p> <ul style="list-style-type: none"> • Search: String - the name of an Enterprise Architect defined search • SearchTerm: String - the term to search for in the project • ShowInEA: Boolean - execute the search and output in the Model Search window
<p>RunReport (string PackageGUID, string TemplateName, string Filename)</p>	<p>protected abstract: Void Notes: Runs a named document report. Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID of the Package or master document to run the report on • TemplateName: String - the document report template to use; if the PackageGUID has a stereotype of MasterDocument, the template is not required • FileName: String - the file name and path to store the generated report; the file extension specified will determine the format of the generated document - for example, RTF, PDF
<p>RunHTMLReport (string PackageGUID, string ExportPath, string ImageFormat, string Style,</p>	<p>String Notes: Runs an HTML report (as for 'Documentation Publish as HTML' when you click on a Package in the Browser window and on the  icon). Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package or master

string Extension)	<p>document to run the report on</p> <ul style="list-style-type: none"> • ExportPath: String - the directory path to store the generated report files • ImageFormat: String - file format in which to store images - .png or .gif • Style: String - name of the web style template to apply; use <default> for the standard, system-provided template • Extension: String - file extension for generated HTML files (example: .htm)
SaveControlledPackage (string PackageGUID)	<p>String</p> <p>Notes: Saves a Package that has been configured as a controlled Package, to Native/XML format. Only the Package GUID is required, Enterprise Architect picks the rest up from the Package control information.</p> <p>Parameter:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package to save
SaveDiagramImageToFile (string Filename)	<p>protected abstract: String</p> <p>Notes: Saves a diagram image of the current diagram to file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the filename of the image to save
ShowWindow (long Show)	<p>protected abstract: Void</p> <p>Notes: Shows or hides the Enterprise Architect User Interface.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Show: Long
SynchronizeClass (string ElementGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Synchronizes a Class with the latest source code.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID (in XML format) of the element to update from code • ExtraOptions: String - enables extra options to be given to the command; currently unused
SynchronizePackage (string PackageGUID, string ExtraOptions)	<p>Boolean</p> <p>Notes: Synchronizes each Class in a Package with the latest source code.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - the GUID (in XML format) of the Package containing the elements to update from code • ExtraOptions: String - enables extra options to be given to the command; currently enables synchronization of all child Packages (children) - for example: children=1
TransformElement (string TransformName, string ElementGUID, string TargetPackage, string ExtraOptions)	<p>Boolean</p> <p>Notes: Transforms an element into a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • TransformName: String - specifies the transformation that should be executed • ElementGUID: String - the GUID (in XML format) of the element to transform • TargetPackageGUID: String - the GUID (in XML format) of the Package to transform into • ExtraOptions: String - enables extra options to be given to the command:

	<p>- GenCode=True / False - articulate code generation from the transformed elements; this option supercedes the current model setting</p>
<p>TransformPackage (string TransformName, string SourcePackage, string TargetPackage, string ExtraOptions)</p>	<p>Boolean</p> <p>Notes: Runs a transformation on the contents of a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • TransformName: String - specifies the transformation that should be executed • SourcePackageGUID: String - the GUID (in XML format) of the Package to transform • TargetPackageGUID: String - the GUID (in XML format) of the Package to transform into • ExtraOptions: String - enables extra options to be given to the command: <ul style="list-style-type: none"> - GenCode=True/False - articulate code generation from the transformed elements; this option supercedes the current model setting - SubPackages=True/False - specify if the child Packages are to be included whilst transforming a Package
<p>ValidateDiagram (string DiagramGUID)</p>	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the diagram (for correctness) and the elements and connectors within the diagram.</p> <p>Output can be viewed through 'Start > Application > Design > System Output > Model Validation'.</p> <p>Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - the GUID of the Diagram Class object
<p>ValidateElement (string ElementGUID)</p>	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the element and all child elements, diagrams, connectors, attributes and operations.</p> <p>Output can be viewed through 'Start > Application > Design > System Output > Model Validation'.</p> <p>Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - the GUID of the Element Class object
<p>ValidatePackage (string PackageGUID)</p>	<p>Boolean</p> <p>Notes: Invokes the Enterprise Architect Model Validation component, then validates the Package and all sub-Packages, elements, connectors and diagrams within it.</p> <p>Output can be viewed through 'Start > Application > Design > System Output > Model Validation'.</p> <p>Returns a Boolean value to indicate the success or failure of the process, regardless of the results of the validation.</p> <p>Parameters:</p>

	<ul style="list-style-type: none">• PackageGUID: String - the GUID of the Package Class object
XMLtoGUID (string GUID)	String Notes: Changes a GUID in XML format to the form used inside Enterprise Architect. Parameters: <ul style="list-style-type: none">• GUID: String - the XML style GUID to convert to Enterprise Architect internal format

Chart Package

The Chart interface can be used to dynamically construct any of the supported Chart types, using the functions provided in the Chart Package. The interface is obtained using the GetChart method on a Dynamic Chart element. A Dynamic Chart element can be created from the 'Charts' page of the Diagram Toolbox, and is typically used on a Dashboard diagram.

Énumérations de graphique

Ces énumérations, utilisées spécifiquement par les méthodes de l'interface Chart, sont décrites dans les rubriques de cette section. Cliquez sur le nom de l'énumération dans la liste à gauche de ce texte.

ChartAxisCrossType

Valeurs d'énumération

Enum	Value
Auto	valeur : 0
MaximumAxisValue	valeur : 1
MinimumAxisValue	valeur : 2
ValeurAxe	valeur : 3
Ignorer	valeur : 4
FixedDefaultPos	valeur : 5

ChartAxisIndex

Valeurs d'énumération

Enum	Value
Inconnu	valeur : -1
X	valeur : 0
Oui	valeur : 1
Z	valeur : 2

ChartAxisLabelType

Valeurs d'énumération

Enum	Value
Aucun libellé	valeur : 0
NextToAxis	valeur : 1
Haut	valeur : 2
Faible	valeur : 3

ChartAxisTickMarkTypeChartAxisTickMarkType

Valeurs d'énumération

Enum	Value
Pas de tiques	valeur : 0
À l'intérieur	valeur : 1
Dehors	valeur : 2
Croix	valeur : 3

Type d'axe de graphique

Ensemble de constantes faisant référence aux différents axes utilisés dans les graphiques.

Valeurs d'énumération

Enum	Value
CHART_Y_PRIMARY_AXIS	valeur : 0
CHART_Y_SECONDARY_AXIS	valeur : 1
CHART_X_PRIMARY_AXIS	valeur : 2
CHART_X_SECONDARY_AXIS	valeur : 3
CHART_Z_PRIMARY_AXIS	valeur : 4
CHART_Z_SECONDARY_AXIS	valeur : 5
CHART_Y_POLAR_AXIS	valeur : 6
CHART_X_POLAR_AXIS	valeur : 7
CHART_A_TERNARY_AXIS	valeur : 8
CHART_B_TERNARY_AXIS	valeur : 9
CHART_C_TERNARY_AXIS	valeur : 10

FormeBarreGraphique

Valeurs d'énumération

Enum	Value
Boîte	valeur : 0
Pyramide	valeur : 1
PyramidePartielle	valeur : 2

Catégorie Graphique

Valeurs d'énumération

Enum	Value
graphiqueDéfaut	valeur : 0
graphiqueLigne	valeur : 1
graphiqueCartet	valeur : 2
graphiquePie3D	valeur : 3
graphiquePyramide	valeur : 4
graphiquePyramide3D	valeur : 5
graphiqueEntonnoir	valeur : 6
graphiqueEntonnoir3D	valeur : 7
graphiqueColonne	valeur : 8
graphiqueBar	valeur : 9
graphiqueHistogramme	valeur : 10
zone de graphique	valeur : 11
graphiqueStock	valeur : 12
graphiquebulle	valeur : 13
graphiqueLongData	valeur : 14
graphiqueLigneHistorique	valeur : 15
graphiquePolaire	valeur : 16
graphiqueDonut	valeur : 17
graphiqueDonut3D	valeur : 18
graphiqueTorus3D	valeur : 19
graphiqueTernaire	valeur : 20
graphiqueColonne3D	valeur : 21

graphiqueBarre3D	valeur : 22
graphiqueLigne3D	valeur : 23
chartArea3D	valeur : 24
graphiqueSurface3D	valeur : 25
graphiqueDonutImbriqué	valeur : 26
graphiqueBoxPlot	valeur : 27
graphiqueBarSmart	valeur : 28
graphiqueBar3DSmart	valeur : 29

ChartColorMode

Valeurs d'énumération

Enum	Values
Seul	valeur : 0
Plusieurs	valeur : 1
Palette	valeur : 2
Coutume	valeur : 3
Série	valeur : 4

ChartCurveType

Valeurs d'énumération

Enum	Value
Pas de ligne	valeur : 0
Doubler	valeur : 1
Spline	valeur : 2
SplineHermite	valeur : 3
Marcher	valeur : 4
Étape inversée	valeur : 5

Style de tiret de graphique

Valeurs d'énumération

Enum	Value
Solide	valeur : 0
Se précipiter	valeur : 1
Point	valeur : 2
TiretPoint	valeur : 3
TiretPointPoint	valeur : 4
Coutume	valeur : 5

Style de cadre de graphique

Valeurs d'énumération

Enum	Value
Aucun	valeur : 0
Engrener	valeur : 1
Contour	valeur : 2
ContourMesh	valeur : 3

ChartGradientTypeChartGradientType

Valeurs d'énumération

Enum	Value
Aucun	valeur : 0
Horizontal	valeur : 1
Vertical	valeur : 2
DiagonaleGauche	valeur : 3
Diagonale Droite	valeur : 4
CenterHorizontal	valeur : 5
CentreVertical	valeur : 6
RadialTop	valeur : 7
RadialCentre	valeur : 8
Fond radial	valeur : 9
RadialGauche	valeur : 10
RadialDroite	valeur : 11
RadialHautGauche	valeur : 12
RadialHautDroite	valeur : 13
RadialBasGauche	valeur : 14
RadialBasDroite	valeur : 15
Biseau	valeur : 16
TuyauVertical	valeur : 17
TuyauHorizontal	valeur : 18

GraphiqueMarkerShape

Valeurs d'énumération

Enum	Value
Cercle	valeur : 0
Triangle	valeur : 1
Rectangle	valeur : 2
Rhombe	valeur : 3

GraphiqueStockSeriesType

Valeurs d'énumération

Enum	Value
Bar	valeur : 0
Bougie	valeur : 1
LigneOuvrir	valeur : 2
LigneHaut	valeur : 3
LigneBas	valeur : 4
LigneFermer	valeur : 5
LignePersonnalisée	valeur : 6

Type de graphique

Valeurs d'énumération

Enum	Value
type de graphiqueDEFAULT	valeur : 0
type de graphique SIMPLE	valeur : 1
type de graphiqueSTACKED	valeur : 2
chartType100STACKED	valeur : 3
type de graphique PLAGE	valeur : 4

ChartWallOptions

Valeurs d'énumération

Enum	Value
Aucun	valeur : 0, 0x0000
FillLeftWall	valeur : 1, 0x0001
ContourGaucheMur	valeur : 2, 0x0002
FillRightWall	valeur : 4, 0x0004
ContourRightWall	valeur : 8, 0x0008
RemplirPlancher	valeur : 16, 0x0010
OutlineFloor	valeur : 32, 0x0020
Dessinertout	valeur : 65535, 0xFFFF
DessinerMurGauche	FillLeftWall ContourGaucheMur
DrawRightWall	FillRightWall ContourRightWall
DessinerPlancher	RemplirPlancher OutlineFloor
DessinerTousLesMurs	DrawLeftWall DrawRightWall
OutlineAllWalls	ContourGaucheMur ContourRightWall
OutlineAll	OutlineAllWalls OutlineFloor
RemplirTousLesMurs	FillLeftWall FillRightWall
Remplir tout	RemplirTousLesMurs RemplirPlancher
Défaut	OutlineAll

Classe de graphique

La classe Chart est l'interface principale pour les éléments Chart ; il est utilisé pour créer une série, ajouter des points de données à une série et configurer l'apparence du graphique.

Attributes du graphique

Attribute	Description
Titre	String Notes : lecture/écriture Le titre du graphique.
Catégorie	CatégorieGraphique Notes : Lecture seule La catégorie graphique ; fourni dans la méthode SetChartType.
Type	Type de graphique Notes : lecture seule Le type de graphique ; fourni dans la méthode SetChartType.

Graphique Méthodes

Method	Description
AddChartDataYXZ(double Y, double X, double Z, longue sérieIndex)	long Ajoute un point de données à une série existante. Paramètres: <ul style="list-style-type: none"> • Y : double, la valeur de l'axe Y primaire • X : double, la valeur de l'axe X primaire • Z : double, la valeur de l'axe Z primaire • seriesIndex : long, l'index de la série (retourné par les méthodes CreateSeries)
AddChartDataYY1 (catégorie string , double Y, double Y1, longue sérieIndex)	long Ajoute un point de données à une série existante. Paramètres: <ul style="list-style-type: none"> • catégorie : string - le groupe, la colonne ou l'étiquette de l'axe x • Y : double, la valeur de l'axe Y primaire • Y1 : double, la valeur de l'axe Y secondaire • seriesIndex : long, l'index de la série (retourné par les méthodes CreateSeries et CreateSeriesEx)
CreateSeries(nom string)	LDISPATCH

	<p>Ajoute une nouvelle série au graphique. Renvoie une interface qui peut être utilisée pour ajouter des données à la série et configurer son apparence.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • name : string , le nom affiché de la série
CreateSeriesEx (nom string , couleur longue, type ChartType, catégorie ChartCategory)	<p>LDISPATCH</p> <p>Crée une série d'une catégorie et d'un type de graphique particulier et renvoie une interface IChartSeries. Cela permet aux graphiques de former plusieurs séries de différentes manières. Les CombinedCharts du Modèle EAExample en sont un exemple, affichant respectivement trois séries pour les catégories Area, Column et Line.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • name: string , le nom de la série • couleur : long, valeur de couleur RVB ,-1 par défaut • type : ChartType, l'une des énumérations ChartType • catégorie : ChartCategory, l'une des énumérations ChartCategory
EnableResizeAxes (booléen bEnable)	<p>annuler</p> <p>Accorde ou refuse la possibilité de redimensionner les axes.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • bEnable : booléen
GetChartAxis (type ChartAxisType)	<p>LDISPATCH</p> <p>Retourne une interface IChartAxis pour l'axe spécifié.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • type : ChartAxisType, l'une des énumérations ChartAxisType
ObtenirDiagramme3D()	<p>LDISPATCH</p> <p>Retourne une interface IChartDiagram qui peut être utilisée pour spécifier le moteur de rendu ; Logiciel ou OpenGL.</p>
GetSeries (index long)	<p>LDISPATCH</p> <p>Retourne une interface IChartSeries pour l'index donné. Les indices des séries commencent à zéro.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • indice : long
GetSeriesCount()	<p>long</p> <p>Renvoie le nombre de séries représentées par le graphique.</p>
Redessiner()	<p>annuler</p> <p>Redessine le graphique.</p>
SetChartType (type ChartType, catégorie ChartCategory, booléen bRedraw, booléen bResizeAxis)	<p>annuler</p> <p>Il s'agit généralement du premier appel effectué sur l'interface graphique. Il définit le style et l'apparence du graphique lorsqu'il est rendu.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • type : type de graphique • catégorie : Catégorie de graphique

	<ul style="list-style-type: none"> • bRedessiner : booléen • bResizeAxis : booléen
SetCurveType (type ChartCurveType)	<p>annuler</p> <p>Ensembles la méthode d'interpolation pour tracer la courbe.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • type : ChartCurveType, l'une des énumérations ChartCurveType, telles que Line, Spline ou SplineHermitte.
SetSeriesShadow(boolean bShow)	<p>annuler</p> <p>Affiche ou masque les ombres sur les séries.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • bShow : booléen
SetThemeOpacity (pourcentage long)	<p>annuler</p> <p>Ensembles l'opacité du graphique.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • pourcentage : long, transparence du graphique en pourcentage
ShowAxis (index long)	<p>annuler</p> <p>Affiche l'axe pour l'index donné.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • index : long, une des énumérations ChartAxisType
ShowDataLabels(boolean show, boolean border, boolean dropLineTomarker)	<p>annuler</p> <p>Affiche ou masque les étiquettes de données sur le graphique.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • show : booléen, afficher ou masquer les étiquettes • bordure : booléen, affiche ou masque la bordure sur les étiquettes • dropLineTomarker : booléen, modifie la position de l'étiquette par rapport à la ligne
ShowDataMarkers (spectacle booléen, taille longue, forme ChartmarkerShape)	<p>annuler</p> <p>Affiche ou masque les marqueurs de données sur le graphique. Permet également de définir l'apparence des marqueurs.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • show : booléen, afficher ou masquer les marqueurs • taille : long, taille des marqueurs en pixels • forme : ChartmarkerShape, l'une des énumérations ChartmarkerShape

Classe ChartAxisIndexChartAxisIndex Class

Attributes ChartAxisIndex

Attribute	Description
Visible	booléen Affiche ou masque l'axe.

Méthodes ChartAxisIndex

Method	Description
EnableMajorUnitIntervalInterlacing (entrelacement booléen)	annuler Active ou désactive l'entrelacement.
GetGuid()	string Renvoie le guide de l'axe. Identifie un axe de manière unique.
GetLabel()	string Renvoie la valeur du label de l'axe.
SetAxisName (étiquette string , booléen showonaxis)	annuler Ensembles le libellé de l'axe et indique s'il doit être affiché sur le graphique. Paramètres: <ul style="list-style-type: none">• label : string , le texte de l'étiquette• showonaxis : booléen, une valeur vraie indique que l'étiquette est affichée
SetCrossType (type long)	annuler Fournit une directive ou un conseil à utiliser lors du calcul de la position des étiquettes sur un axe. Paramètres: <ul style="list-style-type: none">• type : long, une des énumérations ChartAxisCrossType
SetDataFormat(format string , format booléenAsDate)	annuler Ensembles de la string de format pour la conversion des valeurs en chaînes (par exemple "%.4f"). Si les points de données représentent des valeurs datetime, l'argument formatAsDate doit être vrai et la string de format définie de manière appropriée (par exemple "%H:%M") Paramètres: <ul style="list-style-type: none">• format : string , le format à utiliser lors de la conversion des valeurs de point de données en string• formatAsDate : booléen, une valeur vraie indique que le point de données représente une date/heure

SetDisplayUnits (unités doubles)	<p>annuler</p> <p>Ensembles les unités d'affichage sur l'axe. Fondamentalement, les valeurs des points de données sont divisées par ce chiffre pour donner une valeur d'unité majeure. Par exemple, si le point de données contient des valeurs en mètres, une valeur de 1000 entraînerait l'utilisation des kilomètres comme unité principale sur l'axe.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> unités : double, la valeur d'une seule unité sur l'axe
SetFixedDisplayRange(double fmin, double fmax)	<p>annuler</p> <p>Ensembles une plage fixe pour l'axe.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> fmin : double, la valeur minimale fmax : double, la valeur maximale
SetLabelType(long labelpos)	<p>annuler</p> <p>Ensembles la position des étiquettes sur l'axe.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> labelpos : long, l'une des énumérations ChartAxisLabelType
SetTickMark(long tickmarkpos)	<p>annuler</p> <p>Ensembles la position des graduations sur l'axe.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> tickmarkpos : long, l'une des énumérations ChartAxisTickMarkType
ShowMajorGridLines (boolean show)	<p>annuler</p> <p>Affiche ou masque les lignes de la grille.</p>

Classe ChartDataValueChartDataValue Class

La classe ChartDataValue fournit une interface qui permet d'obtenir des valeurs à partir de points d'une série.

Méthodes ChartDataValue

Method	Description
ObtenirValeur()	double Renvoie la valeur associée au point de données.
IsEmpty()	booléen Vrai si aucune valeur n'existe pour le point de données.
SetEmpty(booléen vide)	annuler Ensembles un point de données sur une série pour qu'il soit vide. Paramètres: <ul style="list-style-type: none">vide : booléen, vrai si le point de données doit être considéré comme vide, n'ayant pas valeur
SetValue(double valeur)	annuler Ensembles la valeur d'un point de données. Paramètres: <ul style="list-style-type: none">valeur : double, la valeur du point de données ; définir une valeur rend un point de données non vide

Classe ChartDiagram3DChartDiagram3D Class

Méthodes ChartDiagram3D

Method	Description
SetDrawWallOptions (options longues, rafraîchissement booléen)	annuler Ensembles l'option pour la façon dont les murs et les sols - le cas échéant - sont affichés sur le graphique 3D. Le paramètre options est un masque binaire d'une ou plusieurs valeurs de l'énumération ChartWallOptions. Paramètres: <ul style="list-style-type: none">options : Long, bitmask des options d'affichage au mur et floorredraw : booléen, redessine le graphique une fois la fonction terminée
SetRenderingType (moteur long)	annuler Paramètres: <ul style="list-style-type: none">moteur : long, 0 pour logiciel, 1 pour openGL

Classe ChartFormatSeriesChartFormatSeries Class

Une classe d'assistance pour la classe ChartSeries qui permet de définir des options d'apparence.

Méthodes ChartFormatSeries

Method	Description
SetCurveType (type ChartCurveType)	annuler Ensembles l'option graphique pour le rendu des lignes. Paramètres: <ul style="list-style-type: none">• type : long, une des énumérations ChartCurveType
SetSeriesLineWidth (largeur longue)	annuler Ensembles la largeur de ligne en pixels. Paramètres: <ul style="list-style-type: none">• largeur : long, une valeur de pixel
SetSeriesOutlineDashStyle (ChartDashStyle dashstyle)	annuler Ensembles le style de tiret de la ligne sur le graphique/graphique. Paramètres: <ul style="list-style-type: none">• dashstyle : ChartDashStyle, l'une des énumérations ChartDashStyle

Classe ChartSeries

ChartSeries Méthodes

Method	Description
AddBoxPlotData(double ave, double min, double q1, double q2, double q3, double max, double encoche)	<p>long</p> <p>Pour un graphique ayant la catégorie BoxPlot, ajoute un seul point de données à la série.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • ave : double, la valeur moyenne à ce point • min : double, la valeur minimale à ce stade • q1 : double, la valeur du premier quartile • q2 : double, la valeur du deuxième quartile • q3 : double, la valeur du troisième quartile • max : double de la valeur maximale à ce stade • notched : double, pour une série de style notched, la valeur notched à exprimer à cet endroit
Ajouter un point de données (double Y)	<p>long</p> <p>Ajoute un point de données à la série. Renvoie l'index du point, qui est le nombre de points -1.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • Y : double, la valeur de l'axe Y
AddDataPoint2(double Y, double X)	<p>long</p> <p>Ajoute un point de données à la série. Renvoie l'index du point, qui est le nombre de points -1.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • Y : double, la valeur de l'axe Y • X : double, la valeur de l'axe X
AddDataPoint3(catégorie string , double Y)	<p>long</p> <p>Ajoute une valeur d'axe Y pour une catégorie donnée sur l'axe X.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • category : string , le nom de la catégorie ou de la colonne • Y : double, la valeur
AddStockData (double ouverture, double haut, double bas, double fermeture, horodatage VARIANT)	<p>annuler</p> <p>Ajoute des données à une série pour un graphique de la catégorie Stock.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • ouvert : double, valeur d'ouverture • haut : double, haute valeur • bas : double, faible valeur • fermeture : double, valeur de fermeture • timestamp : {datetime, double utcsecs} soit VARIANT date valeur soit double,

	<p>auquel cas la valeur est interprétée comme le nombre de secondes depuis minuit le 1er janvier 1970, heure UTC</p>
<p>AddSurfaceColors (VARIANTES de couleurs)</p>	<p>annuler</p> <p>Ajoute une ou plusieurs couleurs à la série.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • couleurs : long, ou long[], une seule couleur RVB ou un tableau de valeurs de couleurs RVB
<p>CloseShape (fermeture booléenne, remplissage booléen)</p>	<p>annuler</p> <p>Relie les premier et dernier points de données et remplit la forme si 'fill' est vrai.</p> <p>Paramètres</p> <ul style="list-style-type: none"> • close : booléen, si vrai ferme la série • fill : booléen, remplit la forme
<p>GetDataPointCount()</p>	<p>long</p> <p>Renvoie le nombre de points de données dans la série.</p>
<p>GetDataPointValue (index long)</p>	<p>LDISPATCH</p> <p>Renvoie une interface ChartDataValue pour le point de données avec l'index donné.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • index : long, l'index du point de données (généralement renvoyé par les fonctions AddDataPoint ; une valeur comprise entre 0 et n-1, où n est le nombre de points renvoyés par la fonction <i>GetDataPointCount</i>)
<p>GetSeriesFormat()</p>	<p>LDISPATCH</p> <p>Renvoie une interface ChartFormatSeries qui permet de modifier l'apparence du graphique.</p>
<p>SetBarShape (forme de barre longue)</p>	<p>annuler</p> <p>Ensembles la forme pour les graphiques à barres, 0 pour Box, 1 pour Pyramid, 2 pour PyramidPartial.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • barshape : ChartBarShape, l'une des énumérations ChartBarShape
<p>SetColorMapCount (nombre long)</p>	<p>annuler</p> <p>Ensembles le nombre de couleurs utilisées lors du rendu de la série. Les valeurs typiques sont 4, 8, 16 et 32</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • count : long, le nombre de couleurs à utiliser
<p>SetColorMode (mode ChartColorMode)</p>	<p>annuler</p> <p>Pour les graphiques 3D, définit la méthode d'interpolation pour le remplissage des formes. Simple, par exemple, entraînerait le remplissage de objet 3D en variant légèrement la couleur. Le niveau de variation dépendra du nombre de couleurs utilisées par le graphique (voir <i>SetColorMapCount</i>).</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • mode : ChartColorMode

SetDrawFlat(plat booléen)	<p>annuler</p> <p>Dessine la forme aplatie lorsqu'elle est définie sur true.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • plat : booléen, dessin à plat
SetFrameColor (couleur longue)	<p>annuler</p> <p>Ensembles la couleur du cadre pour les objets 3D.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • color : long, la valeur de couleur RVB pour colorer le cadre
SetFrameStyle (style ChartFrameStyle)	<p>annuler</p> <p>Ensembles le style de cadre pour le graphique - aucun, maillage, contour ou les deux.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • style : ChartFrameStyle, l'une des énumérations ChartFrameStyle
SetGradientType (type long)	<p>annuler</p> <p>Ensembles le type de dégradé à utiliser.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • type : long, une des énumérations ChartGradientType
SetGroupID (identifiant long)	<p>annuler</p> <p>Regroupe les séries sur un graphique empilé ayant le même identifiant. Doit être un nombre non négatif.</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • id : long, un nombre non négatif utilisé pour regrouper les séries sur un graphique
SetLevelRangeMode (mode long)	<p>annuler</p> <p>Ensembles le mode pour les gammes en série.</p> <ul style="list-style-type: none"> • 0 - Minimum et maximum pour la série • 1 - Minimum et maximum pour l'axe Y • 2 - Personnalisé <p>Paramètres:</p> <ul style="list-style-type: none"> • mode : long, 0 ou 1 pris en charge
SetRelatedAxis(axe string , index long)	<p>annuler</p> <p>Ensembles l'axe associé pour une série. L'axe associé est créé à l'aide de la fonction Split de l'interface ChartAxis. L'axe est d'abord créé à l'aide de Split, puis une nouvelle série est créée, et cette fonction l'appelle sur l'un de ses axes. L'axe est spécifié par le paramètre index ; la valeur est l'une des énumérations ChartAxisIndex (0 pour X, 1 pour Y ou 2 pour Z)</p> <p>Paramètres:</p> <ul style="list-style-type: none"> • axis : string , le guid de l'axe renvoyé par un appel de méthode ChartAxis.Split ; renvoyé par la méthode ChartAxis.GetGuid • index : long, une des énumérations ChartAxisIndex
SetStockSeriesType(TypeChartStockSeriesType)	<p>annuler</p>

	<p>Pour les graphiques boursiers, définit le graphique utilisé pour afficher la série.</p> <p>Paramètres:</p> <ul style="list-style-type: none">• type : ChartStockSeriesType, l'une des énumérations ChartStockSeriesType
SetWireFrame (booléen câblé)	<p>annuler</p> <p>Ensembles ou désactive l'option filaire. Lorsqu'il est défini sur true, le graphique n'est plus rendu sous la forme d'un objet solide, mais sous la forme d'un cadre composé de fils.</p> <p>Paramètres:</p> <ul style="list-style-type: none">• wired : booléen, s'affiche sous la forme d'un objet filaire si vrai

Document Generator Interface Package

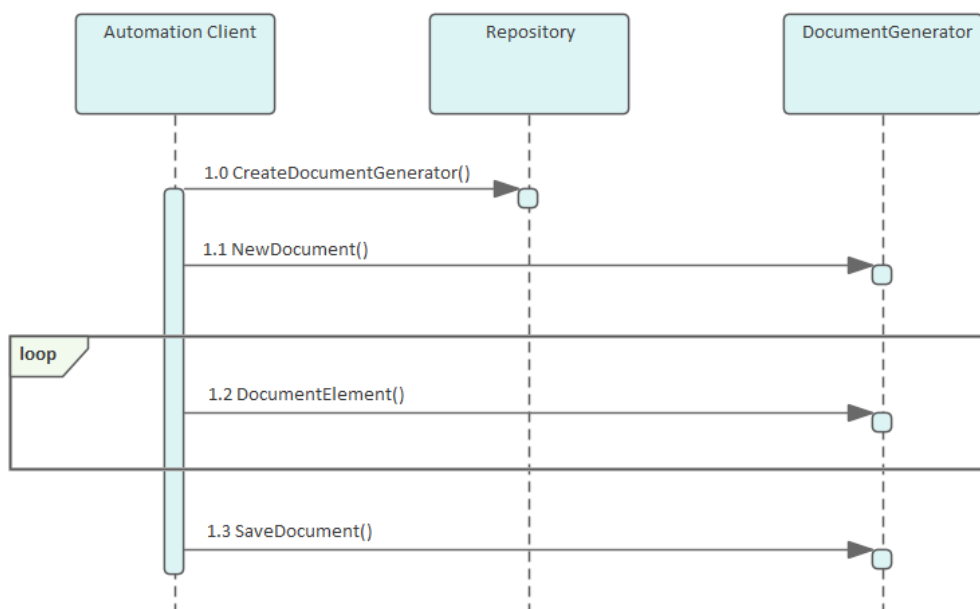
The DocumentGenerator Class provides an interface to the document and web reporting facilities, which you can use to generate reports on specific Packages, diagrams and elements in your model.

Access

Repository Class	You can create a pointer to this interface using the method Repository.CreateDocumentGenerator.
------------------	-------------------------------------------------------------------------------------------------

Example

This diagram illustrates how you might use the Document Generator interface in generating a report through the Automation Interface.



Also look at the:

- Document Generation scripting example in the Scripting window ('Specialize > Tools > Script Library', then expand the 'Local Scripts' folder and double-click on 'JScript - Documentation Example')
- RunReport method in the Project Interface

Classe DocumentGenerator

The DocumentGenerator Class provides an interface to the document and web reporting facilities, which you can use to generate reports on specific Packages, diagrams and elements in your model. This Class is accessed from the Repository Class using the CreateDocumentGenerator() method.

DocumentGenerator Attributes

Attribute	Remarks
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

DocumentGenerator Methods

Method	Remarks
DocumentConnector (long connectorID, long nDepth, string templateName)	Boolean Notes: Documents a connector. Parameters: <ul style="list-style-type: none"> connectorId: Long - the ID of the connector nDepth: Long - the depth by which to adjust the heading level templateName: String - the name of a template to use when documenting connectors; this can be blank
DocumentCustomData (string XML, long nDepth, string templateName)	Boolean Notes: Documents information based on the data supplied. Parameters: <ul style="list-style-type: none"> XML: String - the XML of the data to be documented nDepth: Long - the depth by which to adjust the heading level templateName: String - the name of a template to use when documenting custom data; this can be blank
DocumentDiagram (long diagramID, long nDepth, string templateName)	Boolean Notes: Documents a diagram. Parameters: <ul style="list-style-type: none"> diagramId: Long - the ID of the diagram nDepth: Long - the depth by which to adjust the heading level templateName: String - the name of a template to use when documenting diagrams; this can be blank
DocumentElement (long elementID, long nDepth,	Boolean

string templateName)	<p>Notes: Documents an element.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • elementId: Long - the ID of the element • nDepth: Long - the depth by which to adjust the heading level • templateName: String - the name of a template to use when documenting elements; this can be blank
DocumentModelAuthor (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a model author.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the author • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model authors; this can be blank
DocumentModelClient (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model client.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the client • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model clients; this can be blank
DocumentModelGlossary (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model glossary term.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the term • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model glossary terms; this can be blank
DocumentModelIssue (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model issue.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the issue • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model issues; this can be blank
DocumentModelResource (string name, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model resource.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the resource • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model resources; this can be blank
DocumentModelRole	<p>Boolean</p>

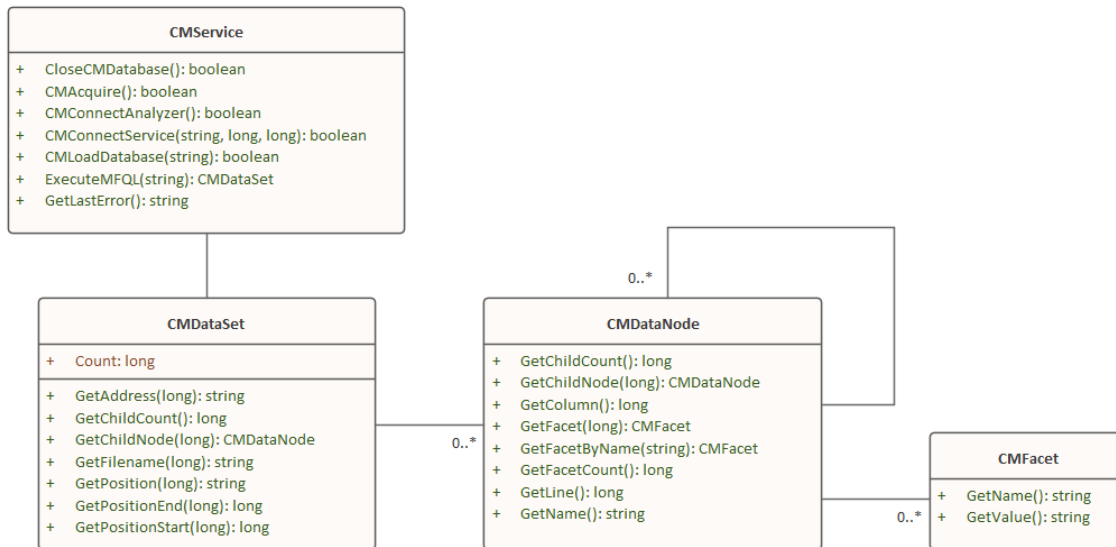
(string name, long nDepth, string templateName)	<p>Notes: Documents a single model role.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the role • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model roles; this can be blank
DocumentModelTask (long id, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a single model task.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • id: Long - the ID of the task • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting model tasks; this can be blank
DocumentPackage (long packageID, long nDepth, string templateName)	<p>Boolean</p> <p>Notes: Documents a Package.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • packageId: Long - the ID of the Package • nDepth: Long - the depth by which to adjust the heading level • templateName: String - a template to use when documenting Packages; this can be blank
GetDocumentAsRTF()	<p>Read Only.</p> <p>Returns a string value of the document in raw Rich Text Format.</p>
GetProjectConstant (string nameVal)	<p>String</p> <p>Notes: Returns the value of a Project Constant.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • nameVal: String - the name of the Project Constant for which to extract the value.
GetLastError ()	<p>String</p> <p>Notes: Returns a string value describing the most recent error that occurred in relation to this object.</p>
InsertBreak (long breakType)	<p>Boolean</p> <p>Notes: Inserts a break into the report at the current location.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • breakType: Long - 0 = page break, 1 = section break
InsertCoverPageDocument (string Name)	<p>Boolean</p> <p>Notes: Inserts the Coverpage into the document at the current location. The style sheet is applied to the document before it is insert into the generated document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the name of the Cover page document found in the Resource tree

<p>InsertHyperlink (string Name, string URL)</p>	<p>Boolean</p> <p>Notes: Inserts a hyperlink at the current location. If you use a URL with the #BOOKMARKNAME syntax, the hyperlink will link to another part of the document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • Name: String - the link text to insert into the report • URL: String - The URL of the website to link to
<p>InsertLinkedDocument (string guid)</p>	<p>Boolean</p> <p>Notes: Inserts a Linked Document into the report at the current location. A Linked Document can used to set the header and footer of the report. These are taken from the first Linked Document added to the report.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • guid: String - the GUID of the element that has a Linked Document
<p>InsertTableOfContents</p>	<p>Boolean</p> <p>Notes: Inserts a Table of Contents at the current position.</p>
<p>InsertTeamReviewPost (string path)</p>	<p>Boolean</p> <p>Notes: Inserts a Model Library posting into the report at the current location.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • path: String - the path of the Model Library post
<p>InsertTemplate (string templateName)</p>	<p>Notes: Inserts the contents of the template directly into the report.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • templateName: String - the name of the template to use
<p>InsertText (string text, string style)</p>	<p>Boolean</p> <p>Notes: Inserts static text into the report at the current location. A carriage return is not included; if you need to use one, you can add it manually.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • text: String - the static text to be inserted • style: String - the name of the style in the template; defaults to Normal style
<p>InsertTOCDocument (string name)</p>	<p>Boolean</p> <p>Notes: Inserts the Table of Contents into the document at the current location.</p> <p>Note: The stylesheet is applied to the document before it is insert into the generated document.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • name: String - the name of the Table of Contents document found in the Resource tree
<p>LoadDocument(string FileName)</p>	<p>Boolean</p> <p>Notes: Inserts an external document into the currently generated file.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FileName: String - the filename of an external document file to insert into the document.

NewDocument (string templateName)	<p>Boolean</p> <p>Notes: Starts a new document; you call this before attempting to document anything else.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>templateName</code>: String - the name of a template to use when documenting elements; this can be blank
ReplaceField (string fieldname, string fieldvalue)	<p>Boolean</p> <p>Notes: Replaces the 'Section' field identified by the <code>fieldname</code> parameter with the value provided in <code>fieldvalue</code>. For example:</p> <pre>ReplaceField ("Element.Alias", "MyAlias")</pre> <p>If you call this function more than once with the same <code>fieldname</code>, the field only has the most recent value set.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>fieldname</code>: String - the field name to find (this does not include the {} braces) • <code>fieldvalue</code>: String - the value to insert into the field; this can be a constant or a derived value
SaveDocument (string filename, long nDocType)	<p>Boolean</p> <p>Notes: Saves the document to disk.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>filename</code>: String - the filename to save the file to • <code>nDocType</code>: Long - 0 = RTF, 1 = HTML, 2 = PDF, 3 = DOCX
SetPageOrientation (long pageOrientation)	<p>Boolean</p> <p>Notes: Sets the current page orientation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>pageOrientation</code>: Long - 0 = Portrait, 1 = Landscape
SetProjectConstant (string newNameVal, string newValue)	<p>Boolean</p> <p>Notes: Sets a Project Constant for the documentation generator; this is saved in the current model.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>newNameVal</code>: String - the name of the Project Constant • <code>newValue</code>: String - the value of the Project Constant
SetStyleSheetDocument (string name)	<p>Boolean</p> <p>Notes: Sets the Stylesheet to be used for TOC, Coverpage and templates used. This can be called before NewDocument.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>name</code>: String - the name of the stylesheet found in the Resource tree
SetSuppressProfile (name)	<p>Boolean</p> <p>Notes: Sets the Suppress Profile to be used during report generation.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • <code>Name</code>: String - The name of the Suppress Profile, as created on the 'Suppress Sections' tab of the 'Document Generation' dialog.

Paquetage Code Miner

Le Paquetage Code Miner fournit l'interface d'automatisation aux éléments Code Miner . Il contient les classes suivantes :



Pour un aperçu de l'utilisation de Code Miner , consultez la rubrique d'aide *Code Miner Framework* sous Ingénierie de Logiciel .

Interface CMService

Le **CMService** est l' objet d'interface principal, utilisé pour accéder aux données Code Miner . Il vous permet de vous connecter à des fichiers locaux ainsi qu'à des services de minage de code locaux ou distants, et supporte l'exécution de requêtes mFQL sur les bases de données Code Miner .

Méthodes CMService

Méthode	Remarques
Obtenir la dernière erreur ()	Renvoie une valeur string décrivant l'erreur la plus récente survenue par rapport à cet object , ou string nulle s'il n'y a pas eu d'erreur. Type de retour : string (lecture seule)
CMLoadDatabase(nom de fichier string)	Charger une base de données Code Miner à partir d'un fichier. Renvoie True en cas de succès ; vérifie GetLastError en cas d'échec. Type de retour : booléen (lecture seule) Paramètres: <ul style="list-style-type: none"> nom de fichier : chemin complet vers le fichier de base de données Code Miner
CMConnectService(string nom d'hôte, port long, dbID long)	Connectez-vous à un service Code Miner . Renvoie True en cas de réussite ; vérifiez GetLastError en cas d'échec. Note : pour interroger toutes les bases de données du Code Miner, spécifiez dbID = -1. Type de retour : booléen (lecture seule) Paramètres: <ul style="list-style-type: none"> nom d'hôte : le nom de l'ordinateur auquel se connecter port : numéro de port que le service écoute dbID : ID de base de données à utiliser.
CMConnectAnalyzer()	Connectez-vous à la base de données Code Miner par défaut. Renvoie True en cas de réussite ; vérifiez GetLastError en cas d'échec. Type de retour : booléen (lecture seule)
CMAcquire()	Actuellement non implémenté. Type de retour : booléen
ExécuterMFQL(string Query)	Renvoie un object CMDataset en cas de succès ; vérifiez GetLastError en cas d'échec. Type de retour : CMDataset (lecture seule) Paramètre: <ul style="list-style-type: none"> requête : une string de requête mFQL
FermerCMDatabase()	Fermez la connexion au service Code Miner ou au fichier de base de données. Renvoie True en cas de réussite ; vérifie GetLastError en cas d'échec. Type de retour : booléen (lecture seule)

Interface CMDataset

Le **CMDataset** est utilisé pour renvoyer les résultats de l'exécution d'une requête mFQL sur une bibliothèque Code Miner .

Les résultats sont renvoyés sous forme de données binaires dans un **CMDataset**, qui se compose d'un tableau d'objets **CMDatasetNode**. Les objets **CMDatasetNode**, dont les détails peuvent être trouvés dans une rubrique distincte, contiennent des informations détaillées sur une « **correspondance** » pour la requête de recherche. Les informations relatives à chaque nœud **CMDataset**, telles que le nom du fichier de code source, le numéro de ligne ainsi que la position de début et de fin dans le fichier de code source, où la correspondance s'est produite, peuvent être obtenues via les fonctions **CMDataset** décrites ci-dessous.

Toutes les données du **CMDataset** sont en lecture seule.

Attributes CMDataset

Attributes	Remarques
Compter	Le nombre d'éléments dans l'ensemble de données. Type : long (lecture seule)

Méthodes CMDataset

Méthode	Remarques
Obtenir le nom du fichier (index long)	Renvoie le nom du fichier de code source dans lequel l'élément correspondant a été trouvé. Type de retour : string (lecture seule) Paramètres: <ul style="list-style-type: none"> index : spécifie la position du tableau du nœud enfant pour lequel le nom de fichier est récupéré. (index basé sur 0.)
GetPosition (index long)	Renvoie la position dans le fichier de code source à laquelle l'élément correspondant a été trouvé. La valeur renvoyée a le format suivant : « <startPosition>:<endPosition> » Type de retour : string (lecture seule) Paramètres: <ul style="list-style-type: none"> index : spécifie la position du tableau du nœud enfant pour lequel les informations de position sont récupérées. (index basé sur 0.)
GetAddress (index long)	Une base de données Code Miner se compose d'une liste de nœuds d'arbre de syntaxe abstraite, où la primary key de chaque nœud est son adresse. Cette fonction opère sur une liste de nœuds AST et renvoie la primary key (adresse) de la n-ième entrée de cette liste. Type de retour : string (lecture seule) Paramètres: <ul style="list-style-type: none"> index : spécifie la position du tableau du nœud enfant pour lequel les informations d'adresse sont récupérées. (index basé sur 0.)

GetPositionStart (index long)	<p>Renvoie la position de départ dans le fichier de code source, à laquelle l'élément correspondant a été trouvé.</p> <p>Type de retour : long (lecture seule)</p> <p>Paramètres:</p> <ul style="list-style-type: none">• index : spécifie la position du tableau du nœud enfant pour lequel les informations de position de départ sont récupérées. (index basé sur 0.)
GetPositionEnd (index long)	<p>Renvoie la position de fin dans le fichier de code source, à laquelle l'élément correspondant a été trouvé.</p> <p>Type de retour : long (lecture seule)</p> <p>Paramètres:</p> <ul style="list-style-type: none">• index : spécifie la position du tableau du nœud enfant pour lequel les informations de position de fin sont récupérées. (index basé sur 0.)
Obtenir le nombre d'enfants ()	<p>Renvoie le nombre de nœuds enfants CMDataNode dans ce CMDDataSet.</p> <p>Type de retour : long (lecture seule)</p>
GetChildNode (index long)	<p>Renvoie l' object CMDataNode à la position d'index spécifiée.</p> <p>Type de retour : CMDataNode (lecture seule)</p> <p>Paramètres:</p> <ul style="list-style-type: none">• index : spécifie la position du tableau du nœud enfant à récupérer. (index basé sur 0.)

Interface CMDataNode

Les objets **CMDataNode** contiennent des informations détaillées relatives à chaque élément correspondant à la requête mFQL exécutée.

Chaque CMDataNode contient le nom de l'élément de code source correspondant, ainsi que le numéro de ligne et le numéro de colonne dans le fichier de code source où l'élément correspondant a été trouvé. Chaque CMDataNode contient également son propre tableau d'objets CMDataNode ainsi qu'un tableau d'objets CMFacet. Les objets CMFacet sont détaillés dans une rubrique distincte.

Méthodes CMDataNode

Méthode	Remarques
Obtenir le nom ()	Renvoie le nom du CMDataNode Type de retour : string (lecture seule)
Obtenir la ligne ()	Renvoie la ligne sur laquelle ce nœud peut être trouvé. Type de retour : long (lecture seule)
Obtenir la colonne ()	Renvoie la colonne sur laquelle ce nœud peut être trouvé. Type de retour : long (lecture seule)
Obtenir le nombre d'enfants ()	Renvoie le nombre de nœuds enfants CMDataNode appartenant au CMDataNode actuel. Type de retour : long (lecture seule)
GetChildNode (index long)	Renvoie l' object CMDataNode trouvé à la n-ième position dans le tableau des nœuds enfants CMDataNode, ou NULL s'il n'est pas trouvé. Type de retour : CMDataNode (lecture seule) Paramètres: <ul style="list-style-type: none"> index : spécifie la position du tableau du nœud enfant à récupérer. (index basé sur 0)
ObtenirFacetCount()	Renvoie le nombre d'objets CMFacet qui font partie du nœud actuel. Type de retour : long (lecture seule)
GetFacet (index long)	Renvoie l' object CMFacet trouvé à la nième position dans le tableau des nœuds enfants CMFacet, ou NULL s'il n'est pas trouvé. Type de retour : CMFacet (lecture seule) Paramètres: <ul style="list-style-type: none"> index : spécifie la position du tableau du nœud de facette à récupérer. (index basé sur 0)
GetFacetByName(nom string)	Renvoie l' object CMFacet nommé, ou NULL s'il n'est pas trouvé. Type de retour : CMFacet (lecture seule) Paramètres: <ul style="list-style-type: none"> nom : spécifie le nom de la facette à trouver.

Interface CMFacet

Chaque **CMDataNode** contient un tableau d'objets **CMFacet** . Les objets **CMFacet** sont des paires de données Nom/Valeur.

Les données de nom et de valeur présentes varient en fonction du **CMDataNode** particulier auquel on accède et, bien sûr, de la requête mFQL d'origine qui a été exécutée.

Méthodes CMFacet

Méthode	Remarques
Obtenir le nom ()	Renvoie le nom du nœud CMFacet actuel. Type de retour : string (lecture seule)
Obtenir la valeur ()	Renvoie la valeur du nœud CMFacet actuel. Type de retour : string (lecture seule)

Exemple de script Code Miner

L'exemple de script suivant est conçu pour démontrer l'utilisation de toutes les méthodes disponibles pour tous les objets de l'interface API Code Miner .

Example Script

```
!INC Local Scripts.EAConstants-JScript
/*
* Script Name:      CodeMiner API Test
* Author:           Sparx Systems
* Purpose:          Exercise all of the functions available within the Code Miner API
* Date:             May 2024
*/
var q1 = "{
    byItem("OPERATION\","NAME\","LoadBinary"),\
    byItem("OPERATION\","NAMESPACE\","C2DShapeEditDlg")\
}";
function PrintFacetInfo(indent, facet, k)
{
    indent = indent + "  ";
    // print out the Name/Value data pair, for the specified CMFacet object
    Session.Output(indent + "--- Facet " + k + " Info ---");
    Session.Output(indent + "Facet name: " + facet.GetName() );
    Session.Output(indent + "Facet value: " + facet.GetValue() );
    Session.Output(" ");
}
function PrintDataNode(indent, dataNode, n, level)
{
    indent = indent + "= ";
    level++;
    // print out the details for the specified CMDataNode object.
    Session.Output(" ");
    Session.Output(indent + "DataNode " + level + " - " + n + " -----");
    Session.Output(indent + "DataNode name: " + dataNode.GetName() );
    Session.Output(indent + "Line: " + dataNode.GetLine() );
    Session.Output(indent + "Column: " + dataNode.GetColumn() );
    Session.Output(indent + "- - - - -");
    Session.Output(indent + "Facet count: " + dataNode.GetFacetCount() );
    // iterate through the array of CMFacet objects and print out their contents
    var facet as EA.CMFacet;
    var facetCount;
```



```

        facetCount = dataNode.GetFacetCount();
        var k;
        for(k=0; k < facetCount; k++)
        {
                facet = dataNode.GetFacet(k);
                PrintFacetInfo(indent, facet, k+1);
        }
// iterate through the array of CMDDataNode child objects and print out their contents
        var nextLevel = level + 1;
        Session.Output(" ");
        Session.Output(indent + "Level " + nextLevel + " DataNode count: " + dataNode.GetChildCount() );
        var childDataNodeCount;
        childDataNodeCount = dataNode.GetChildCount();
        var l;
        for(l=0; l < childDataNodeCount; l++)
        {
                childDataNode = dataNode.GetChildNode(l);
                PrintDataNode(indent, childDataNode, l+1, level);
        }
        Session.Output(indent + "-----");
}
function PrintResultSetElement(indent, resultset, i)
{
// print out the details of the specified result set element
        indent = indent + " ";
        var elementNumber = i + 1;
        Session.Output(indent + "ResultSet - Element(" + elementNumber + ")");
        Session.Output(indent + "-----");
        Session.Output(indent + " Filename : " + resultset.GetFilename(i) );
        Session.Output(indent + " Address : " + resultset.GetAddress(i) );
        Session.Output(indent + " Position : " + resultset.GetPosition(i) );
        Session.Output(indent + " Start : " + resultset.GetPositionStart(i) );
        Session.Output(indent + " End : " + resultset.GetPositionEnd(i) );
        Session.Output(indent + " Top Level DataNode count: " + resultset.GetChildCount()
);
        var childCount;
        childCount = resultset.GetChildCount();
        var level = 0;
        var dataNode as EA.CMDDataNode;
        var j;
        for(j=0; j < childCount; j++)
        {

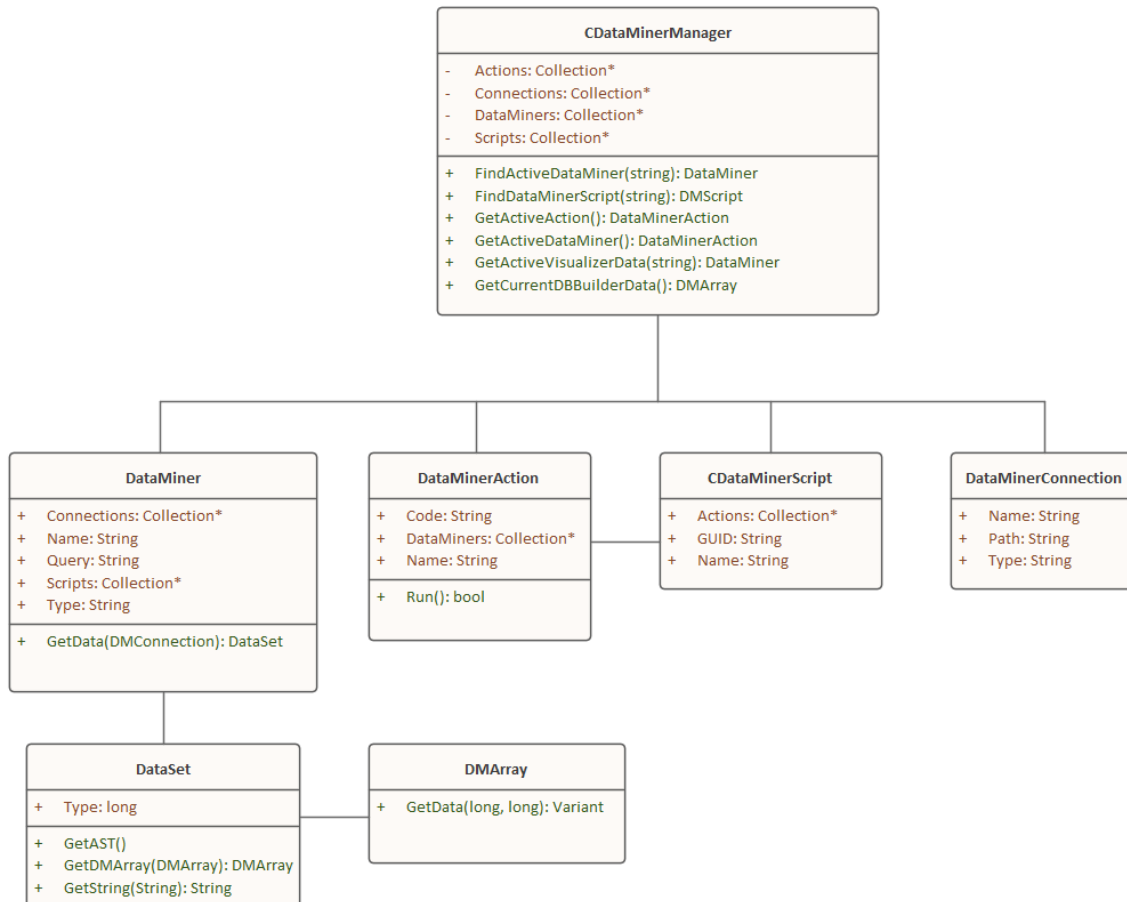
```

```
        dataNode = resultset.GetChildNode(0);
        PrintDataNode(indent, dataNode, j+1, level);
    }
    Session.Output(indent + "-----");
}
function main()
{
    var minerService as EA.CMService;
    minerService = Repository.CodeMinerService;
//    Connect to a Code Miner database
    if( minerService.CMLoadDatabase("C:\\CodeMiner Example\\Project1.cdb") )
    {
        var resultset as EA.CMDataSet;
//        Execute a query against the Code Miner database
        Session.Output("Query: " + q1);
        resultset = minerService.ExecuteMFQL(q1);
        Session.Output("Results Count: " + resultset.Count);
        Session.Output("Child count : " + resultset.GetChildCount() );
        var indent;
        indent = " ";
//        Iterate through the result set, printing out the details of each 'match' returned by the query.
        var i;
        for(i=0; i < resultset.Count; i++)
        {
            PrintResultSetElement(indent, resultset, i);
        }

        /* Out of Range Test */
        var x = resultset.GetFilename(8);
        Session.Output(indent + "Node Filename: " + x );
    }
    else
        Session.Output(indent + "Failed to Open Database: " + minerService.GetLastError() );
}
main();
```

Paquetage Data Miner

The Data Miner Package provides the Automation Interface to the Data Miner elements. It contains these Classes:



For an overview of using the Data Miner see the *Data Miner* Help topic under the *Model Exchange* group of topics.

Notes

- The Data Miner is available in the Unified and Ultimate Editions

DataMinerManager Class

DataMinerManager Attributes

Attribute	Remarks
Actions	Collection Notes: Returns a pointer to the EA.DMAction objects.
Connections	Collection Notes: Returns a Collection of EA.DMConnection objects.
DataMiners	Collection Notes: Returns a Collection of EA.DataMiner objects
Scripts	Collection Notes: Returns a Collection of EA.DMScript objects.

DataMinerManager Methods

Method	Remarks
FindActiveDataMiner (string guid)	DataMiner Object Loads the DataMiner object from the model specified by its GUID. Returns an EA.DataMiner object or NULL if the current selected object isn't a DataMiner object. Parameters: <ul style="list-style-type: none"> GUID: string - GUID of the Data Miner object to look up
FindDataMinerScript (string guid)	DMScript object Returns an EA.DMScript object in the model. Parameters: <ul style="list-style-type: none"> GUID: string - GUID of DMScript object.
GetActiveAction ()	DMAction Object When you run an Action (operation), from a diagram, this returns the Action's EA.DMAction object. Note: This is generally used for an Action to work out what DataMiner and DMConnections it is linked to.
GetActiveDataMiner ()	DataMiner Object Returns a pointer to an EA.DataMiner object, or NULL if the currently selected object is not a DataMiner object.

<p>GetActiveVisualizerData (string name)</p>	<p>DataSet Object</p> <p>Get the EA.DataSet of the currently open Visualizer.</p> <p>Parameters:</p> <ul style="list-style-type: none">• Name: string - Name of Open Visualizer <p>Note: Passing in a blank name will return the first Visualizer tab.</p>
<p>GetCurrentDBBuilderData ()</p>	<p>DMArray Object</p> <p>Get the current data from the Database Builder's latest SQL query. Returns the current output of the SQL scratch window. Accessible via:</p> <p>Ribbon: Develop > Data Modeling > Database Builder > SQL Scratch Pad.</p> <p>Return Type: DMArray</p> <p>Returns a pointer to an EA.DMArray object, or NULL if there is not a current Database Builder window with returned data.</p> <p>See The Database Builder Help topic for more information on how to get data into this window.</p>

DataMiner Class

DataMiner Attributes

Attribute	Remarks
Connections	Collection A collection of EA.DMConnection's, Notes: Read Only
Name	String Name of the Script object. Notes: Read Only
Query	String Query of the Data miner object Notes: Read Only
Scripts	Collection A collection of EA.DMScript's, Notes: Read Only
Type	String Type of the Data miner object Notes: Read Only

DataMiner Methods

Method	Remarks
GetData (DMConnection Connection)	DataSet Returns an EA.DataSet object that represents the query on the connection. Parameters: <ul style="list-style-type: none">• connection: DMConnection - A DMConnection object

DataSet Class

DataSet Attributes

Attribute	Remarks
Type	long Type of data contained in this data set. 1. Safe Array 2. Abstract Data type 3. JSon 4. Text Notes: Read Only

DataSet Methods

Method	Remarks
GetAST ()	Currently not supported
GetDMArray ()	DMArray Returns an EA.DMArray object Note: Only supported when Type = 1
GetString ()	String Returns a string of the data. NOTE: Only supported when Type = 3 or 4.

DMArray Class

DMArray Attributes

Attribute	Remarks
ColumnCount	long Notes: Read Only Number of Columns returned in this dataset
RowCount	long Notes: Read Only Number of rows returned in this dataset

DMArray Methods

GetData (long row, long column)	Variant Notes: When the database returns a NULL value, this will return an empty string. Return: Variant. Parameters: <ul style="list-style-type: none">• row: Row number of data• column: Column number of data

DMAction Class

DMAction Attributes

Attribute	Remarks
Code	String The code on the Action Notes: Read Only
DataMiners	Collection A Collection of DMDataMiner objects Notes: Read Only
Name	String Name of the Action. Notes: Read Only

DMAction Methods

Run ()	Boolean Returns TRUE if the script was run successfully.

DMScript Class

DMScript Attributes

Attribute	Remarks
Actions	Collection Returns a Collection of EA.DMAction's
GUID	String Guid of the Script object. Notes: Read Only
Name	String Name of the Script object. Notes: Read Only

DMConnection Class

DMConnection Attributes

String

Sets the type that the connect object is.

Notes: Read Only

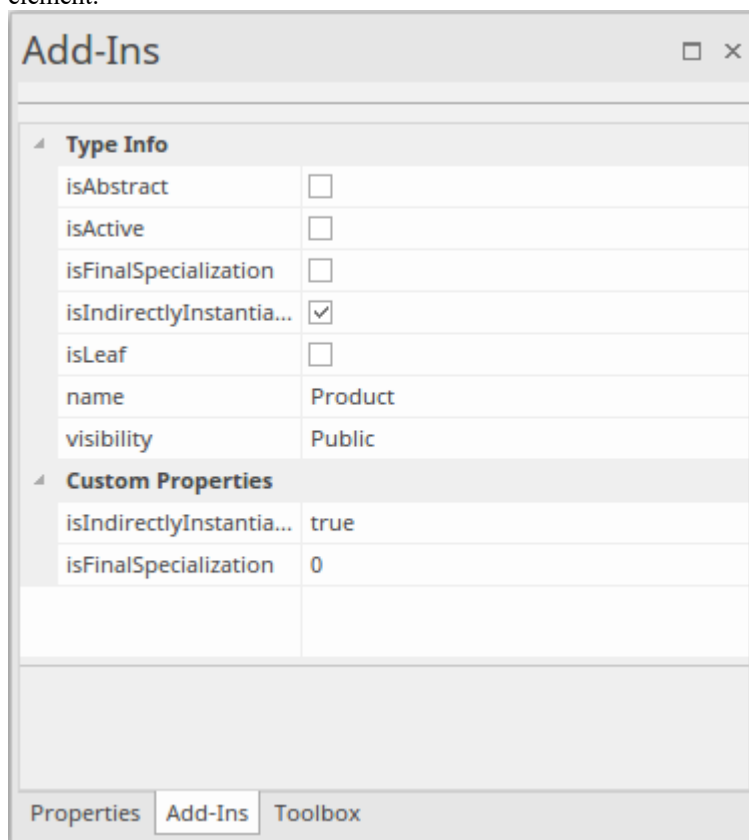
Attribute	Remarks
Name	Type: String Notes: Read Only Name of the Connection object.
Path	Type: String Path to the data we are connecting to. Notes: Read Only
Type	Type: String Notes: Read Only Type of Connection. Options: <ul style="list-style-type: none">• ODBC• EA Repository• File• URL

TypeInfoProperties Package

The TypeInfoProperties Package provides an interface to the properties of an object from the perspective of the technology rather than the Enterprise Architect database, allowing read and write access to those properties. It effectively shows the properties contained in the technology-specific and custom categories of the Properties window for the object (and omits the Enterprise Architect specific properties such as the General and Project properties). The interface hides the origin of the properties - whether they are from the base object directly, a Tagged Value, or are MOF properties.

You can see this interface in action in the EA.Example model ('Start > Help > Help > Open the Example Model'). When you open this model:

1. Select the 'Specialize > Manage Addin' ribbon option.
2. Select the checkbox against 'Type Info' and click on the OK button. An icon for 'Type Info' displays on the right of the Add-Ins panel.
3. Click on the drop-down arrow and select the 'Show Type Info' option. The Add-Ins window displays, showing the type information (properties) for the currently-selected object.
4. If you also want to display custom properties in the Add-Ins window, click on the 'Type-Info' icon again and select the 'Include Custom Properties option'. The window resembles this illustration, which is for a UML Component element.



5. Browse the EA.Example model, clicking on different types of object. You will see a different list of properties for, say, an Action than for a Class. Then you can both read and write to those properties. Also compare the list with the Properties window for the same objects.

TypeInfoProperties Class

TypeInfoProperties Attributes

Attribute	Remarks
Count	long Returns the number of TypeInfo Properties.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

TypeInfoProperties Methods

Method	Remarks
GetLastError ()	String Notes: Returns a string value describing the most recent error that occurred in relation to this object.
GetProperty (String PropName)	Returns the property value as a string. Parameters: <ul style="list-style-type: none">• PropName : String - Name of the property
HasProperty (String PropName)	Returns True if the object has the property. Parameters: <ul style="list-style-type: none">• PropName : String - Name of the property
Items (object Index)	TypeInfoProperty collection Notes: Accesses an individual TypeInfoProperty. Parameters: <ul style="list-style-type: none">• Index: Object - Either a string representing the title text or an integer representing the zero-based index of the TypeInfoProperty to get
SetProperty (String PropName, String Value)	Returns True if the property was set. Parameters: <ul style="list-style-type: none">• PropName : String - Name of property• Value : String - Value of property

TypeInfoProperty Class

TypeInfoProperty Attributes

Attribute	Remarks
Name	String Notes: Readonly. Name of the property.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Value	String Get/Sets the Property value.

TypeInfoProperty Methods

<None.>

Method	Remarks
--------	---------

Mail Interface Package

The MailInterface Package contains:

- A function to retrieve a pointer to the interface
- Functions to create and send a mail message within the current mode
- Utility functions for creating hyperlinks to selected model elements

You can get a pointer to this interface using the method `Repository.GetMailInterface`.

MailInterface Class

The MailInterface interface can be accessed from the Repository using GetMailInterface(). The returned interface provides access to the Enterprise Architect Model Mail Interface. Use this interface to automate the process of creating and sending messages using Enterprise Architect's Model Mail system.

MailInterface Attributes

Attribute	Remarks
MessagingEnabled	Boolean Notes: Read Only Advises whether messaging is enabled on the current model.
ObjectType	ObjectType Notes: Read Only Distinguishes objects referenced through a dispatch interface.

MailInterface Methods

Method	Remarks
ComposeMailMessage(string InitialRecipientGUID, string InitialSubject, messageflag InitialFlag, string InitialMessageText)	Boolean Notes: Creates a new mail message using the values specified in the input parameters; the message is displayed in the composition window, ready for sending. This method does NOT send the message. Parameters: <ul style="list-style-type: none"> InitialRecipientGUID: String - Initial value for the GUID of the addressee user (an Enterprise Architect user defined in the current model) InitialSubject: String - Initial value for the Subject text to display for this message InitialFlag: MessageFlag - Initial value for the flag type/color to attach to this message InitialMessageText: String - Initial value for the text that is the body of the message
GetAttributeHyperlink(string AttributeGUID, string LinkText)	String Notes: Returns a string containing a hyperlink to the attribute specified by the input parameter AttributeGUID. Parameters: <ul style="list-style-type: none"> AttributeGUID: String - The GUID of the attribute for which a hyperlink is required LinkText: String - The text to display for the hyperlink (such as the attribute name)

<p>GetDiagramHyperlink (string DiagramGUID, string LinkText)</p>	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the diagram specified by the input parameter DiagramGUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • DiagramGUID: String - The GUID of the diagram for which a hyperlink is required • LinkText: String - The text to display for the hyperlink (such as the diagram name)
<p>GetElementHyperlink (string ElementGUID, string LinkText)</p>	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the element specified by the input parameter ElementGUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • ElementGUID: String - The GUID of the element for which a hyperlink is required • LinkText: String - The text to display for the hyperlink (such as the element name)
<p>GetFileHyperlink (string FilePath, string LinkText)</p>	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the file specified by the input parameter FilePath.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • FilePath: String - The path name of the file for which a hyperlink is required • LinkText: String - The text to display for the hyperlink (such as the file name)
<p>GetLastError ()</p>	<p>String</p> <p>Notes: Returns the last error message set for the MailInterface.</p>
<p>GetMethodHyperlink (string MethodGUID, string LinkText)</p>	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the method specified by the input parameter MethodGUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • MethodGUID: String - The GUID of the method for which a hyperlink is required • LinkText: String - The text to display for the hyperlink (such as the method name)
<p>GetPackageHyperlink (string PackageGUID, string LinkText)</p>	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the Package specified by the input parameter PackageGUID.</p> <p>Parameters:</p> <ul style="list-style-type: none"> • PackageGUID: String - The GUID of the Package for which a hyperlink is required • LinkText: String - The text to display for the hyperlink (such as the Package name)
<p>GetRecipientGUID (string UserName)</p>	<p>String</p> <p>Notes: Returns the GUID of the specified Enterprise Architect user.</p> <p>Parameters:</p>

	<ul style="list-style-type: none">• UserName: String - The name of a user defined in the current model
GetWebHyperlink (string URL, string LinkText)	<p>String</p> <p>Notes: Returns a string containing a hyperlink to the URL specified by the input parameter URL.</p> <p>Parameters:</p> <ul style="list-style-type: none">• URL: String - The URL of the item for which a hyperlink is required• LinkText: String - The text to display for the hyperlink
SendMailMessage (string RecipientGUID, string Subject, messageflag Flag, string MessageText)	<p>Boolean</p> <p>Notes: Creates and sends a new mail message using the values specified in the input parameters.</p> <p>Parameters:</p> <ul style="list-style-type: none">• RecipientGUID: String - The GUID of the addressee user (an Enterprise Architect user defined in the current model)• Subject: String - The Subject text to display for this message• Flag: MessageFlag - The flag type/color to attach to this message• MessageText: String - The text that is the body of the message

Search Window Package

The Search Window Package contains:

- The EAContext Class, which provides a description of a single selected item
- The EASelection Class, which provides optimized functions to access information about the current selection
- The SearchWindow Class, which provides a method for displaying the results of your operation using the Search Window

EAContext Class

The EAContext Class provides a description of a single selected item. The fields with values depend on the location of the selected item.

EAContext Attributes

Attribute	Remarks
Alias	String Notes: Read only The Alias of the context item.
BaseType	String Notes: Read only Returns the base UML type of the context item.
ContextType	ContextType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
ElementGUID	String Notes: Read only The GUID of the current context object; empty if an object isn't selected. That is: <ul style="list-style-type: none">• ElementGUID if an element has context• AttributeGUID if an attribute has context• MethodGUID if an operation has context.• DiagramGUID if a diagram has context• PackageGUID if a Package has context
ElementID	Long Notes: Read only The ID of the current context object; 0 if an object isn't selected. That is: <ul style="list-style-type: none">• ElementID if an element has context• AttributeID if an attribute has context• MethodID if an operation has context.• DiagramID if a diagram has context• PackageID if a Package has context
Locked	Boolean Notes: Read only Indicates if the context item is locked.
MetaType	String Notes: Read only

	Returns the metatype of the context item.
Name	String Notes: Read only The name of the context item.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EAContext Methods

Method	Remarks
HasStereotype (String stereo)	Boolean Returns: True if the stereotype is applied to an object. Parameters <ul style="list-style-type: none">stereo: String - the stereotype to check against the context object, to see if has been applied

EASelection Class

The EASelection Class provides optimized functions to access information on the current selection. It should be used when building Add-In menus and setting the menu state, as almost all properties can be used without any database queries being made.

EASelection Attributes

Attribute	Remarks
Context	EAContext Notes: Describes the currently focused element without requiring any database calls.
ElementSet	Collection Notes: When the selection consists of one or more objects of type otElement, this provides a collection giving optimized access to all of those elements.
List	Collection Notes: For any window where multiple selection is supported, this provides a list describing the type of every selected element without requiring any database calls.
Location	String Notes: Provides the type of window that contains the current selection. Possible values are: <ul style="list-style-type: none"> • Calendar • Diagram • Dialog • Element List • Gantt • Model View • Browser window • Project View • Relationship Matrix • Reviews • Search • Specification Manager Further values could be added to this list in the future.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

EASelection Methods

None.

SearchWindow Class

The SearchWindow Class provides a method for displaying the results of your operation using the Search Window.

SearchWindow Attributes

Attribute	Remarks
FieldChooserVisible	Boolean Shows or hides the search Field Chooser.
FiltersVisible	Boolean Shows or hides the search filters.
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.

SearchWindow Methods

Method	Remarks
AddColumn (string Name, long Width)	Adds the column into the current Search window. Returns the column number, or -1 on error. Parameters: <ul style="list-style-type: none"> Name: String - Name of the column Width: Long - Width of the column
AddRow (ObjectType ot, String ElementGUID, Long ElementID, String ClassType, VARIANT Values)	Returns the row inserted into the search. Parameters: <ul style="list-style-type: none"> ot: ObjectType - the Object Type ElementGUID: String - GUID of the element ElementID: long - Object ID of the element ClassType: String - the type of object Values: an array of values
ClearGrouping ()	Clear all groupings in the search. Returns FALSE on error.
ClearSorting ()	Clear all column sorting in the search. Returns FALSE on error.
EnsureVisible ()	Make the Search window visible.

	Returns FALSE, if the Search window isn't open.
GetCell (long Row, long Column)	Returns the value of the cell. Parameters: <ul style="list-style-type: none"> • Row: long - Row number • Column: long - Column number
GroupByColumn (long Column)	Sets the group order by column. Returns FALSE if it cannot group by the specified column. Parameters: <ul style="list-style-type: none"> • Column: Long - Column number
LoadLayout (string LayoutGUID)	Set the layout of the Search window. Returns FALSE if the layout cannot be set. Parameters: <ul style="list-style-type: none"> • LayoutGUID: String - Layout GUID
NewLayout (string LayoutGUID)	Saves the layout of the Search window. Parameters: <ul style="list-style-type: none"> • LayoutGUID: String - Layout GUID
SetCellString (long Row, long Column, String Data)	Sets a value in a cell. Parameters: <ul style="list-style-type: none"> • Row: long - Row number • Column: long - Column number • Data: String - Value to set the cell to
SetCellVariant (long Row, long Column, VARIANT Data)	Sets an alternative value in a cell. Parameters: <ul style="list-style-type: none"> • Row : long - Row number • Column : long - Column number • Data: Value to set the cell to
SortByColumn (long Column)	Sets the column to sort by. Returns FALSE if it cannot sort by the specified column. Parameters: <ul style="list-style-type: none"> • Column: Long - Column number

Simulation Package

The Simulation Package contains:

- An attribute to set, increase and decrease the speed of the simulation
- A function to check if a simulation is currently running
- Functions to Start, Stop, Step Into, Step Out of, Step Over and Pause a simulation
- A function to send a broadcast signal to the simulation that is currently running

Simulation Class

The Simulation Class provides an interface to the Enterprise Architect Model Simulation facilities.

Simulation Attributes

Attribute	Description
ObjectType	ObjectType Notes: Read only Distinguishes objects referenced through a Dispatch interface.
Speed	Long Notes: Read/Write Retrieve or set the current simulation running speed.

Simulation Methods

Method	Description
BroadcastSignal(string sSignalName, string sParameters)	Boolean Notes: Send a signal into the running simulation. If the simulation is stopped, do nothing. Parameters: <ul style="list-style-type: none"> sSignalName: String - the name of the signal OR the GUID of the Signal element sParameters: String - a string of one or more signal parameters, in this format: {parameter1: 5, parameter2: "test", parameter3: 3.2}
IsSimulatorRunning()	Boolean Notes: Check the state of the simulation. Returns True if the simulation is running; returns False if the simulation is stopped.
Pause()	Boolean Notes: Pause the simulation if it is running.
Start()	Boolean Notes: Start the simulation based on the current selection. If the current simulation is in a paused state, then the simulation is resumed.
StepIn()	Boolean Notes: Step In to the routine in the current simulation.
StepOut()	Boolean

	Notes: Step Out of the routine in the current simulation.
StepOver()	Boolean Notes: Step Over the routine in the current simulation.
Stop()	Boolean Notes: Stop the simulation.

Compositeur de Schéma Paquetage

The Schema Composer can be accessed from the Enterprise Architect automation interface. A client (script or Add-In) can obtain access to the interface using the SchemaComposer property of the Repository object. This interface is available when a Schema Composer has a profile loaded.

SchemaProperty Class

SchemaProperty Attributes

Attribute	Description
TypeID	long Notes: Read only The classifier ID of the property.
PropID	long Notes: Read only The property ID.
Guid	string Notes: Read only The unique model GUID of the property.
Name	string Notes: Read only The name of the property.
Cardinality	string Notes: Read only The cardinality of the element.
UMLType	string Notes: Read only The UML type, such as attribute, association or aggregation.
Parent	long Notes: Read only The classifier of the owner Class.
PrimitiveType	string Notes: Read only The property's primitive type if property represents a simple type.
Annotation	string Notes: Read only The model notes for the property.
Stereotype	string Notes: Read only The stereotype of the property.

Choices	<p>SchemaTypeEnum</p> <p>Returns an iterator allowing navigation of choice elements in <i>model</i>, defined for this property in the Schema Composer. Combine with SchemaChoices attribute to obtain all available choices.</p>
SchemaChoices	<p>SchemaTypeEnum</p> <p>Returns an iterator allowing navigation of choice elements in <i>schema</i>, defined for this property in the Schema Composer. Combine with Choices attribute to obtain all available choices.</p>
TypeName	<p>string</p> <p>Returns a string naming the type of the property</p>
Type	<p>SchemaType</p> <p>Returns an interface to the property's type for complex types.</p>

SchemaProperty Methods

Method	Description
IsInline	<p>Boolean</p> <p>If true, the property is marked as 'Inline'. XML schema generators would emit an inline definition when detecting this attribute.</p>
IsPrimitive	<p>Boolean</p> <p>Returns true for a property whose type is maps to a built in type such as xs:integer, xs:string, xs:date or other XML Schema built-in type.</p>
IsByReference	<p>Boolean</p> <p>Returns true for a property marked as 'By Reference' in the profile.</p>

SchemaProfile Class

The interface representing the technology governing the naming and design rules on which the schema is built.

SchemaProfile Methods

Method	Description
AddExportFormat(string description)	<p>void</p> <p>Notes: Use this function to add entries that are offered by the Schema Composer when the user clicks on the Generate button.</p> <p>Parameters:</p> <ul style="list-style-type: none">description: describes the export format provided by the Add-In
SetCapability(string name,boolean enabled)	<p>void</p> <p>Notes: Use this function to enable/disable capabilities.</p> <p>Parameters:</p> <ul style="list-style-type: none">name: name of the capabilityenabled: True or False <p>Capabilities:</p> <p>'allowCardinality' - allows/denies restrictions to cardinality</p> <p>'allowRootElement' - allows/denies setting root element</p> <p>'allowPropByRef' - allows/denies By Reference restriction</p> <p>'allowRedefine' - allows/denies ability to redefine an element</p>
SetProperty(string name, string value)	<p>void</p> <p>Notes: Sets properties displayed in the Schema Composer.</p> <p>Parameters:</p> <ul style="list-style-type: none">name: property namevalue: property value <p>Properties:</p> <p>'Namespace' - Target namespace for XML schema</p> <p>'Namespace Prefix' - Namespace prefix for XML schema</p> <p>'Qualifier' - string qualifier that prepends schema type names</p>

SchemaComposer Class

The SchemaComposer Class provides the interface to the Enterprise Architect Schema Composer facility.

SchemaComposer Attributes

Attribute	Description
ModelReference	String Notes: The model ref listed in the Schema Composer for the current profile.
Namespace	String Notes: The namespace listed in the Schema Composer for the current profile.
NamespacePrefix	String Notes: The namespace prefix listed in the Schema Composer for the current profile.
TargetDirectory	String Notes: The target directory selected by the user after clicking on the Generate button.
SchemaName	String Notes: Returns the name of the schema profile currently being generated.
SchemaSet	String Notes: Returns the schema set used when the schema was created.
SchemaType	String Notes: The schema type listed in the Schema Composer for the current profile, either 'schema' or 'transform'.
SchemaTypes	SchemaTypeEnum Notes: Read only Enumerator for the type collection represented in the currently open schema.
Namespaces	SchemaNamespaceEnum Notes: Read only Enumerator for the namespaces referenced by schema

SchemaComposer Methods

Method	Description
FindInSchema(long	SchemaType

typeID)	<p>Notes: Obtains an interface to a Class as represented in the schema for a given model Class ID.</p> <p>Parameters:</p> <ul style="list-style-type: none">• typeID: the model Class ID
FindInModel(long typeID)	<p>ModelType</p> <p>Notes: Obtains an interface to a Class as represented in the UML model for a given model Class ID</p> <p>Parameters:</p> <ul style="list-style-type: none">• typeID: the model Class ID
FindSchemaTypeByName(string typename)	<p>SchemaType</p> <p>Notes: Returns an interface to the schema type that matches the type specified or null if no type exists.</p> <p>Parameters:</p> <ul style="list-style-type: none">• name : the name of the type
GetNamespacePrefixForType(long typeID)	<p>String</p> <p>Notes: Returns the schema namespace prefix for a given type</p> <p>Parameters:</p> <ul style="list-style-type: none">• typeID: the model Class ID
GetNamespaceForPrefix(string prefix)	<p>String</p> <p>Notes: Returns the URI for a given schema namespace prefix</p> <p>Parameters:</p> <ul style="list-style-type: none">• name: the namespace prefix

ModelTypeEnum Class

An enumerator interface for schema types as represented in the UML model.

ModelTypeEnum Methods

Method	Description
GetCount()	long Returns the number of types present in the collection.
GetFirst()	ModelType Returns the first type interface in a collection of types.
GetNext()	ModelType Returns the next type in the collection of types or null if end is reached.

ModelType Class

Provides an interface to the Class of a schema type as represented in the model.

ModelType Attributes

Attribute	Description
PropertyCount	long Notes: Read only The total number of properties for this Class available in the Properties collection.
Properties	SchemaPropEnum Notes: Enumerator Collection of properties for the Class as defined in the model.
TypeID	long Notes: Read only The Class ID of the type.
Guid	string Notes: Read only A GUID that uniquely identifies a type in the model.
Typename	string Notes: Read only The name of the type as represented in the model.
ClassifierPath	string Notes: Read only The qualified path of the type in the model.
ClassifierPathID	string Notes: Read only A GUID that uniquely identifies a ClassifierPath in the model.
Stereotype	string Notes: Read only The stereotype of the Class as defined in the model.
Annotation	string Notes: Read only Any notes present in the model describing the Class.

ModelType Methods

Method	Description
GetSuperClassEnum(SearchType searchtype)	<p>ModelTypeEnum</p> <p>Notes: Enumerator</p> <p>Returns an enumerator that can be used to traverse the Class ancestry.</p> <p>Parameters:</p> <ul style="list-style-type: none">• searchtype: the type of traversal to use, breadth first or depth first
GetSubClassEnum(SearchType searchType)	<p>ModelTypeEnum</p> <p>Notes: Enumerator</p> <p>Returns an enumerator that can be used to iterate over any descendents of the Class.</p> <p>Parameters:</p> <ul style="list-style-type: none">• searchtype: the type of traversal to use, breadth first or depth first
IsEnumeration	True where type represents an enumeration element

SchemaTypeEnum Class

An enumerator interface for schema types as represented in XML schema.

Methods

Method	Description
GetCount()	Returns the number of properties for an element.
GetFirst()	Returns the first property for the element in alphabetical order.
GetNext()	Returns the first property for the element in alphabetical order or null if no more are present.

SchemaType Class

Represents a type as it is defined in the schema.

Methods

Method	Description
GetFacet(BSTR name)	Returns the value of the named facet. 'Root', for example' returns a value indicating whether a type is a root element.
GetRestriction(BSTR guid)	Returns the restriction as a string for the property having the supplied guid.
IsRoot()	True if Class is marked as 'root' in the Composer.
IsEnumeration()	True if the type represents an enumeration element

Properties

Property	Description
PropertyCount [type: long]	Returns the number of properties held by 'type'.
Properties [type: IEASchemaPropEnum]	Returns an enumerator for 'type's' properties.
TypeID	The model Class ID.
Guid	The unique model GUID of the type.
Typename	The type's name.
Parent	The parent type - if any - that this Class extends. Could be null depending on composition method.

SchemaPropEnum Class

An enumerator for properties of a UML model type or XML schema type.

Methods

Method	Description
GetCount()	Returns the number of properties for an element.
GetFirst()	Returns the first property for the element in alphabetical order.
GetNext()	Returns the first property for the element in alphabetical order or null if no more are present.

SearchType Enumeration

SearchType Attributes

Attribute	Description
searchDepthFirst	Navigate children before siblings.
searchBreadthFirst	Navigate siblings before children.

SchemaNamespace Class

An interface presenting namespace information

SchemaNamespace Attributes

Name	string Notes: Read only The namespace prefix.
URI	string Notes: Read only The URI of the namespace.

SchemaNamespaceEnum Class

An enumerator interface for namespaces referenced by schema.

SchemaNamespaceEnum Methods

GetFirst()	SchemaNamespace Returns the first namespace interface in a collection of namespaces.
GetNext()	SchemaNamespace Returns the next namespace interface in a collection of namespaces

Exemples de code

As you write or edit code for using the Automation Interface, you might want to review these public Object examples, written in VB.Net.

Examples

Name
Open the Repository
Iterate Through a .cap File
Add and Manage Packages
Add and Manage Elements
Add a Connector
Add and Manage Diagrams
Add and Delete Features
Element Extras
Repository Extras
Stereotypes
Work with Attributes
Work with Methods

Open the Repository

This is an example of the VB.Net code to open an Enterprise Architect repository.

```
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
```

Iterate Through a .EAP File

This is an example of the VB.Net code to iterate through a .eap file starting at the Model level, after the repository has been opened.

```
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
```

Add and Manage Packages

This example illustrates how to add a model or a Package to the project.

Sub TestPackageLifecycle

Dim idx as integer

Dim idx2 as integer

Dim package as object

Dim model as object

Dim o as object

"first add a new Model

model = m_Repository.Models.AddNew("AdvancedModel", "")

If not model.Update() Then

 Console.WriteLine(model.GetLastError())

End If

"refresh the models collection

m_Repository.Models.Refresh

"now work through models collection and add a Package

For idx = 0 to m_Repository.Models.Count - 1

 o = m_Repository.Models.GetAt(idx)

 Console.WriteLine(o.Name)

 If o.Name = "AdvancedModel" Then

 package = o.Packages.Addnew("Subpackage", "Nothing")

 If not package.Update() Then

 Console.WriteLine(package.GetLastError())

 End If

 package.Element.Stereotype = "system"

 package.Update

 "for testing purposes just delete the

 "newly created Model and its contents

 m_Repository.Models.Delete(idx)

End If

Next

End Sub

Add and Manage Elements

This is an example of the code for adding and deleting 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 this example, Enterprise Architect loads the
```

```
    "element required every time 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
```

Add a Connector

This is an example of code to add a connector and set its 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 ElementIDs to quickly load an element in this example
```

```
    "... you must find suitable IDs 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

Add and Manage Diagrams

This is an example of the code for creating a diagram and adding an element to it. Note the optional use of the element rectangle setting, using left, right, top and bottom dimensions in the 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
```

Add and Delete Features

An example of code to add and delete Features of an object.

```
Dim element as object
```

```
Dim idx as integer
```

```
Dim attribute as object
```

```
Dim method as object
```

```
'just load an element by ID - you must
```

```
'substitute a valid ID from your model
```

```
element = m_Repository.GetElementByID(246)
```

```
"create a new method
```

```
method = element.Methods.AddNew("newMethod", "int")
```

```
method.Update
```

```
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
```

Element Extras

These are examples of code 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
    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

Repository Extras

These are examples of code for accessing 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)
    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
```

Stereotypes

This is some example code for adding and deleting stereotypes.

```
Sub TestStereotypes
```

```
    Dim o as object
```

```
    Dim idx as integer
```

```
    "add a new stereotype to the Stereotypes collection
```

```
    o = m_Repository.Stereotypes.AddNew("funky","class")
```

```
    If(o.Update=false) Then
```

```
        Console.WriteLine (o.GetLastError())
```

```
    End if
```

```
    o = nothing
```

```
    "make sure you refresh
```

```
    m_Repository.Stereotypes.Refresh
```

```
    "then iterate through - deleting our new entry in the process
```

```
    For idx = 0 to m_Repository.Stereotypes.Count-1
```

```
        Console.WriteLine(m_Repository.Stereotypes.GetAt(idx).Name)
```

```
        If(m_Repository.Stereotypes.GetAt(idx).Name = "funky") then
```

```
            m_Repository.Stereotypes.DeleteAt(idx,false)
```

```
            Console.WriteLine("Delete element")
```

```
        End if
```

```
    Next
```

```
End Sub
```

Work With Attributes

This is an example of code for 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
```

Work With Methods

This is an example of code for 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
```

