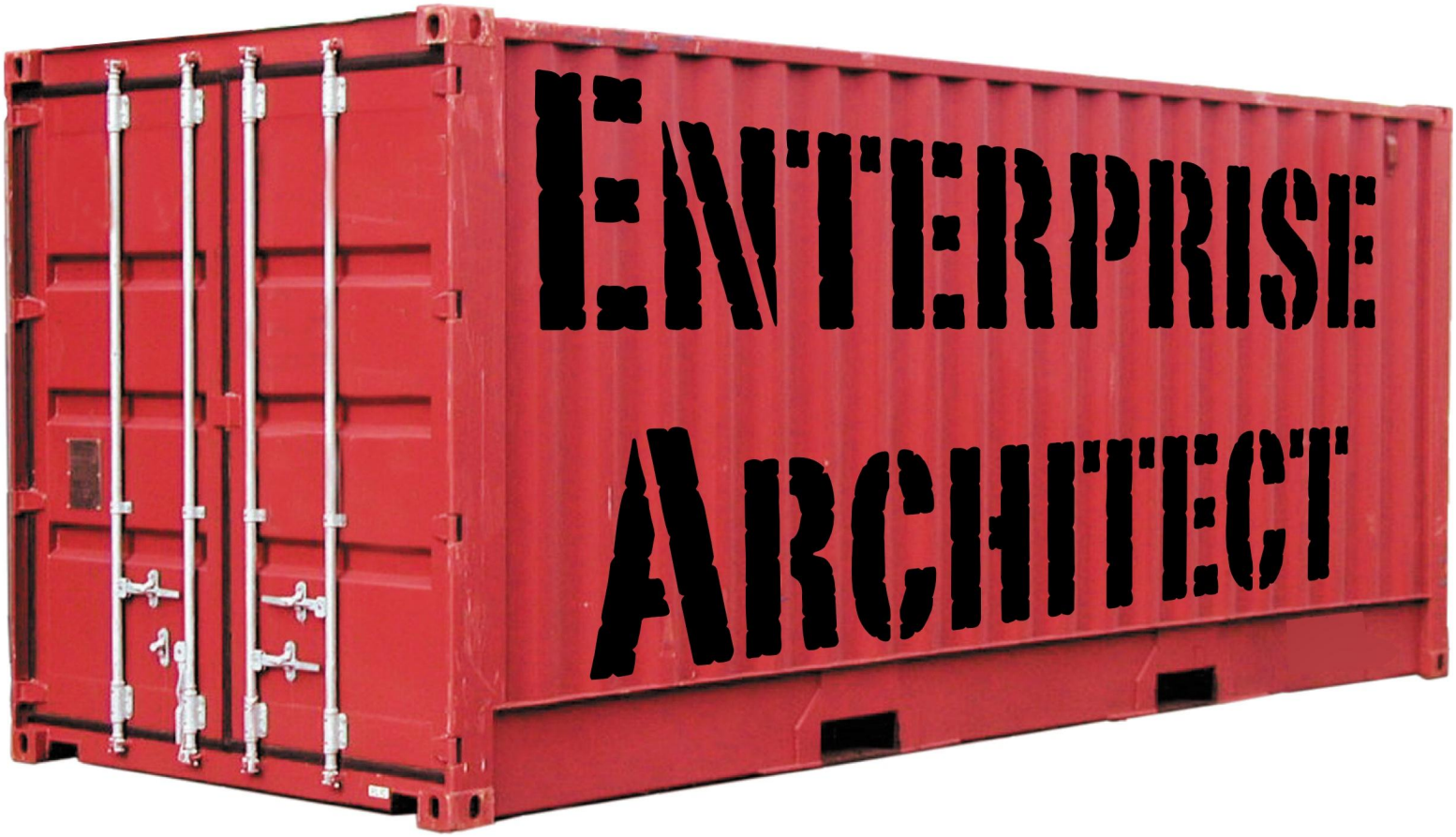
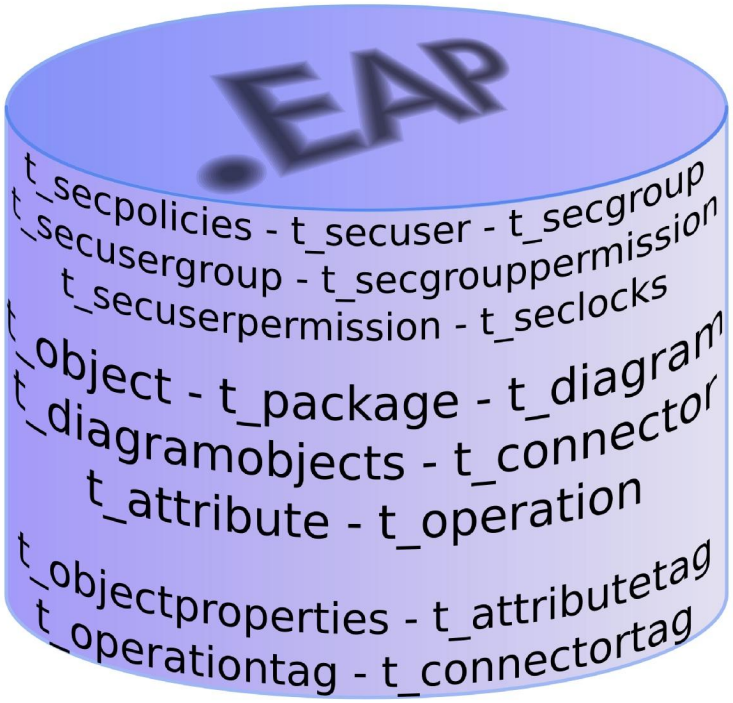
Inside



Querying EA's Database



By Thomas Kilian

Inside Enterprise Architect

Querying EA's Database

Thomas Kilian

This book is for sale at http://leanpub.com/InsideEA

This version was published on 2023-04-29

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

1. Preface

Enterprise Architect[1] (EA) offers a wealth of API functions to support automated manipulation of UML models. However, quite a number of tasks require actions not directly supported by the API. Here the fact comes handy that EA is based on a database model which has proven to be very stable with respect to its structure. The last major change was introduced with audit functionality which added a couple of new tables but left the structure of the existing tables untouched. So you can assume that your add-ins will run in future versions of EA if you follow a few rules.

The contents of this book is the essence of a continuous work with EA since end 2003. It surely lacks prose but likely you won't need that anyway. I'd call it a hacker's guide into EA[2].

Special thanks to Peter Doomen who inspired me to write this book. You likely might be interested in his book Fifty Enterprise Architect Tricks[3]. Also I like to thank Helmut Ortmann for supplying me with most of the query examples and a couple of hints which had passed my attention[4]. Probably I should mention a couple of other guys[5] but I'm not going to bother you with my family history.

This book starts with a short introduction on how to query EA's database. This is followed by a concise list of all available tables and details for the most important ones. The details contain cross references into more details as well as screen shots of the GUI where the appropriate elements appear. Vice versa the screen shots point to the according table columns. The final sections conclude with a practical approach to using SQL in Enterprise Architect.

[1] The EA version used to create this book was actually 9.3 (build 930). However, most of the references are also valid for earlier versions of EA.

[2] Not all tables/columns are clear in their meaning (to me). A ?! mark is placed where this is the case. Comments about clarification of their meaning are welcome! Just send me a mail to thomas.kilian@me.com.

[3] http://leanpub.com/entarch

[4] The latest finding by Colin Wood has been fixed December 2017. Thanks for notifying me.

[5] Cheers to Paolo and all the supporters at Sparx.

2. Copyright and Disclaimer

Also all of the information in this book has been tested by me in many circumstances I can not hold any liability for use of the here presented information[1]. However, I'd be glad to receive any kind of feedback to correct future updates of this book which you will receive for free in turn. Having said this, all information presented here is subject to change without notice.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Important note: this book is about **querying** EA's database. You might think that updating the database is easy with an UPDATE statement. Sure. But that's playing with a loaded and unsecured weapon! You might shoot yourself in your knee or even in the middle of your heart. If you are going to change your repository: use the API.

[1] I really loathe writing such legal blurb since it should be obvious. By the way: German Law applies! (Does that change anything?)

3. Accessing the Database

The lowest layer in EA is that of its database. When you first start with EA you will most likely deal with EAP files. A simple though not official fact is: EAP is MS Access. So if you want to play around just open one of those EAP files and see what MS Access is telling you. If you are using a Corporate license you will most likely use a more advanced SQL server. Be it MS SQL Server, Oracle, MySQL or whatever. In that case you need some client software to perform manipulations.

Before doing so you should get familiar with the database in a more simple way.

3.1 Inspecting EA's Tables

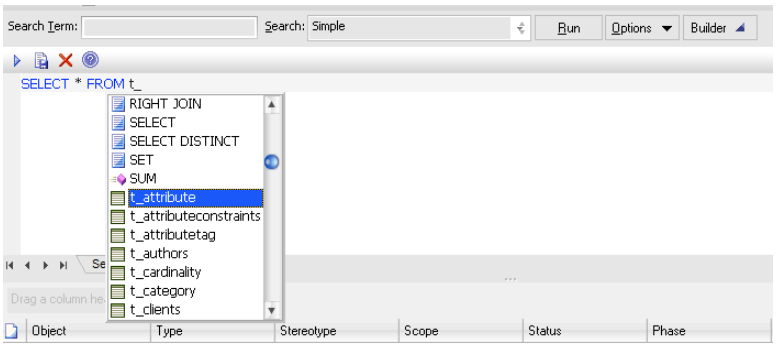
The most simple way is to open the respective EA repository with EA itself.

Access seems to be picky with some columns and suppresses them. So preferably you use EA like explained here to actually inspect the database.

Use the EAExample.EAP which comes with your EA installation. Now press `Ctrl-F` to open the search window. Click the Builder button and select the SQL tab. Type the text

```
1          SELECT * FROM t_
```

and press Ctrl-Blank. You should now be presented with the following:



Search Window

As you can see this is a list of tables which reside in EA's database. For a start let's choose one of the important tables: `t_object`. After pressing the Run button or the little triangle top left you will get a list of all elements present in the repository. Obviously the column `Object_Type` is the type of an element and Name its name. Simple! You might go on with `t_package` to see details of all packages in the repository.

3.2 Ways to Query Tables

The clean way to query the tables is the API. This is recommended for most cases. However, there's also a demand to be able to access these data where the API is simply too slow. EA is kind of object oriented in how it queries its database. That means for a global change (like changing the status) it will issue single UPDATEs instead of a compound. Of course you can query the database much faster with an intelligent query than any iterative API calls.

Anticipating a simple structure element of the table containing the elements (`t_object`) the best way to retrieve an element is by

```
1         elem = Repository.GetElementByID(4711);
```

which yields the 'same' element as

```
1         SELECT * FROM t_object WHERE Object_ID = 4711
```

where just the API object data are cooked while those of the SQL are raw. So that's no wizardry. But imagine you need all elements with a certain stereotype. While in the API this would need quite some programming effort, the SQL is simple:

```
1         SELECT * FROM t_object WHERE stereotype = "stereo"
```

Now it's upon your imagination what you can do by joining different tables in SQL. A few sample can be found in this chapter at the end of this book.

Note:

This window accepts only *SELECT* statements. Any other SQL (like e.g. *UPDATE*) is silently ignored!

3.3 A List of All Tables

Name	Description
t_attribute	Attributes defined for elements
t_attributeconstraints	Constraints for attributes
t_attributetag	Tagged values for attributes
t_authors	List of authors defined with `Settings/Project Types/People/Project Author(s)`
t_cardinality	List of cardinalities defined with `Settings/UML Types/Cardinality Value`
t_category	Legacy ?!
t_clients	List of authors defined with `Settings/Project Types/People/Project Clients`
t_complexitytypes	Legacy ?!
t_connector	Connectors between elements

Name	Description
t_connectorconstraint	Constraints for connectors
t_connectortag	Tagged values for connectors
t_connectortypes	This table is used for the connector metatypes shown in the profile dialog. Also t_connector.Connector_Type can contain just elements from this table.
t_constants	Various key/value pairs in misc. dialogues
t_constrainttypes	List of constraint types defined with `Settings/Project Types/General/Constraint`
t_datatypes	Definitions from `Settings/Code Datatypes`
t_diagram	Diagram properties
t_diagramlinks	Non-standrad links appearing in diagrams
t_diagramobjects	Diagram elements
t_diagramtypes	Legacy ?!
t_document	Contents of linked documents, baselines and more
t_ecf	List of complexity factors defined with `Settings/Project Types/Estimation Factors/Environment...`
t_efforttypes	List of effort factors defined with `Settings/Project Types/Project Indicators/Effort`
t_files	?!
t_genopt	Various options
t_glossary	The system glossary
t_html	Some HTLM strings for the doc generation
t_image	The alternate pictures defined with `Settings/Images...`
t_implement	Legacy ?!
t_issues	The system issues defined with `View/More Project Tools/Project Information/Issues`
t_lists	Status Types defined with various `Settings/Project Types/General Types/...` tabs
t_mainttypes	?!
t_method	Legacy ?!
t_metrictypes	List of metric factors defined with `Settings/Project Types/Project Indicators/Metric`
t_object	Basic UML elements
t_objectconstraint	Constraints for elements
t_objecteffort	Something related to project estimation
t_objectfiles	The files linked in the properties/files for elements
t_objectmetrics	Something related to project estimation
t_objectproblems	?!
t_objectproperties	Tagged values
t_objectrequires	The internal requirements defined for elements
t_objectresource	Something related to project estimation
t_objectrisks	Something related to project estimation
t_objectscenarios	The use case scenarios
t_objecttests	Tests defined for elements
t_objecttrx	?!
t_objecttypes	EA internal rendering support
t_ocf	Something related to project estimation
t_operation	Operations for elements
t_operationparams	Parameters for operations

Name	Description
t_operationposts	Postconditions for operations
t_operationpres	Preconditions for operations
t_operationtag	Tagged values for operations
t_package	Package container
t_palette	Legacy ?!
t_paletteitem	Legacy ?!
t_phase	Legacy ?!
t_primitives	Code primitive types
t_problemtypes	Problem types defined with `Settings/Project Types/Maintenance/Problem Types`
t_projectroles	Values from `Settings/People/Project Roles`
t_propertytypes	Predefined tagged values
t_requiretypes	List of requirement types defined with `Settings/Project Types/General/Requirements`
t_resources	List of resource defined with `Settings/Project Types/People/Resources`
t_risktypes	List of risk factors defined with `Settings/Project Types/Project Indicators/Risk`
t_roleconstraint	Legacy ?!
t_rtf	EA internal doc generation settings
t_rtfreport	Some doc generation settings
t_rules	Related to model validation ?!
t_scenariotypes	List of scenario types defined with `Settings/Project Types/General/Scenario`
t_script	Local scripts
t_secgroup	Security groups
t_secgrouppermission	Security group permissions
t_seclocks	Security locks
t_secpermission	?!
t_secpolicies	Security settings
t_secuser	Security users
t_secusergroup	Security user/group assignments
t_secuserpermission	Security user permissions
t_snapshot	Audit log
t_statustypes	List of status types defined with `Settings/Project Types/General/Status`
t_stereotypes	Stereotypes
t_taggedvalue	Smorgasbord for WSDL model elements. These are NOT the tagged values. Instead use t_objectproperties
t_tasks	The system tasks
t_tcf	List of complexity factors defined with `Settings/Project Types/Estimation Factors/Technical...`
t_template	RTF templates ?!
t_testclass	?!
t_testplans	This table is not currently used by EA[1]
t_testtypes	List of test factors defined with `Settings/Project Types/Maintenance/Test Types`
t_trxtypes	Matrix Profile; Painter Settings; Auto counters defined with `Setting/Auto Names and Counters...` more ?!

[1] As of EA version 9.3

Name	Description
t_umlpattern	UML patterns imported via `View/More Project Tools/Project Resources/UML Patterns`
t_version	?!
t_xref	This and that
t_xrefsystem	Various profiles
t_xrefuser	?!
usys_system	Key-value pairs for repository wide settings
usystables	A list of all the tables above along with the version where they were introduced. This table is needed (only?) during a project transfer.
usysoldtables	I have not the faintest idea ?!
usysqueries	I have not the faintest idea ?!

4. Most Important Tables

In this chapter we are going into quite some details of the most important tables. Namely these are that for elements, packages, diagrams, diagram objects, connectors and tagged values. Those are the ones you most likely need to retrieve often.

The single columns have a short description of what I think is their meaning. Some are obvious, some are just smoke signals. A reference to the GUI screen shots is placed where this is possible.

To improve readability and reference the properties were either split into several sub-tables or the table has an *indicator* on top of a logical section. Each sub-table is sorted alphabetically according to the name in `Column`.

4.1 More things than you find in the Project Browser: t_object

As you already know, this table holds all elements stored in the repository. That is any element you can see in the project browser plus those not shown like notes, boundaries and a couple of other elements. Please note that the Package element is a pendant[1] to the Package itself. Part of the information in both is redundant and both link to each other.

The following table lists the most important properties also to be found in the General properties window. The remaining properties are listed subsequently in logical groups.

Column	Description
Alias	Alias property
Author	Author property
Complexity	Complexity property Valid values are: *1* for Easy, *2* for Medium, *3* for Difficult
GenType	Language property String value. Note that this value does not appear unless it is defined as Product in the Language Datatypes
Name	Name property
Notes	Notes property
PDATA5	Keywords property
Phase	Phase property String value
Scope	Scope property String value
Status	Status property corresponds to values in `t_statustypes`
Stereotype	Stereotype property
Version	Version property

[1] I have detailed this in my book Scripting EA.

In order to retrieve the `Notes` column with EA's SQL Search you need to name the column like this:

```
1  SELECT t_object.Note AS [Notes], * FROM t_object
```

The `Note` column will not appear in the `*` nor when selected unnamed. EA will filter all columns named `Note`. The square brackets around `Notes` are meant for use in EAP files and might have to be removed for other DB providers.

This warning applies not only for the table `t_object` but all tables which have a `Note` column.

4.1.1 Key links

Column	Description
Diagram_ID	Only for Text elements; reference to primary key of the diagram
ea_guid	A global UID shown here Use `Repository.GetElementByGUID (ea_guid)` to retrieve this element
Object_ID	Primary, unique key for the element Use `Repository.GetElementByID (Object_ID)` to retrieve this element
Package_ID	Primary key of the package where the element is located
ParentID	Only for nested elements: primary key of the object
Classifier_guid	Redundant GUID for the Classifier property

For `ActivityParameter` elements the `Classifier_guid` encodes the type of the parameter. This can eventually be one of

{EABOOL00-B653-4f3c-A010-30205D67F5F5}
{EAINT000-B653-4f3c-A010-30205D67F5F5}
{EAREAL00-4339-434b-BC17-A5E1FDC63F6C}
{EASTRING-B653-4f3c-A010-30205D67F5F5}
{EAUNAT00-B653-4f3c-A010-30205D67F5F5}

And now guess…

4.1.2 Details

Column	Description
Abstract	Abstract property
Cardinality	Cardinality property String value
Concurrency	String equivalent of the Concurrency property for class elements
IsActive	Boolean values for the Is* properties
IsLeaf	ditto
IsSpecification	ditto
IsRoot	ditto
Persistence	Persistence property String value

4.1.3 Dock

Column	Description
CreatedDate	Created property
GenFile	Filename property String value
ModifiedDate	Modified property
Multiplicity	Multiplicity property String value

4.1.4 Appearance

Column	Description
Backcolor	Background Color property RGB values in decimal
Bordercolor	Border Color property RGB values in decimal
BorderStyle	For frame-like elements (boundaries etc.). Corresponds to the style (0..3) where *3* = solid line
BoderWidth	Border Width property
Fontcolor	Font Color property RGB values in decimal
StyleEx	Individual font settings

4.1.5 `Object_Type` and `NType`

Another classical database design anti-pattern can be found in the two columns `Object_Type` and `NType`. While `Object_Type` is a string value, `NType` adds forgotten salt to any of the expressions depending on some additional context informtion.

For `Object_Type` EA seems to mangle the display value in a query. In particualar this happens with a BPMN2.0 `StartEvent`. Other cases may exist. If you have a model with such a `StartEvent` the `Object_Type` contains `Event` but the SQL result is displayed as `StartEvent`. You can verify this:

```
1  SELECT count(*),object_type FROM t_object GROUP BY object_type
```

will show that actually there are `Object_Type = 'Event'` rows. Now

```
1  SELECT * FROM t_object where object_type = 'Event'
```

will find that row(s) but display `StartEvent` instead of `Event`!

The meaning of `NType` per `Object_Type` is hard to structure due to its context sensitivity. So here are a couple of statements:

- If `Object_Type` is one of `Activity`, `Artifact`, `Class`, `Interaction`, `Requirement`, `State`, `StateMachine`, `UseCase` (and probably some more) and `NType` is 8 and `PDATA1` has a number greater then zero then `PDATA1` corresponds to t_diagram.Diagram_ID of the composite diagram.
- `Artifact` and 32 means that it renders an image. Unfortunately I have no idea where that image comes from.
- `NType` has 0 or 1 for lots of plain elements. It's not obvious what might be the difference.
- `Constraint` and `MessageEndpoint` seem to have 2 for `NType`.
- The following table lists a couple of further combinations:

Object_Type	NType	Meaning
Text	0	plain text
	19	hyperlink; `Name` has `$help://`, `$inet://`, etc.
	76	legend
	82	diagram hyperlink; `PDATA1 == t_diagram.Diagram_ID`
Event	0	send
	1	receive
	2	accept timer
UMLDiagram	0	frame
	1	diagram reference
StateNode	3	initial
	4	final
	5	history
	10	junction
	11	choice
	13	entry point
	14	exit point
	15	deep history
	100	activity initial
	101	activity final
	102	flow final

Object_Type	NType	Meaning
InteractionFragment	0	alt
	1	opt
	2	break
	3	par
	4	loop
	5	critical
	6	neg
	7	assert
	8	strict
	9	seq
	10	ignore
	11	consider

4.1.6 Misc

Column	Description
Classifier	NULL or *0* where not defined. If > *0* then it is the primary key of the element which classifies the element
EventFlags	semi-colon separated list of attributes with links into the Risk/Metrics/etc. tables
GenOptions	Some very nasty semi-colon separated list of attributes (used for code generation?!)
GenLinks	String value of the class which this one is specialized from Usually only set where classes are reverse engineered and the general class is missing
Header1/2	Used for code generation
PDATA1	For Package elements: primary key of the package For Elements: Same as the Status column For Parts/Instances: GUID of the classifier For UseCase: *#EXP#=<ep>;* semi-colon separated list of Extension Points *<ep>* For Notes: linked element feature name For Text displaying as hyperlink: t_diagram.Diagram_ID For Requirements: Status property For UMLDiagram: Diagram_ID of the underlying diagram;
PDATA2	For Elements: Same as the Priority column For Notes: Object_ID of the linked feature element For Requirements: Priority property
PDATA3	For Elements: Same as the Difficulty column For Notes: Reference name into the linked feature element For Ports in classified Parts: the GUID of the corresponding Port in the Classifier. For State: t_diagram.Diagram_ID of the composite diagram For Requirements: Difficulty property
PDATA4	For Note elements: *Yes* if the note is linked to an element feature and *idref=<val>;* list where *<val>* is the primary key of the connector(s) to which the note is linked For elements: If > *0* this is the primary key of the connector for which this element is defines as association class

Column	Description
RunState	For objects a list of run state variables
Style	see below
Tagged	Flag to show that an element it bookmarked (little red triangle in diagram)
TPos	Tree order of the element in the project browser

I have no idea why the column Style is called by that name since only StyleEx really has some character style information. Alas, this is a semi-colon separated list of attribute assignments. It's a Sparxian zoo where all sorts of homeless animals can gather. For example the `Locked=true;` appears whenever an element has been marked on a diagram with the `Is Locked` context. The element is now protected from manual changes in the GUI (and even the API, though it does not tell the reason for an update failure in `GetLastError`) and the element shows a red excalamtion mark in the browser. Another on indicates a linked document if = *MDoc=1;* There are some more options used like `ShowBeh`, `EScrpt` and `LinkOpen` which seem to be too exotic to be explained here (currently).

4.1.7 Unknown or heritage

Column	Description
Effort	Always *0* ?!
Visibility	Always NULL ?!
StateFlags	Always NULL ?!
PackageFlags	Always NULL ?!
ActionFlags	Always NULL ?!

4.2 The Repository Structure: t_package

This table holds property information for packages. As packages hold some extra properties different to those in the t_object this extra table is needed.

Column	Description
ea_guid	A global UID shown here Use `Repository.GetPackageByGUID (ea_guid)` to retrieve this package Use `Repository.GetElementByGUID (ea_guid)` to retrieve related element[2]
Package_ID	Primary, unique key of the package Use Repository.GetPackageByID (Package_ID) to retrieve this package — *Links*
Parent_ID	Package_ID of the parent package — *General*
BatchSave	Batch Export property
BatchLoad	Batch Import property
CodePath	Code path from where this package has been imported
CreatedDate	Created property
IsControlled	Control Package property
LastLoadDate	Last Load Date property
LastSaveDate	Last Save Date property

[2]This is one of those cases where GUIDs are used to cross-reference different things.

Column	Description
LogXML	Log Import/Export property
ModifiedDate	Modified property
Name	Name property value
Namespace	*1* if this package is defined as Code Engineering/Set as Namespace Root
Notes	A mirror of the Notes property of t_object.`ParentID == Package_ID`
PackageFlags	Mixed information detailed here
PkgOwner	Owner property
Protected	Always FALSE (?!)
TPos	A mirror of the TPos property of t_object.`ParentID == Package_ID`
UMLVersion	Version ID property
UseDTD	Use DTD property
Version	A mirror of the Version property of the t_object.`ParentID == Package_ID` This value is only set when explicitly changed
XMLPath	XMI Filename property

Watch out! t_package uses **Parent_ID** while t_object (and other tables) uses **ParentID** (without underline)!

4.3 The Diagram Frame: t_diagram

This table contains the properties for all diagrams. The objects you see in the diagram itself are stored in t_diagramobjects.

Column	Description
Diagram_ID	Primary, unique key of the diagram Use `Repository.GetDiagramByID (Diagram_ID)` to retrieve this diagram
ea_guid	A global UID Use Repository.GetDiagramByGUID (ea_guid) to retrieve this diagram — *Links*
Package_ID	Primary key of the package
ParentID	Only for diagrams nested inside an elements: primary key of the object — *General*
AttPub	Public property
AttPri	Private property
AttPro	Protected property
Author	Author property
CreatedDate	Created property
cx	Number of pixels in X-direction used at maximum[3]
cy	Number of pixels in Y-direction used at maximum
Diagram_Type	See the `t_diagramtables` table for valid values appearing here
HTMLPath	?!

[3] It appears that these pixels are derived from the printer resultion being current at the time of diagram creation. The numbers only change when paper dimension and orentiation are changed. They do **not** represent the used amount by diagram elements.

Column	Description
Locked	Diagram has been locked via security. Can also be set without security
ModifiedDate	Modified property
Name	Name property
Notes	Notes property
Orientation	Diagram/Advanced…/Page Setup/Orientation first char (*P* or *L*)
PDATA	All the gory details about other diagram properties
Scale	Scaling in percent. Manipulated with the magnification glass icons
ShowBorder	Show Page Border (Current) property
ShowDetails	Show Diagram Details property
ShowForeign	Show Namespace property
ShowPackageContents	Package Contents check mark
Stereotype	Stereotype property
StyleEx	Even more details describing the style of diagram, like whether swim lanes are active and so on
Swimlanes	Details defined with `Swimlanes and Matrix...` context menu
TPos	Tree order of the diagram in the project browser
Version	Version property

4.4 Elements Inside Diagrams: t_diagramobjects

Each rendering in a diagram is that of a respective Element[4]. This table refers those elements and stores their position and style in the respective diagram.

Column	Description
Instance_ID	Primary, unique key of the diagram object — *Links*
Diagram_ID	Primary key of the diagram
Object_ID	Primary key of the element which is rendered — *General*
ObjectStyle	Single elements on a diagram can be rendered individually overriding the Style of Element. See DiagramObject ObjectStyle Property
RectTop	Top Y-coordinate of the element. 0 is the top and any negative value is the y-value (below the top)
RectLeft	Leftmost X-coordinate starting from 0 being the leftmost position
RectRight	Rightmost X-coordinate of the object. This is a greater value than `RectLeft`
RectBottom	Bottom Y-coordinate of the element. This is a smaller (more negative) value than `RectTop`
Sequence	Layer of the object. All elements will be drawn in the order of the Sequence

[4]Note that for the relations the table `t_diagramlinks` is used. This table will be detailed in a future release of this book.

4.5 Non-Standard Connectors: t_diagramlinks

EA renders connectors between elements in any diagram with a default (straight connector). Unless the user tells to use something else. In this and only this case EA creates an entry in `t_diagramlinks` where it stores the settings for this connector. These settings are valid for the very single diagram which is specified along with the record. That means that if you want a certain connector to appear differently you need to specify this for each single diagram.

Column	Description
DiagramID	Primary key of the diagram
ConnectorID	Primary key of the connector
Geometry	A CSV list. See below
Style	A CSV list. See below
Hidden	Boolean which is true if the connector shall be hidden
Path	A semi-colon separated list of X:Y coordinates. At each coordinate EA will render a bend.
Instance_ID	Primary, unique key of the diagram link

4.5.1 Geometry and Label Formatting

The `Geometry` keeps information about start and ending point of a connector. The text is separated by a dollar sign[5] which separates geometry from label information. Both are CSV lists where the single tags have the following meaning:

Geometry Tag	Description
SX	Relative start X coordinate from the start object (t_connector.Start_Object_ID). The value ranges ±width/2 of the start object
SY	Relative start Y coordinate from the start object (t_connector.Start_Object_ID). The value ranges ±height/2 of the start object
EX	Relative start X coordinate from the end object (t_connector.End_Object_ID). The value ranges ±width/2 of the end object
EY	Relative start Y coordinate from the end object (t_connector.End_Object_ID). The value ranges ±height/2 of the end object
EDGE	Specifies the edge from where the connector starts at the start object: 1=bottom; 2=left; 3=top; 4=right

Which of the labels appear and are relevant depends on the type of connector. Some have only a subset of visible labels. The Label Tags have the general format *<labelTag>=<colonList>*.

[5] Thanks to Heidi for supplying me with additional information about the use of the `Geometry`.

Label Tag	Description	Usage
LLT	Label Left (source) Top	Source Role
LLB	Label Left (source) Bottom	Source Multiplicity
LMT	Label Middle Top	Name
LMB	Label Middle Bottom	Stereotype
LRT	Label Right (dest) Top	Dest Role
LRB	Label Right (dest) Bottom	Dest Multiplicity
IRHS	Information Flows realized (dest)	=
ILHS	Information Flows realized (source)	=

<*colonList*> itself is a list of attributes separated by colon having the format <*fmtTag*>=<*val*>. The tags have the following meaning:

Fmt Tag	Description
CX	Width of the label box
CY	Height of the label box
OX	X-offset from the default position
OY	Y-offset from the default position
HDN	0=label is visible; 1=label is hidden
BLD	Set to 1 if the Bold option is set via the GUI. The label does not render bold, however
ITA	Obviously reserved for future use. Label will not render in italics
UND	Obviously reserved for future use. Label will not render underlined
CLR	RGB value of the label color. -1 is the default color
ALN	Alignment of the label. 0=left; 1=center; 2=right
ROT	Rotation of the label. 0=none; 1=clockwise; -1=anti-clockwise
DIR	-1=to source; 1=to destination; 0 or not present=no indicator

Whenever you reassemble this list be sure to give it the same format with the dollar inside and the single lists separated with colon or semi-colon as appropriate.

4.5.2 Style

The `Style` specifies the appearance of a connector:

Tag	Description
Mode	The Style corresponding to the drop down: 1=Direct; 2=Auto Route; 3=Custom. In case this value is 3 then the `TREE` style tag may appear specifying the values below `Custom` in the drop down. 8 would be the number for bezier style.
EOID	Some EA internal GUID you can ignore
SOID	Some EA internal GUID you can ignore
COLOR	Some RGB value where *-1* means the default color.
LWidth	The width of the connector, where *0* is the default, *1* the thinest and *5* the thickest
TREE	*OR*=`Orthogonal Rounded`; *OS*=`Orthogonal Square`; *LH*=`Lateral - Horizontal`; *LV*=`Lateral - Vertical`; *V*=`Tree (Vertical)`; *H*=`Tree (Horizontal)`

4.6 Connecting Elements: t_connector

Any relation between different elements is stored in this table. A connector refers to two Elements being source/start and destination/end. If source and target are the same this is a self reference.

Column	Description
Connector_ID	Primary, unique key of the connector
ea_guid	A global UID Use Repository.GetConnectorByGUID (ea_guid) to retrieve this connector — *Links*
End_Object_ID	Target of the connector (with name Target)
DiagramID	Primary key of the diagram Applies to Sequence connectors only
Start_Object_ID	Source of the connector (with name Source) — *General*
Connector_Type	See the `t_connectortypes` table for valid values appearing here Appears as window title of the properties window
Direction	String equivalent of the Direction property
Name	Name property
Notes	Notes property
PDATA1	For StateFlow connectors: Trigger name For Collaboration and Sequence connectors: Message/Synch property For association class connectors: t_object.Object_ID of the Association Class
PDATA2	For Collaboration: Message/Return Value property For ControlFlow: Constraints/Guard property
PDATA3	For Sequence connectors: Message/Control Flow Type/Kind property For StateFlow connectors: Effect property
PDATA4	For Collaboration connectors: rendered sequence number of the message For Realisation connectors: the first constraint For Sequence connectors: Message/Control Flow Type/Is Return property
PDATA5	Advanced style information
SeqNo	Only for `Connector_Type == Sequence`. The order of the message
StateFlags	For Collaboration connectors: *isReturn=true/false;* reflecting the `Message/Control Flow Type/Is Return` property Advanced settings for Sequence connectors
Stereotype	Stereotype property
StyleEx	Advanced properties
SubType	A categorization of some connectors
VirtualInheritance	VirtualInheritance property — *Formatting*
IsBold	Line thickness. *0* is default. *1..3* bold steps

Column	Description
LineColor	*-1* for default color. RGB value else

4.6.1 Connector source/target

Column	Description
SourceAccess	Access property
SourceCard	Multiplicity property
SourceChangeable	Changeable property (*none, frozen* or *addOnly*)
SourceContainment	Containment property (*Unspecified, Reference* or _Value)
SourceConstraint	For Association connectors: Target Role/Constraint(s) property For Sequence connectors: Sequence Expression/Constraint property
SourceElement	Member Type property
SourceIsAggregate	Aggregation property. *0*=none, *1*=shared, *2*=composite
SourceIsNavigable	TRUE if `Source Role/Navigability ==` *Navigable*
SourceIsOrdered	Multiplicity/Ordered property
SourceQualifier	Qualifiers property. A semi-colon separated list
SourceRole	Source Role/Role property
SourceRoleNote	Role Notes property
SourceStereotype	Source Role/Stereotype property
SourceStyle	Likely some redundant information like the following: *Union=0;Derived=0;AllowDuplicates=0;Owned=0; Navigable=Non-Navigable;alias=alias;*
SourceTS	Scope property (*instance* or *classifier*)

Target properties are analogous and have a *Dest* prefix instead of *Source*.

4.6.2 Connector labels

Column	Description
Btm_End_Label	Rendered source multiplicity
Btm_Mid_Label	Rendered Stereotype property
Btm_Start_Label	Rendered source role display where public is '+' and so on
End_Edge	Only for qualified properties
PtEndX	Dimension of the qualifier (for associations) or object life (for messages) box
PtEndY	ditto
PtStartX	ditto
PtStartY	ditto
Start_Edge	Only for qualified properties
Top_End_Label	Rendered target multiplicity
Top_Mid_Label	Rendered Name property
Top_Start_Label	Rendered target role display where public is '+' and so on

4.6.3 Connector unknown

Column	Description
ActionFlags	Always NULL ?!
DispatchAction	Always NULL ?!
DestRoleType	?!
EventFlags	Always NULL ?!
HeadStyle	?!
IsRoot	Always FALSE ?!
IsLeaf	Always FALSE ?!
IsSignal	Always FALSE ?!
IsSpec	Always FALSE ?!
IsStimulus	Always FALSE ?!
LineStyle	?! The line style is actually encoded in the `t_diagramlinks.Style`
LinkAccess	Always NULL ?!
RouteStyle	?!
SourceRoleType	?!
Target2	Always NULL ?!

4.6.4 Association classes

I rather put that info here as the construct starts with association, though it's a class as well. Anyhow, when you link a class to an association with the Advanced/Assocation Class context menu you will end up with bits in t_connector

Column	Description
Connector_type	*Class* in case of an association class else *NULL*
PDATA1	t_object.Object_ID of the association class or *0* else

as well as in t_object

Column	Description
NType	17 if it's an association class
PDATA4	t_connector.Connector_ID referring the association itself

5. Element Feature Tables

Mainly the element features comprise attributes and methods. Both are stored in the tables detailed below.

5.1 Attributes: t_attribute

This table holds property information for attributes. It is noticeable that the attributes property dialogue appears as column - and here in different variants.

Column	Description
ea_guid	A global UID shown here Use Repository.GetAttributeByGUID (ea_guid) to retrieve this attribute
ID	Primary, unique key of the attribute Use Repository.GetAttributeByID (ID) to retrieve this attribute — *Links*
Object_ID	Element for which the attribute is defined — *General*
AllowDuplicates	Allow Duplicates property value
Const	Const property value
Classifier	Element from which the attribute is classified
Container	Container type property value
Containment	Containment property value
Default	Initial property value
Derived	Derived property value
GenOption	?! Sometimes contains *SourceClass=<guid>;* or similar stuff *PROPERTY=<name>;* where *<name>* = Property property value
IsStatic	Static property value
IsCollection	Attribute is a Collection property value
IsOrdered	Ordered Multiplicity property value
Length	Length property value
LowerBound	Lower bound property value
Name	Name property value
Notes	Notes property value
Pos	Ordering position starting from 0
Precision	Precision property value
Scale	Scale property value
Scope	Scope property value
Stereotype	Stereotype property value
Style	Alias property value
StyleEx	?! contains sometimes values like *volatile=<n>;Literal=<n>;* where *<n>* is either *0* or *1*
Type	Type property value
UpperBound	Upper bound property value

5.2 Operations: t_operation

Like for attributes the operations for elements are stored in their own table.

Column	Description
ea_guid	A global UID shown here Use Repository.GetOperationByGUID (ea_guid) to retrieve this operation
OperationID	Primary, unique key of the operation Use Repository.GetOperationByID (OperationID) to retrieve this operation — *Links*
Object_ID	Element for which the attribute is defined
Classifier	Element from which the operation return value is classified via Return Type property value — *General*
Abstract	Abstract property value
Behavior	Behavior property value
Code	Code property value
Concurrency	Concurrency property value
Const	Const property value
GenOption	?! Sometimes contains *SourceClass=<guid>;* or similar stuff
IsQuery	Is Query property value
IsStatic	Static property value
Name	Name property value
Notes	Notes property value
Pos	Ordering position starting from 0
Pure	Pure property value
ReturnArray	Return Array property value
Scope	Scope property value
Stereotype	Stereotype property value
Style	Alias property value
StyleEx	Show Behavior in Diagram property value *ShowBeh=1;* if property is checked
Synchronized	Synchronized property value
Type	Return Type property value — *Unknown*
IsRoot	?! Always *FALSE*
IsLeaf	?! Always *FALSE*
StateFlags	?! Always empty
Throws	?! Always empty

6. Tagged Value Tables

Tagged values are not stored in `t_taggedvalue` but a couple of different tables.

6.1 Element Tagged Values: t_objectproperties

Any tagged value for an Elements is stored in this table. There is not much notable about this table except the following:

- The Property column is not unique for a single element. Thus multiple tagged values of the same name can appear for a single element. If you have not set `Show Duplicate Tags` in the options you will be presented with that one having the lowest `PropertyID`.
- Operations, Attributes and Relations have their own `t_<...>properties` tables storing the appropriate tags.
- If `Value` contains the text *<memo>* the tagged value is of memo-type. That is, it appears with an ellipsis right to the *<memo>* text in the tagged values window.
- If `Notes` contains something like two line with *Values: <semi-colon separated list>* and *Default: <element from list>* this tag will appear as drop-down.

Column	Description
ea_guid	A global UID
PropertyID	Primary, unique key of the tagged value — *Links*
Object_ID	The element for which the tag applies — *General*
Notes	The notes for the tag
Property	The name of the tagged value
Value	The value for the tag

6.2 Attribute Tagged Values: t_attributetag

This table holds the tagged values for attributes. It looks exactly the same as that for elements except for one column:

Column	Description
ea_guid	A global UID
PropertyID	Primary, unique key of the tagged value — *Links*
ElementID	The attribute for which the tag applies — *General*
Notes	The notes for the tag
Property	The name of the tagged value
Value	The value for the tag

6.3 Operation Tagged Values: t_operationtag

This table holds the tagged values for operations. It looks exactly the same as that for elements except for one column:

Column	Description
ea_guid	A global UID
PropertyID	Primary, unique key of the tagged value — *Links*
ElementID	The operation for which the tag applies — *General*
Notes	The notes for the tag
Property	The name of the tagged value
Value	The value for the tag

6.4 Connector Tagged Values: t_connectortag

This table holds the tagged values for connectors. It looks exactly the same as that for elements except for one column:

Column	Description
ea_guid	A global UID
PropertyID	Primary, unique key of the tagged value — *Links*
ElementID	The connector for which the tag applies — *General*
Notes	The notes for the tag
Property	The name of the tagged value
Value	The value for the tag

7. Security Related Tables

The following tables are only relevant if user security has been turned on. In order to check whether security is turned on in the repository the `t_secpolicies` table must be queried.

7.1 Settings: t_secpolicies

This table stores the settings for security as key-value pairs.

Column	Description
Property	Name of the property
Value	Value of the property

The following properties are defined:

Property	Value	Comment
UserSecurity	Enabled	Property is only present if the user has performed Project/Security/Enable…
RequireLock	1	Property is only present if the user has performed Project/Security/Require…
	0	Require User Lock to Edit has been turned off

7.2 Users: t_secuser

The list of users allowed in the system. UserLogin/Password are prompted during login.

Column	Description
UserID	GUID to identify a user
Department	Department entered in the user description
FirstName	First name
Password	Encrypted password. Does not contain the user name as part of the encryption
Surname	Surname
UserLogin	User name to be typed in the prompt

7.3 Groups: t_secgroup

The list of groups allowed in the system. All users can be assigned to an arbitrary number of groups.

Column	Description
GroupID	GUID to identify a group
GroupName	Name and
Description	description of the group entered in the properties

7.4 Assignment of users to groups: t_secusergroup

The list of groups allowed in the system. All users can be assigned to an arbitrary number of groups.

Column	Description
UserID	GUID of the user table
GroupID	GUID of the group table

7.5 Group permissions: t_secgrouppermission

Currently there are 40 different permission defined (0..39). Each number represents one permission in the group permission settings. Sparx might add new ones to the end of the list if new features will be added in the future.

Column	Description
GroupID	GUID of the group table
PermissionID	One of the numbers from the table below[1]

Key	Permission
35	Admin Workflow
5	Administer Database
30	Audit Settings
31	Audit View
33	Baselines - Manage
34	Baselines - Restore model
15	Change Password
17	Check Data Integrity
18	Configure Datatypes
20	Configure Images
24	Configure Packages
38	Configure Project Prerequisites
9	Configure Resources
19	Configure Stereotypes

[1] Courtesy of Guillaume: http://www.umlchannel.com/en/enterprise-architect/item/197-sparx-enterprise-architect-permissions-id-and-name-mapping-for-your-scripts-and-add-ins

Key	Permission
28	Configure Version Control
39	Edit Scripts
11	Export XMI
23	Generate Documents
21	Generate Source Code and DDL
12	Import XMI
7	Lock Elements
25	Manage Diagrams
36	Manage Glossary
14	Manage Issues
37	Manage Project Calendar
8	Manage Project Settings
3	Manage Reference Data - Update
6	Manage Replicas
13	Manage Tests
22	Reverse Engineer from DDL and Source Code
40	Run Scripts[2]
0	Security - Enable/Disable
2	Security - Manage Locks
1	Security - Manage Users
26	Spell Check
16	Transfer Data
32	Transform Package
4	Update Diagrams
10	Update Elements
29	Use Version Control
27	View Locks

7.6 User permissions: t_secuserpermission

Similar to the group permission the single users can be assigned individual permissions.

Column	Description
UserID	GUID of the user table
PermissionID	same as in group permission

[2]Introduced with V13

7.7 Locks: t_seclocks

These are the individual locks for packages, diagrams and elements.

Column	Description
UserID	GUID of the user table
GroupID	GUID of the group table if the lock was applied to that group
EntityType	*Element, Diagram, Package*
EntityID	GUID to identify the lock
LockType	?!
Timestamp	Time stamp when the lock was set

8. Rarely Used Tables

Here you will find some details about tables which are not of major importance. However, from time to time you will also need to deal with them. Note that this section is going to be populated with more information during the next near future.

8.1 Stereotypes: t_stereotypes

This table stores the definitions of stereotypes as found under `Settings/UML Types/Stereotypes`.

Column	Description
ea_guid	A global UID used as primary key
AppliesTo	The Base Class property
Description	The Notes property
Metafile	NULL
MFEnabled	The Metafile property
MFPath	The path to the metafile assigned with the Assign button
Stereotype	The Name property
Style	An XML-like string holding all the other attributes
VisualType	NULL

The `Style` column has the format

<STYLE fill="<color>" text="<color>" border="<color>" groupname="<group>" type="<type>"/>

Here *<color>* is a decimal RGB color value. *<group>* is the Group Name property. *<type>* is either *none, metafile* or *script* according to the properties. Some kind of duplicate definition with `MFEnabled` but that's what we already know from EA.

In case *<type> == script* the `Style` string is appended with

<SHAPE file="<contents>" type="EAShapeScript 1.0" enabled="1"/>

Here the *<contents>* is the HTML-escaped string representing the shape script.

8.1.1 Some history

Stereotypes are one of those open construction places introduced in EA which never come to an end. In the (not so far) past EA had[1] just "plain stereotypes": a simple string one could invent arbitrarily and assign it anywhere. For example you could take a class and invent a new stereotype "nice" enter it in the stereotype property of the class and hit enter. Now you had a new

[1] Past tense is not really correct since EA (open construction place) still allows doing it this way.

stereotype called “nice”. Nicely. Or not. Well, Sparx wasn’t wrong about that. The former UML specs was not specific about what a stereotype actually is. It was merely a meta-classification for UML elements. The specs listed a few examples and that was it. Later OMG got more specific about stereotypes and linked them strongly with profiles. That is, a stereotype became a means to extend basic UML meta types. This extension can be recognized by

- a different rendering of the element, in the simplest case you see the stereotype enclosed in guillemets above the element name.
- a number of tagged values which can be used to store domain specific properties in an element.

IIRC that was about UML 1.5 or so (not going to do detailed research since that’s not really relevant). However, since then you definitely have stereotypes in the above sense. The Sparxians are with OMG and did not ignore this semantics. But rather than sticking to the new definition they keep on living with their zombie stereotypes. So EA pretends that stereotypes are a property of named elements, which is not the case. It can’t be helped but having some background can’t be wrong either.

8.1.2 Available stereotypes

When you look into EA’s UML Types dialog (go figure where that’s located in your current EA version’s menu structure), you find around 120 different pre-defined stereotypes. Those are a mix of relicts (for “compatibility sake”) from the unspecific stereotypes of older UML versions and from the so-called UML Standard Profile (see chapter 22 of OMG’s UML 2.5 specification). And EA allows to create more of those wild stereotypes. You are advised here to enable security and restrain users from creating more of them. Even better you remove any entry in the table `t_stereotypes` and watch for new wild ones appearing there to have them removed.

Legal stereotypes are just those you can choose from the drop down/check list that appears when clicking the ellipsis after the element stereotype which are connected to some profile previously enabled via some MDG. Speaking of the latter, there are a couple of “basic” MDGs now (don’t remember exactly when they introduced that since in older EA versions it all was hard coded. Once you enable the `Basic UML ® 2 Technology` EA shows a `StandardProfileL2` which offers

- Type
- Auxillary
- Specification
- Utility
- Metaclass
- ImplementationClass
- Realization
- Focus

That matches some of the before mentinoned UML standard profile. It’s obviously incomplete, though.

Further enabling the `Core Extensions` MDG reveals another list named `EAUML` that contains

- BPSimCustomResultChart[2]
- BPSimResultChart[2]
- Chart[2]
- dbsequence[3]
- enumeration – just colored green; used in profiles
- function[3]
- functions[3]
- model document[4]
- ModelView[4]
- package[3]
- procedure[3]
- procedures[3]
- sequences[3]
- table[3]
- TimeSeriesChart[4]
- view[3]
- XSDall[5]
- XSDchoice[5]
- XSDsequence[5]

8.1.3 Qualified names

You might have noticed (especially when you have dealt with profiles extending other profiles like e.g. BPMN) that you need to specify the profile name to access a certain element. E.g. `BPMN2.0::BusinessProcess` refers to the `BusinessProcess` metaclass from the `BPMN2.0` profile. Now, unlike standard UML stereotypes which EA assigns via the `Stereotype` property of elements, you need to "assign the stereotype"[6] by dragging the `BPMN2.0` element from the toolbox over the element. If you try to enter "BPMN2.0::BusinessProcess" in the element's `Stereotype` property you end up with a new entry in `t_stereotypes` that has the name "BPMN2.0::BusinessProcess". It looks like a qualified name but it is just a string that coincidentally contains two consecutive colons.

If you see a qualified stereotype rendered in some diagram you can be pretty sure that something went wrong and you don't have the inteded profile element but rather some zombie. To remedy this you need to locate where that stereotype is used (query `t_object.stereotype` if it's for elements) and make a list of them. Now clear their stereotype field and remove the entry from `t_stereotypes`. Finally re-assign the correct stereotype from the according toolbox for all of them.

Lastly, how can you know from which profile a stereotype has actually been taken? Well, like always in such cases: consult `t_xref`.

[2] enables different properties; adds misc tagged values once edited
[3] For database modeling
[4] EA gimmicks
[5] For XSD modeling
[6] Here's that mess again. Stereotypes are indicators of instances from a profile, not properties. You can not "assign" them. It's just EA's implementation that causes trouble here.

8.2 Not the Tagged Values: t_taggedvalue

As indicated in the introduction this is kind of a smorgasbord. Actually these are the tags for methods, parameters and partially connectors. Interesting that these were put in one table but the other tags were moved to separate tables although all share the same column information. As far as I discovered the values in this table are used with WSDL and (as a reader found out) generally for connector source and target tagged values. Most likely this information here is still incomplete. But it will help you to find out the 'unknown' whenever you have close encounters of the 3rd kind.

Column	Description
PropertyID	Primary, unique key of the tagged value — *Links*
ElementID	A global UID referring the `ea_guid` of `BaseClass` — *General*
BaseClass	String literal describing the table where `ElementID` was taken from OPERATION_PARAMETER -> `t_operationparams` PACKAGE -> t_package ASSOCIATION_SOURCE -> t_connector ASSOCIATION_TARGET -> t_connector
Notes	Context dependent note[7]
TagValue	Context dependent value

To go backwards from the entries in this table

- get the table name from `BaseClass` and
- search the `ea_guid` matching `ElementID`.

There you are. Now for what these values are actually used is a bit harder to find out. Here's what I have so far: For the `BaseClass` column containing

- *OPERATION_PARAMETER*: holds the `Details` for the WSDL Operation Binding Parameters. The `Notes` column takes the property name (e.g. *use* or *encondingStyle*) while `TagValue` holds the parameter itself (e.g. *literal* for a *use*).
- *PACKAGE*: for WSDL packages. `TagValue` holds *LastImportFileDate* with `Notes` being either that date or empty.
- *ASSOCIATION_SOURCE*: For WSDL `TagValue` has the value *position* and `Notes` the decimal position value[8]. For general connector sources the `TagValue` is the name and `Notes` holds the value. Notes for the tagged value are appended to the `Notes` with *$ea_value=<notes>* where *<notes>* is the text you entered in the `Tagged Value Note` window. If you assign a text like *val$ea_notes=notes* then EA will assign *val* to the tag value and *notes* to its notes. EAUI!
- *ASSOCIATION_TARGET* : Is equivalent for connector target tagged values.

[7] It appears that this column is only generated when needed.
[8] Too long ago to dig into this deeper. But I guess those being concerned will know what is meant.

I once had a model with the additional columns `s_Generation`, `s_GUID`, `s_Lineage`, `Gen_Notes` and `Gen_TagValue` where `TagValue` contained *LastImportFileDate* and `s_Lineage` some base64 encoded crap. I have no idea how that was produced. None of the GUIDs showing up in that row was used elsewhere. Also (almost 100% sure) I had not tinkered with WSDL in that or any other model for 2 years. Maybe you have an idea about it?

8.3 Attribute Constraints: t_attributeconstraints

Column	Description
Object_ID	t_object.Object_ID of the attribute's element
Constraint	The Name property
AttName	?! always empty
Type	The Type property (*Invariant, Post-condition*, etc.)
Notes	The Notes property
ID	Primary unique key

8.4 Table Attribute Tags: t_attributetag

Column	Description
PropertyID	Primary unique key
ElementID	t_attribute.Object_ID of the attribute
Property	The Name property
Value	The Value property
Notes	usually empty; I only found notes from a tagged value `Correlation` here ?!
ea_guid	A global UID for whatever use (it can be changed by the API)

8.5 Linked Documents and Baselines: t_document

The main purpose of this table is to store baselines for packages and linked documents for elements. However, over time its use has been extended for a variety of other purposes[9].

Column	Description
DocID	A global UID used as primary key — *Links*
ElementID	ea_guid of the containing element — *General*
Author	Context dependent for forum entries
BinContent	Packed data
DocDate	Date when the record was created
DocName	Name of the containing element
DocType	*Baseline, ModelDocument, ExtDoc*[10]

[9] More than the documentation below will be detailed in a future release of this book.

[10] Other values appearing here are *DTree_RuleSet, Forum_Category, Forum_Subject, Forum_Thread, Model_Forum, SSDOCSTYLE, SSMODELDOCSTYLE* and *event_calendar* which control the meaning of the other columns.

Column	Description
ElementType	*Package, ModelDocument, ELEMENTSCRIPT, Post*
IsActive	A strange number ?!
Notes	Context dependent for forum entries
Sequence	NULL
StrContent	The string value for certain `DocTypes`
Style	Context dependent
Version	NULL

The value definition of the single columns depend on the contents of `ElementType`:

***Baseline* —**
the record stores a single baseline record. Besides the baseline creation properties `ElementType` contains `Package`, `Style` contains `Zip=1;` and `BinContent` the packed xml of the baseline.

***ModelDocument* —**
the record stores a linked document. `DocName` contains the string `a::<name>` where `<name>` is the name of the element which holds the linked document information. `ElementID` is the guid of this element. `Style` is empty and the `BinContent` contains the packed `str.dat` which holds the RTF formatted linked document.

***ExtDoc* —**
In that case the `BinContent` holds a zip with a file *str.dat* having the according doc or image.

***ELEMENTSCRIPT* —**
the record stores a script created for SysML element scripts. `DocName` contains the name of the element and `ElementID`its guid. `ElementType` contains something like

```
1  `Lang=JScript;Type=ElementScript;`
```

and `BinContent` contains the packed file `str.dat` which holds the script text.

***GAPMATRIXPROFILE* —**
the record stores a profile saved from `View/Gap Analysis Matrix`. `DocName` contains the name of the profile. The column `StrContent` hold the XML-formatted values for the profile.

***Post* —**
Contains forum discussions referring `ElementID`.

8.6 Mixed option: t_genopt

Model global settings are stored in this table. Most of them stem from `Tools/Options` but a there are a couple of other places where such settings can be applied. Further it is not indicated whether an entry in `Tools/Options` is meant globally (stored in t_genopt) or locally (stored in the user registry).

Column	Description
AppliesTo	Name of the option group
Option	A semi-colon separated list list of options as *NAME* or *NAME=VAL* entries

The following table lists some of the keywords to be found in `AppliesTo` along with a description of the according `Option` contents.

AppliesTo	Option
CMACRO	Language macros defined with `Settings/Language Macros` as *NAME* entries
class	Encoded options from `Tools/Options/Source Code Engineering` These appear as *NAME=VAL* entries and will not be detailed here Some are obvious but others need a little experimenting to find out their use
scenario	*usesManagedList=0;* ?! No idea where is defined/used
auditing	Encoded options from `Audit Settings` These appear as *NAME=VAL* entries and are quite obvious

There might be more values in `AppliesTo` ?!

8.7 Alternate Images: t_image

Column	Description
ImageID	Primary, unique key of the image
Image	The packed image in the format described in `Type`
Name	The 'Name' column from `Settings/Images...`
Type	The 'File Type' column from `Settings/Images...` *Bitmap* or *Metafile*

Actually you can save the binary data always as .PNG and they will be displayed correctly. I have no real knowledge about image formats, but `Bitmap` seems to be "real" .png while `Metafile` is .emf format. Obviously the Windoze viewer does not rely on the suffix but looks into the file to detect the magic strings.

8.8 User Defined Scripts: t_script

This table holds the groups and scripts you define locally via the `Scripting` window.

Warning: Not suitable for database designers with heart insufficiency!

Column	Description
ScriptID	Primary, unique key of the script/group
Notes	An XML string describing the group/script *<Group Type="<type>" Notes=""/>* for groups. See below for *<type>*. *<Script Name="<name>" Type="Internal" Language="<lang>"/>* for scripts. *<name>* is the name of the script. See below for the values of *<lang>*.
Script	For groups this is the name of the group. For scripts this is the plain script contents.
ScriptAuthor	Wow! This is the value of *ScriptName* for the group entry for scripts
ScriptCategory	A value to distinguish between group and script. It looks like a GUID but it has no "{}" 3955A83E-9E54-4810-8053-FACC68CD4782 = group

Column	Description
	605A62F7-BCD0-4845-A8D0-7DC45B4D2E3F = script[11]
ScriptName	Don't get confused. This is a GUID which identifies the entry

<*type*> from the `Notes` can take one of the following values: *NORMAL, PROJBROWSER, DIAGRAM, WORKFLOW, SEARCH* and *MODELSEARCH* which corresponds to the group types available from the group creation menu.

<*lang*> can take the following values: *VBScript, JScript* and *JavaScript* which also obviously corresponds to the values from the script creation menu.

If you feel like having a large cold beer now: go ahead. You deserved it.

8.9 Element Requirements: t_objectrequires

These are better called responsibilities, but for historical reasons the term requirements is still in use.

Column	Description
ReqID	Primary, unique key
Object_ID	t_object.ObjectID for which the responsibility applies
Requirement	The Name property
ReqType	The Type property (*Display, Functional*, etc.)
Status	The Status property (*Approved, Implemented*, etc.)
Notes	The Notes property
Stability	Always NULL ?!
Difficulty	The Difficulty property (*High, Low*, etc.)
Priority	The Priority property)*High, Low*, etc.)
LastUpdate	Time stamp from last modification

8.10 Element Constraints: t_objectconstraints

Column	Description
Object_ID	t_object.ObjectID for which the responsibility applies
Constraint	The Constraint property
ConstraintType	The Type property (*Invariant*, etc.)
Weight	not mapped in the GUI. According to Geert's finding this stores actually the sort order if you happen to re-arrange the constraints. Stored as a real number! Weird, weirder, EA.
Notes	The Notes property
Status	The Status property (*Approved*, etc.)

8.11 Element Files: t_objectfiles

The element related files are created from element properties Files.

[11] I have not the faintest idea why this is such a large string where a simple boolean would suffice.

Column	Description
Object_ID	t_object.ObjectID to which the file is related
FileName	The File Path property
Type	The Type property (*Local File* or *Web Address*)
FileSize	The Size property (as shown in Windoze explorer; e.g. *3 k*)
FileDate	The Last Write property
Note	The Notes property

File size and date are stored when adding the entry. I have no idea whether the will be updated at any place later.

8.12 Scenarios for (mainly) Use Cases: t_objectscenarios

Column	Description
ea_guid	The ea_guid of the related element — *Links*
Object_ID	t_object.ObjectID of the containing element — *General*
EValue	A float formatted string. Used for ?!
Notes	Description property
Scenario	Notes property
ScenarioType	Type property
XMLContent	The xml- formatted Structured Specification property

The `XMLContent` is rather trivial, e.g.:

```
<path>
 <step name="step1" guid="{E51CC8C2-01D3-4575-B92F-B7D6B6551720}" level="1"
  uses="use" useslist="" result="res" state="stat" trigger="1" link=""/>
 <step name="step2" guid="{2C9C7BE8-2C6F-4187-B79F-5D9F4BAD60A9}" level="2"
  uses="use2" useslist="" result="res2" state="stat2" trigger="0" link=""/>
 <context/>
</path>
```

The `step` property appears for each defined step resulting in a single row in the Structured Specification property. The `name` attribute represents the Action column, `used` attribute is Uses, `result` is Results, `state` is State and `level` is the Step column[12].

8.13 Parameters for Operations: t_operationparms

Operation parameters as defined in the Parameters Properties Window.

[12]This is not the full story yet. The rest will come in a later book release – if so requested.

Column	Description
OperationID	[t_operation].OperationID or the related operation
Name	The Name property
Type	Type of the parameter
Default	The Default Value property
Notes	The Notes property
Pos	Sort oder within the parameters of the operation
Const	The Fixed Value property
Style	Always NULL ?!
Kind	The Direction property
Classifier	t_object.ObjectID of the classifier or *0*
ea_guid	A global UID (for whatever use)
StyleEx	Another smorgasbord

8.14 Various Profiles: t_trxtypes

Any profile defined under `View/Relationship Matrix` is stored in this table. Further the Painter settings from the diagramming tool bar. And also the Auto counters defined with `Setting/Auto Names and Counters...`.

Column	Description
Description	Relationship Matrix: *MXProfile* ; Painter: *Style*; Counter: *Autocount*
NumericWeight	*1*
Notes	The settings applied to the profile / auto counter; A semi-colon separated list list of *NAME=VAL* entries
TRX	The name of the profile / auto count element type
TRX_ID	The primary key
Style	Always NULL

For the auto counters the `Notes` column contains a CSV

```
prefix=<pre>;suffix=<suf>;active=<0|1>;active_a=<0|1>;counter=<ctr>;
```

The `active` fields are the check boxes for name and alias. `counter` is the formatted number which will next be created for the element specified in the `TRX` column.

Like many other EA tables this one looks like it could store also different information. However, I only found the profiles, the diagram styles and recently the auto counters to be stored here.

8.15 Status Types: t_lists

This table contains the values defined with various `Settings/Project Types/General Types/...` tabs.

Column	Description
ListID	A global UID used as primary key
Category	Defines the tab where Name is defined
Name	The name as shown in the GUI
NVal	Numeric order of the single `Category` entries
Notes	NULL

Initially only `Category` with the value *ConstStatusType* is found in this table. Other values only appear if individual names are entered via the according `Settings/Project Types/General Types/...` tabs. The following table lists the tab names that correspond to `Category`.

Category	Tab
ConstStatusType	Constraint Status
DifficultyType	Difficulty
PriorityType	Priority
TestStatusType	Test Status

The columns `NVal` and `Notes` are not supplied from the GUI.

8.16 Maintenance: t_objectproblems

This table contains the maintenance entries defined with `View/More Element Tools/Maintenance` dialogue.

Column	Description
Object_ID	t_object.ObjectID of the referred element
Problem	Name property of the Defect/Change/Issue/Task
ProblemType	Defects/.../Tasks tab
DateReported	Reported property
Status	Status property
ProblemNotes	Description property
ReportedBy	Reported by property
ResolvedBy	Resolved by property
DateResolved	Resolved property
Version	Version property
ResolverNotes	History property
Priority	Priority property
Severity	NULL

8.17 Various Profiles: t_xrefsystem

This table is used for multiple profile settings. Accordingly some columns have common meaning while others depend on the value stored in the column `Type`. Here are the common columns:

Column	Description
XrefID	A global UID used as primary key
ToolID	NULL
Name	Name of the profile as shown in the GUI
Type	Type of the profile (see below)
Visibility	NULL
Namespace	see below
Requirement	NULL ?!
Constraint	NULL ?!
Behavior	NULL ?!
Partition	NULL ?!
Description	see below
Client	see below
Supplier	see below
Link	see below

Valid values for `Type` are:

Type	Value
PView	Model views defined with `View/Model Views`
DFilter	Diagram filters defined with `View/Diagram Filters`

8.17.1 Diagram Filters

For `Type` having the value *DFilter* the other columns in `t_xrefsystem` mean:

Namespace	NULL
Description	The XML formatted search
Client	The name of the Author (as shown in the GUI)
Supplier	empty
Link	empty

8.17.2 Model Views

As stated above the entries with `Type=='PView'` describe model views. Other columns in this context have the following meaning:

Type	Value
Name	Name of the View
Namespace	Decimal from 1 to 10 corresponding to the Model View icon
Description	For packages (Namespace=3): `sType=Package;` For diagrams (Namespace=2): `sType=Custom;` For searches (Namespace=9): `srchID=<guid>;AutoRefresh=<0/1>;Notify=<0/1>;RefreshSeconds=<tnum>;`
Client	Always 0
Supplier	For the root (Namespace=8): `ModelRoot` Else the GUID of the parent
Link	ea_guid of the relevant package/diagram/element

8.18 RTF: t_rtf

This table contains various user defined options defined in the `Project/Documentation/Generate Documentation` window.

Column	Description
Type	A keyword description the type of options
Template	A semi-colon separated list list of options as *NAME* or *NAME=VAL* entries

Valid entries in `Type` are

Type	Description
ProjectOpts	Entries from the `Project Constants` tab
LangOpts	Entries from the `Word Substitution` tab
LangTags	Entries from the `Codepage`tab

8.19 Repository Settings: usys_system

This table contains a lot of `Property/Value` pairs. And honestly I don't know what they are used for in most cases.

`LastUpdate`
: Only a few seem obvious like this. But it contains a cryptic hex string.

`ProjectGUID`
: Same goes with this one. Although it's likely what the name suggests, I have no idea where it is used. A XMI export does not seem to have it included.

`VersionDate`
: This one states Jan-31 2004. IIRC that time the 3.x release was replaced by 4.0 and from then on the database was not really changed.

`TemplatePkg`
: Geert posted the use of this one. It holds the t_package.Package_ID of the package which was set to be the Template package (`Settings/Project Template Package...`).

`Default Diagram`
: Another hint from Geert. It holds the GUID of the diagram that has been defined as user default.

`Diagram_Layout`
: If you save layout options to be default then this property holds the same formatting information as described in DiagramObject ObjectStyle Property.

`Version`
: This value changes obviously only after several decades (see below) and is currently *4.01*.

`VersionDate`
: is *Jan-31-2004*. It looks as if the database of EA does not change too often.

8.20 Auditing: t_snapshot

There had been a couple of questions regarding the audit format, which is why I add a few examination results here. Still nothing detailed, but it might help the one or other.

Column	Description
SnapshotID	Primary key
SeriesID	Just fond *LOG* here
Position	not unique so obviously some changes take more than one row
SnapshotName	the affected table name like *t_object*
Notes	see below
Style	type of update like *INSERT* etc.
ElementID	Was just 1 or 0. ?!
ElementType	?!
StrContent	?!
BinContent1	see below
BinContent2	see below

Notes
: I had lines starting with "Audit"... which obviously told that auditing was changed. Others started with a GUID that was the element/connector/etc. GUID of the respective table in *SnapshotName* followed by mixed data.

BinContent1
: Zip with some XML data like

```
<LogItem>
  <Row Number="0">
    <Column Name="Audit Options">
      <Old Value=""/>
      <New Value="All"/>
    </Column>
    <appliesTo/>
    <Column Name="Auditing">
      <Old Value="Disabled"/>
      <New Value="Enabled"/>
    </Column>
  </Row>
  <Details User="Thomas" DateTime="2019-08-15 22:11:47"/>
</LogItem>
```

BinContent2
: UTF8 string with some XML data like

```
<metadata>
  <Row Number="0" Standard="1">
    <Level LevelName="Audit Settings" Name="Auditing" GUID="Auditing"/>
  </Row>
  <Details User="Thomas" DateTime="2019-08-15 22:11:47" Standard="1"/>
</metadata>
```

If I happen to dig out more it will be added in the future. For now the above should already help you to figure out who made a change to what from the audit log.

9. Marvelous References

This chapter might be the start of a new book. Or maybe it will stay thin because the treasures inside the `t_xref` table are too secret to be uncovered. We will see.

However, here are some bits from this marvelous table. Currently they are about stereotypes and MDG profiles but there's a lot more hidden.

9.1 A simple table: t_xref

Column	Description
XrefID	A GUID which identifies the record uniquely — *Links*
Behavior	NULL, a GUID or any from the list below or *generalizationSet* for `Generalization` connectors having defined a generalization set
Client	The ea_guid column of the referenced element
Description	see details below
Link	NULL or the ea_guid column of another related element
Supplier	NULL or the ea_guid column of a related element — *General*
Constraint	NULL
Name	The context description for the entry. In conjunction with `Type`
Namespace	NULL or any from the list below
Partition	NULL or a numeric value
Requirement	NULL or *back=-1;*
Type	See possible values in the table below
Visibility	NULL or *Public*

Here are the possible values for some of above columns. All values are context dependent and will be explained as discovered in the following sections. There might be more values, but that's the truth so far, starting with the Name-Type tuple:

Name	Type
Stereotypes	attribute property
Stereotypes	element property
Stereotypes	operation property
CustomProperties	connector property
MOFProps	connector property
OwnedMembers	connector property
Stereotypes	connector property
CustomProperties	connectorDestEnd property
CustomProperties	connectorSrcEnd property
CustomProperties	element property

Name	Type
DefaultDiagram	element property
MOFProps	element property
OwnedMembers	element property
Partitions	element property
MOFProps	ownedelement property
Analysis	swimlane
Business Model	swimlane
Deployment	swimlane
Design	swimlane
Requirements	swimlane
Use Cases	swimlane
Class	Transformation
EJBDeploymentDescriptor	Transformation
EJBEntityBean	Transformation
EJBHomeInterface	Transformation
EJBKeyClass	Transformation
EJBRemoteInterface	Transformation
EJBSessionBean	Transformation
LinkTable	Transformation
Table	Transformation
ProfileOptions	package property

Name

Analysis, Business Model, Class, CustomProperties, DefaultDiagram, Deployment, Design, EJBDeploymentDescriptor, EJBEntityBean, EJBHomeInterface, EJBKeyClass, EJBRemoteInterface, EJBSessionBean, LinkTable, MOFProps, OwnedMembers, Partitions, Requirements, Stereotypes, Table, Use Cases or *XSDClass*

Type

attribute property, connector property, connectorDestEnd property, connectorSrcEnd property, element property, operation property, ownedelement property, swimlane or *Transformation*

Namespace

C#, DDL, EJB Entity, EJB Session, ERD to Data Modeling, Java or *XSD*

Behavior

actual, argument, event, formal, result, specification, target or *trigger*

Description

This either a GUID or a semi-colon separated list. The latter appears as

RefGUID=<guid>;RefName=<name>;

or

OE0=<guid>;OE1=<guid>;OE2=...

or

@<tag>;<list>@END<tag>;

where *<tag>* is one of *ELEMENT, PAR, PROP* or *STEREO* and *<list>* is a semi-colon separated list. This entry can appear itself as list, e.g. for multiple stereotypes.

9.2 Definition of Multi-Stereotypes

Here's one aspect of `t_xref` which is related to storing stereotypes derived from MDG profiles. In the beginning the world was just like a big ball with nothing remarkable on it. But the stereotypes started their evolution. First there was not only one stereotype but multiple that could apply to a single element. Then MDG profile appeared and made it even more complicated. So Sparx decided: there must be reference and it created `t_xref`.

Honestly I can't remember whether `t_xref` existed in the early version of EA but my stomach tells me: no. However, one reason for `t_xref` is to sort out stereotypes. As you recall t_object.Stereotype holds just one single stereotype for an element. But since quite a while EA has this ellipsis button near the stereotype so you can define more than one stereotype. The 'primary' stereotype is still placed in the `Stereotype` column. But the complete list goes into a single `t_xref` record.

Here is the format of such a record. The equal sign means that the contents is that of the table description above:

Column	Value
XrefID	=
Behavior	NULL
Client	=
Description	see below
Link	NULL
Supplier	NULL
Constraint	NULL
Name	*Stereotypes*
Namespace	NULL
Partition	0
Requirement	NULL
Type	*element property*
Visibility	*Public*

For each of the multiple stereotypes a string like the following is appended to the `Description`:

@STEREO;Name=<stereo>;GUID=<guid>;@ENDSTEREO;

Here *<stereo>* is the name of the applied stereotype. A good idea, not to place a semi-colon in one of those multi-stereotypes. Just see what happens if you add one. (Would you call this a bug?)

The *<guid>* is the t_stereotypes.ea_guid of the according stereotype.

9.3 Default Composite Diagrams

Whenever you check the `Advanced/Make Composite` flag for elements EA associates the first diagram inside the element with the element (if there is no diagram present, EA creates one). This connection is persistent even if you move the diagram to somewhere else. While the element composite flag is identified via

t_object.NType == *8*

the according diagram reference is stored in `t_xref`:

Column	Value
XrefID	=
Behavior	NULL
Client	=
Description	*@STEREO...* see above
Client	t_object.ea_guid
Supplier	t_diagram.ea_guid
Constraint	NULL
Name	*DefaultDiagram*
Namespace	NULL
Partition	0
Requirement	NULL
Type	*element property*
Visibility	*Public*

Activity and InteractionOccurrence elements are an exception (it's EA!). Here the t_diagram.Diagram_-ID is stored in t_object.PDATA1 instead of `t_xref`.

9.4 Profile file locations

After so many years Sparx now has implmented to save the file locations for profiles. And where did it end up? Yes, right here. The `Name` *ProfileOptions* holds some XML where the names are stored. Interestingly there's a tagged value *_profile_data* for profile packages which holds the exactly same XML - but empty.

10. API Cross References

This chapter contains a cross reference from table columns to object properties in the API[1].

The references are presented for both directions. As you will notice not all columns map to an API property and vice versa. The table also omits EaCollections as those are a result of a query itself and not a simple column.

I have currently only included the two most important table `t_package` and `t_object`.

The API help tells that `Created` and `Modified` are writeable. However, neither will be saved on an `Update()`. `Created` remains unchanged and `Modified` will be set to the current time stamp. In order to change them you must bypass the API.

10.1 t_package — EaPackage

t_package	EaPackage	EaPackage	t_package
BatchLoad	BatchLoad	Alias	t_object.Alias
BatchSave	BatchSave	BatchLoad	BatchLoad
CodePath	-	BatchSave	BatchSave
CreatedDate	Created	Connectors	-
ea_guid	PackageGUID	Created	CreatedDate
Gen_Notes	-	Diagrams	-
IsControlled	IsControlled	Element	-
LastLoadDate	LastLoadDate	Elements	-
LastSaveDate	LastSaveDate	Flags	PackageFlags
LogXML	LogXML	IsControlled	IsControlled
ModifiedDate	Modified	IsModel	-
Name	Name	IsNamepace	Namespace
Namespace	IsNamepace	IsProtected	Protected
Notes	Notes	IsVersionControlled	-
Package_ID	PackageID	LastLoadDate	LastLoadDate
PackageFlags	Flags	LastSaveDate	LastSaveDate
ParentID	ParentID	LogXML	LogXML
PkgOwner	Owner	Modified	ModifiedDate
Protected	IsProtected	Name	Name
TPos	TreePos	Notes	Notes
UMLVersion	UMLVersion	ObjectType	fixed 5
UseDTD	UseDTD	Owner	PkgOwner
Version	Version	PackageGUID	ea_guid
XMLPath	XMLPath	PackageID	Package_ID
		Packages	-

[1] To find out more about the API have a look in my book Scripting EA.

t_package	EaPackage	EaPackage	t_package
		ParentID	ParentID
		TreePos	TPos
		UMLVersion	UMLVersion
		UseDTD	UseDTD
		Version	Version
		XMLPath	XMLPath

10.2 t_object — EaElement

t_object	EaElement	EaElement	t_object
Abstract	Abstract	Abstract	Abstract
ActionFlags	ActionFlags	ActionFlags	ActionFlags
Alias	Alias	Alias	Alias
Author	Author	AssociationClassConnectorID	
Backcolor	see below: SetAppearance	Author	Author
BoderWidth	see below: SetAppearance	ClassifierID	Classifier
Bordercolor	see below: SetAppearance	ClassifierName	- via Classifier
BorderStyle	see below: SetAppearance	Complexity	Complexity
Cardinality	-	CompositeDiagram	- via t_xref
Classifier	ClassifierID	Created	CreatedDate
Classifier_guid	-	Difficulty	PDATA3
Complexity	Complexity	ElementGUID	ea_guid
Concurrency	-	ElementID	Object_ID
CreatedDate	Created	EventFlags	EventFlags
Diagram_ID	-	ExtensionPoints	- via PDATA1
ea_guid	ElementGUID	GenFile	#EXP# entries GenFile
Effort	-	GenLinks	GenLinks
EventFlags	EventFlags	GenType	GenType
Fontcolor	see below: SetAppearance	Header1/2	Header1/2
GenFile	GenFile	IsActive	IsActive
GenLinks	GenLinks	IsComposite	- via NType == 8
GenOptions	-	IsLeaf	IsLeaf
GenType	GenType	IsSpecification	IsSpecification
Header1/2	Header1/2	Locked	- via t_seclocks
IsActive	IsActive	Metatype	-
IsLeaf	IsLeaf	MiscData(0)	PDATA1
IsRoot	-	MiscData(1)	PDATA2
IsSpecification	IsSpecification	MiscData(2)	PDATA3
ModifiedDate	Modified	MiscData(3)	PDATA4
Multiplicity	Multiplicity	MiscData(4)	PDATA5
Name	Name	Modified	ModifiedDate
NType	IsComposite if NType == 8	Multiplicity	Multiplicity

t_object	EaElement	EaElement	t_object
Object_ID	ElementID	Name	Name
Object_Type	Type	ObjectType	- via Object_Type
Package_ID	PackageID	PackageID	Package_ID
PackageFlags	-	ParentID	ParentID
ParentID	ParentID	Persistence	Persistence
PDATA1	MiscData(0);	Phase	Phase
	Status for Requirements	Priority	PDATA2
PDATA2	MiscData(1);	PropertyType	- via PDATA1
		GUID	
	Priority for Requirements	RunState	RunState
PDATA3	MiscData(2);	Status	Status (and PDATA1)
	Difficulty for Requirements	Stereotype	Stereotype
PDATA4	MiscData(3)	StereotypeEx	- via t_xref
PDATA5	MiscData(4); Tag	StyleEx	StyleEx
Persistence	Persistence	SubType	- via NType/Object_-Type
Phase	Phase	Tablespace	-
RunState	RunState	TreePos	TPos
Scope	Visibility[2]	Type	Object_Type
StateFlags	-	Version	Version
Status	Status	Visibility	Scope
Stereotype	Stereotype		
Style	-		
StyleEx	StyleEx		
Tagged	-		
TPos	TreePos		
Version	Version		
Visibility	-		

`SetAppearance` —

The marked columns appear to be not readable from the API. However, you can set them via the `SetAppearance` method.

[2]Please note that the DB has `Visibility` and `Scope`. But only the DB `Scope` matches the `Visibility` property in the `EaElement`. The DB `Visibility` is just there to confuse the user.

11. Bits and Pieces

This section contains a couple of details for selected columns from where they are referenced. A back reference is included at the end of each chapter. Note that most of the following descriptions do not detail the contents but give a sample of the contents only. For those having dug that far it will be obvious how to decode the contents.

11.1 CSV Lists

A number of columns contain semi-colon separated lists in the format `<key>=<value>;` where these pairs can appear more than once thus forming a list of key-value pairs. Usually `<key>` is alphanumeric including '_' (underscore). Value itself can contain any chars except '=' and ';'.

I found that EA often does not check this constraint and if people enter e.g. a semi-colon in a name it will simply confuse EA in it's later behavior but will not croak[1] that an illegal char is used. Well, it's EA.

11.2 Object Types

This string value should correspond to one of the values in `t_objecttypes.Object_Type`.

The API returns this value a `EAElement.Type` while `EAElement.ObjectType` is a numeric value with different semantic.

Action	EntryPoint	Package
ActionPin	Enumeration	Parameter
Activity	Event	Part
ActivityParameter	ExceptionHandler	Port
ActivityPartition	ExecutionEnvironment	PrimitiveType
ActivityRegion	ExitPoint	ProtocolStateMachine
Actor	ExpansionNode	ProvidedInterface
Artifact	ExpansionRegion	Region
Association	Feature	Report
Boundary	GUIElement	RequiredInterface
CentralBufferNode	InformationItem	Requirement

[1] Just try this with a stereotype. Enter *abc;def* as stereotype. Save, close and re-open the element. Now it shows just *abc*. However, using the ellipsis will show *abc;def* as possible (but unchecked) stereotype. I already reported that as bug years ago. It's still not fixed in V12…

Change	Interaction	Risk
Class	InteractionFragment	Screen
Collaboration	InteractionOccurrence	Sequence
CollaborationOccurrence	InteractionState	Signal
Comment	Interface	State
Component	InterruptibleActivityRegion	StateMachine
ConditionalNode	Issue	StateNode
Constraint	Label	Synchronization
DataStore	LoopNode	Task
DataType	MergeNode	Text
Decision	MessageEndpoint	TimeLine
DeploymentSpecification	Node	Trigger
Device	Note	UMLDiagram
DiagramFrame	Object	UseCase
Entity	ObjectNode	User

From t_object.Object_Type

Running a test on creating elements for all of the above EA told me that `ProtocolStateMachine`, `Label` and `User` are invalid types. I hardly remember where I got the above list from but most likely it was a scan of the example EA project. Seems it did contain some scratch too.

11.3 What is an Instance

Following a thread on the forum I used above list to create instances to see what EA creates. For entries with a "-" in the instance column EA will not create an instance.

Object_Type	instance
Action	-
Activity	Action
ActivityPartition	-
ActivityRegion	-
Actor	Object
Artifact	Artifact
Association	Object
Boundary	-
CentralBufferNode	-
Package	-
Enumeration	Object
Event	-
ExceptionHandler	-
ExecutionEnvironment	ExecutionEnvironment
ExpansionRegion	-
Feature	Object
GUIElement	Object
InformationItem	Object
Package	-
Part	-
PrimitiveType	Object
Region	-

Object_Type	instance
Report	-
Requirement	Object
Change	Object
Class	Object
Collaboration	CollaborationOccurrence
CollaborationOccurrence	-
Note	-
Component	Component
ConditionalNode	-
Constraint	-
Object	-
DataType	Object
Decision	-
DeploymentSpecification	DeploymentSpecification
Device	Device
DiagramFrame	-
Entity	Object
Interaction	Object
InteractionFragment	-
InteractionOccurrence	-
InteractionState	-
Interface	Object
InterruptibleActivityRegion	-
Issue	Object
LoopNode	-
MergeNode	-
MessageEndpoint	-
Node	Node
Note	-
Object	-
Risk	Object
Screen	Object
Sequence	-
Signal	Object
State	-
StateMachine	State
StateNode	-
Synchronization	-
Task	Object
Text	-
TimeLine	-
Trigger	-
UMLDiagram	Object
UseCase	UseCase

According to UML 2.5 the metaclass `Object` does no longer exist. Instead there is `InstanceSpecification`. Sparx however will stay with `Object`, probably because it's just hammered into the code as it is.
I haven't cross-checked the other metaclasses with compliance to the UML specs.

11.4 Concurrency

A string value to specify a concurrency parameter for single elements.

Sequential
Active
Guarded
Synchronous

From t_object.Concurrency

11.5 GUID

A Globally Unique IDentifier which is used to identify elements throughout many repositories.

{XXXXXXXX-XXXX-xxxx-XXXX-XXXXXXXXXXXX}

where XX is a hex code with upper case chars and xx one with lower case. There's a bit of magic in some GUID especially with that of tagged values.

From t_object.ea_guid, t_package.ea_guid, t_diagram.ea_guid, t_connector.ea_guid, t_attribute.ea_guid, t_operation.ea_guid, t_stereotypes.ea_guid, t_objectproperties.ea_guid, t_connectortag.ea_guid, t_xref various, t_secuser.UserID, t_secuserpermission.UserID, t_secusergroup various, t_secgroup.GroupID, t_secgrouppermission.GroupID, t_seclocks various, t_script various

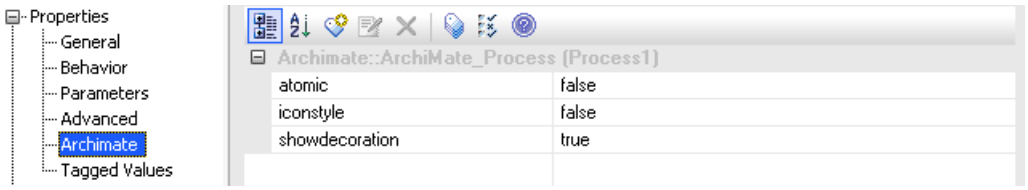
11.5.1 GUID in tagged values

As mentioned above there's some magic with GUIDs for tagged values that come from a MDG profile. These look exactly like 'normal' GUIDS:

{aaaaaaaa-aaaa-aaaa-bbbb-bbbbbbbbbbbb}

Here the `a` and `b` hex codes are not completely random. Only the `b` part is completely random, but the `a` part is a hash code computed individually per profile. The hash algorithm is internal to EA and unfortunately not known.

E.g. if you look at the tagged value `atomic` of the `Process` metaclass in the `Archimate` profile



Atomic Tag

you will find that it looks like

{EAB63B1D-465F-e346-bbbb-bbbbbbbbbbbb}

That means that you can not create tagged values for metatypes out of a MDG profile unless you know the `a`-part of the GUID. The only way to do that is by creating all possible elements and tagged values and then make a static cross reference.

11.5.2 GUID in Package and Element

Rumors have it that t_object.ea_guid and t_package.ea_guid are identical. While this has proven true and spread by a Sparxian initially, another Sparxian said, that this might change without warning in future EA versions[2].

11.6 Object Run State Property

Appears in diagrams for objects. Can be set via `Advanced/Set Run State...` from the object's context menu.

@VAR;Variable=<name>;Value=<val>;Note=<note>;Op=<op>;@ENDVAR

where the tuple Variable/Value/Note/Op appears consecutive for all defined run states of an object. The Note attribute is omitted if the note is empty.

From t_object.RunState

11.7 TPos Property

A numeric value which is part of the sort order for elements and package elements in the project browser. All elements of a package with the same TPos are sorted alphabetically. These groups appear in ascending order of TPos. Additionally EA groups elements according to their type. E.g. Diagram appear always first before any Packages and then elements grouped by type. EA does not set TPos for all elements of a level below a package but only those being moved with the hand icons. You may force your ordering by giving any arbitrary numbering order. This does not override the element type grouping though.

From t_object.TPos, t_package.TPos, t_diagram.TPos

11.8 Object StyleEx Property

Stores an individual font for an element.

font=ARIAL;fontsz=80;bold=0;italic=0;ul=0;charset=0;pitch=0;

[2]Needles to say that this warning applies to most of the stuff published in this book. However, Sparx is **very** conservative when it comes to changes in their database. So I guess it's pretty safe to use when needed.

font:
: Font property

fontsz:
: Size property multiplied by ten

bold:
: Font Style property. *1* if bold, else *0*

italic:
: Font Style property. *1* if italic, else *0*

ul: Underline property

charset, pitch:
: These values are fixed as shown above. It is possible to modify *charset* and it is rendered correctly in the Sample but not in the diagram. Also when saving the properties manually these values are forced to the above defaults.

From t_object.StyleEx

11.9 Package Flags Property

For a view package this contains the string

isModel=1;VICON=<vi>;CRC=<numeric>;

where it is unknown what the CRC means. Anyway it is not recommended to change these values! The *<vi>* is a numerical value according to the chosen View icon (a number between 0 and 5; see context menu `Set View Icon`).

For version controlled packages you will find something like

Recurse=0;VCCFG=<VC>;

where *<VC>* is the unique ID of the appropriate Version Control provider. The complement to that can be found in the text file

```
%APPDATA%\Sparx Systems\EA\paths.txt
```

If you have chosen to in-/exclude a package via the context menu `Documentation/Generated Report Options` you will also encounter

RTF=<bool>;

where *<bool>;* is either *T* or *F* for true or false.

From t_package.PackageFlags

11.10 Diagram PDATA Property

A semi-colon separated list like this one:

> *HideRel=0;ShowTags=0;ShowReqs=0;ShowCons=0;OpParams=1;ShowSN=0;ScalePI=0;*
> *PPgs.cx=1;PPgs.cy=1;PSize=9;ShowIcons=1;SuppCN=0;HideProps=0;HideParents=0;*
> *UseAlias=0;HideAtts=0;HideOps=0;HideStereo=0;HideEStereo=0;FormName=;*

Here are some decoded values:

Value	Diagram/Diagram property
UseAlias	`Use Alias if Available`
HideParents	**not** `Show Additional Parents`
ShowSN	`Show Sequence Notes`

Value	Diagram/Element property
ShowCons	`Show Compartments/Constraints`
ShowIcons	`Use Stereotype Icons`
ShowReqs	`Show Compartments/Requirements`
ShowTags	`Show Compartments/Tags`
HideAtts	**not** `Show Compartments/Attributes`
HideEStereo	**not** `Show Element Stereotypes`
HideOps	**not** `Show Compartments/Operations`

Value	Diagram/Feature property
OpParams	`Show Parameter Detail`
HideProps	**not** `Property Methods`
HideStereo	**not** `Show Stereotypes`

Value	Diagram/Connector property
HideRel	**not** `Show Relatiosnships`
SuppCN	**not** `Show Collaboration Numbers`
HideStereo	**not** `Show Stereotypes`
HideProps	**not** `Property Methods`

The **not** indicates that switch and value are contrary to each other.

From t_diagram.PDATA

11.11 Diagram Swimlanes Property

Something like this one:

> *locked=false;orientation=1;width=0;inbar=false;names=true;color=0;*
> *bold=false;fcol=0;;cls=0;SW1=330;SW2=343;SW3=116;*

Probably not too difficult to decode this but definitely not important enough to be detailed here.

From t_diagram.Swimlanes

11.12 Diagram StyleEx Property

Again something like this:

SaveTag=63BF5A5A;ExcludeRTF=0;DocAll=0;HideQuals=0;AttPkg=1;ShowTests=0;
ShowMaint=0;SuppressFOC=1;MatrixActive=0;SwimlanesActive=1;MatrixLineWidth=1;
MatrixLocked=0;TConnectorNotation=UML 2.1;TExplicitNavigability=0;
AdvancedElementProps=1;AdvancedFeatureProps=1;AdvancedConnectorProps=1;
ProfileData=;MDGDgm=;DefaultLang=Java;STBLDgm=;ShowNotes=0;
VisibleAttributeDetail=0;ShowOpRetType=1;SuppressBrackets=0;
SuppConnectorLabels=0;PrintPageHeadFoot=0;ShowAsList=0;

Here are some decoded values:

Note how consequently inconsequent those properties are scattered amongst PDATA and StyleEx.

Value	Diagram/Diagram property
ShowFQN	`Fully Qualified Namespace`
HandDraw	`Hand Drawn`
Whiteboard	`Whiteboard Mode`
PrintPageHeadFoot	`Print Page Header and Page Footer`
ExcludeRTF	`Exclude image from RTF Documents`
DocAll	`Document each contained element in RTF`
ShowDiagramInPages	`Divide Diagram into Multiple Pages`
RotateImages	`Rotate Images`

Diagram Layout property *Layout* holds a colon-separated list like this:

Layout=l=20:c=20:d=0:cr=0:la=2:i=1:it=4:a=0:

Tag	Meaning	Tag	Meaning
cr	Cycle Remove property	*it*	`Iterations`
	`Greedy= 0`	*a*	`Aggressive`
	`Depth...= 1`	*l*	`Layer Spacing`
la	Layering Options property	*c*	`Column Spacing`
	`Longest Path Sink= 0`	*d*	`Direction`
	`Longest Path Source= 1`		`Up = 0`
	`Optimal Link Length= 2`		`Down = 1`
i	`Initialize Options`		`Left = 2`
	`Optimal Link Length= 2`		`Right = 3`
	`Naïve = 0`		
	`Depth First Search Outward = 1`		
	`Depth First Search Inward = 2`		

Value	Diagram/Element property
AdvancedElementProps	**not** `Show Element Property String`
DefaultLang	`Language`
ShowMaint	`Show Compartments/Maintenance`
ShowNotes	`Show Compartments/Notes`
ShowTests	`Show Compartments/Testing`

Value	Diagram/Feature property
AdvancedFeatureProps	`Show Property String`
AttPkg	`Package`
OverrideLinkedF	`Always Show Linked Features`
ShowOpRetType	`Show Operation Return Type`
SuppressBrackets	`Suppress Brackets for Operation without Parameters`
VisibleAttributeDetail	`Show Attribute Detail`
HideQuals	**not** `Show Qualifiers and Visibility Indicators`

Value	Diagram/Connector property
TExplicitNavigability	`Show Non-Navigable Ends`
AdvancedConnectorProps	**not** `Show Connector Property String`
SuppConnectorLabels	`Suppress All Connector Labels`
HideConnStereotype	**not** `Show Stereotype Labels`
TConnectorNotation	`Connector Notation`
HideQuals	**not** `Show Qualifiers and Visibility Indicators`
AdvancedFeatureProps	`Show Property String`
ShowOpRetType	`Show Operation Return Type`
SuppressBrackets	`Suppress Brackets for Operation without Parameters`
OverrideLinkedF	`Always Show Linked Features`
AttPkg	`Package`
VisibleAttributeDetail	`Name and Type = 0, Name Only = 1`
DLKO	`Freeze Visible`

The **not** indicates that switch and value are contrary to each other.

A value in the form *OPTIONS_<duid>=<colon-list>* is related to the t_diagramobject.ObjectStyle *DUID=<duid>*. The *<colon-list>* appears as *<key>=<value>:*.

This is usually a colon separating multiple key/value-pairs. Of course they also use different styles. Something where other companies would tar and feather you when attmepting it.

And I only decoded the meaning of two keys, namely *InfoView* and *GID*. A couple of other keys are *DOI*, *DAT*, *DTS*, *CCOMP_* (and some crude variants), *BPMN...* (the ellipsis means a number of different variants), *RoadmapSegs* and a many more. I might dig into those if demand will arise.

InfoView=<value>: is used when context menu `Info View/Enabled` has been checked. *<value>* is the sum of the following bits

Value	Description
1	Show info view
2	Type
4	Stereotype
8	Status
32	Version
16	Phase
64	Author

So a value of 7 will show the info view with Type and Stereotype. A value of 6 will hide the info view and `Info View/Enabled` will make it appear with Type and Stereotype.

The second application is used for diagram element grouping and an entry would look e.g. like

OPTIONS_CA81B41F=GID=0C08:;OPTIONS_D744D373=GID=0C08:;

The four hex bytes after the *GID=* are (as far as I found out) an arbitrary code which denotes the group. The above snippet would therefore mean a group of two diagram elements.

Another example where Sparxians added a feature and using a crowbar to fit it in the existing database.

From t_diagram.StyleEx

11.13 DiagramObject ObjectStyle Property

Even more semi-colon stuff:

Dockable=on;DUID=BA09D7A8;VPartition=1;LCol=0;BCol=14938876;BFol=0;
font=ARIAL;fontsz=80;bold=0;italic=0;ul=0;charset=0;pitch=0;

A part of these are detailed in Sparx' automation object reference. Here are some more details, related to font settings:

Name	Meaning
font	Name of the font for text in the element
fontsz	Font size in pt times 10
ul	Text is underlined
bd	Text is bold
charset	Western=0, Hebrew=177, Arabic=178, Greek=161, Turkish=162, Baltic=186, Central European=238, Cyrillic=204, Vietnamese=163
pitch	No idea. Is set to 34 when the label color is changed?!

For "regular" diagrams the following settings can be found:

Name	Meaning
DUID	A Diagram UID used for internal references, e.g. for the grouping of elements
HDN	For Ports indicating the visibility of its label
VPartition	1 if a Partition is shown vertically
Dockable	on/off corresponding to the context menu setting
NSL	1/0 corresponding to the context menu setting Selectable
UCRect	1 to show either `Action`, `Activity` or `UseCase` elements in rectangular notation.
ActRect	1 to show an 'Actor element in rectangular notation.

And finally for sequence diagrams you can encounter:

Name	Meaning
AOC	1/0 corresponding to the context menu Advanced/Activate On Create

From t_diagramobject.ObjectStyle

11.14 Connector SubType Property

The following Values may appear for specific Connector_Types to detail specific connector attributes.

Connector_Type	Values
Aggregation	*Weak* -> Shared
	Strong -> Composite
Association	Sometimes *Class* ?
Sequence	Message Lifecycle Property
	New
	Delete
Sequence	Timing Diagram Message
	Timing
UseCase	Association show as stereotyped dependency when it has non-null values: *Extends* *Includes*

For sequence messages which are linked to an operation there is an according tagged value `operation_guid` which corresponds to t_operation.ea_guid.

From t_connector.SubType

11.15 Connector Direction Property

One of the following string values:

Unspecified
Bi-Directional

Source -> Destination
Destination -> Source

From t_connector.Direction

11.16 Connector PDATA5 Property

A semi-colon separated attribute list like that below. Parts of these are documented in EA's object reference.

SX=0;SY=0;EX=0;EY=0;$LLB=;LLT=;
LMT=CX=134:CY=39:OX=106:OY=-32:HDN=0:BLD=0:ITA=0:UND=0:CLR=-1:ALN=0:DIR=0:ROT=0;
LMB=;LRT=;LRB=;IRHS=;ILHS=;

The rest is up to the willing reader to decode.

From t_connector.PDATA5

11.17 Connector StateFlags Property

Something like the following:

Activation=0;ExtendActivationUp=1;Initiate=0;StartCoregionHead=0;EndCoregionHead=0;
StartCoregionTail=0;EndCoregionTail=0;StopActivation=1;EndActivation=1;

From t_connector.StateFlags

11.18 Connector StyleEx Property

For certain connector types the contents of the `StyleEx` property can take certain values.

Connector_Type	Values
Transition	*alias=<Alias property>;* The Alias property
Sequence	*aliasparamsTO=<return>;* where *<return>* is the Return Value property *paramvalues=<argument>;* where *<argument>* is the Argument(s) property *SEQDC=<val>;* where *<val>* Duration Constraint *SEQDO=<val>;* where *<val>* Duration Observation *SEQTC=<val>;* where *<val>* Timing Constraint *SEQTO=<val>;* where *<val>* Timing Observation *DCBM=<val>;* where *<val>* Duration Constraint Between Messages

Connector_Type	Values
Association	*DCBMGUID=<guid>;* where *<guid>* is the t_connector.ea_guid of the related connector *LF<dir>P=<guid><pos>;* connector is attached to attribute/operation *<dir>* = *S* or *E* meaning Start (source) or End (target) *<guid>* = ea_guid of t_attribute or t_operation *<pos>* is the edge (*L* or *R*) where the connector had been attached to in the moment when the link has been created. This is a superfluous information since the renderer will attach the link to whatever place is relevant. There can be one *LFSP*, one *LFEP* or both be present in one `StyleEx` property

From t_connector.StyleEx

11.19 Binary Data

Various columns contain binary data which are additionally zipped. If you want to access the contents you have to follow these steps:

- Decode the column using base64 (if you retrieved the data via `Repository.SQLQuery()`.
- Unzip the resulting binary blob.
- Extract the file `str.dat` from the unzipped data.

This file finally contains the data.

An exception is Image which yields the raw image data right after decoding it via base64.

11.20 RGB Values

Coloring values in EA have simple RGB values without transparency. Each red, blue and green value can take an intensity from 0 to 255. The RGB itself is calculated by

```
RGB = (blue * 256 + green) * 256 + red
```

The default coloring is equal to -1 (black).

11.21 Boundaries and Placeholders

As I only recently found out EA will create placeholders for missing imports. That is if you import a package via XMI that references elements outside that package. Also you need to have turned on the *Create placeholders* option in Start/Preferences/XML...[3]. In that case you will find `Boundary` elements which have `t_objecttypes.NType` set to 1001 (greets from Aladdin) and `t_objecttypes.Note` holding a short semi-colon separated list with the name and type of the external element. Usually these elements will not appear in diagrams, but if you place them they will look like:

Placeholder

Here the `Note` contains *Name=bla;Type=Class;*. You will notice that these boundaries are not meant to be used on diagrams.

[3]Where ever that might be in your EA version. I was told that the upcoming V16 has found another place for it.

12. User Settings

There are a couple of locations where EA stores information locally per user. These can be found in

- Registry
- %APPDATA%Sparx SystemsEA
- %PROGRAMFILES%Sparx SystemsEA

These locations are mainly used to hold user options (those from `Tools/Options`), Layout information, MDG data and more.

12.1 Registry

In former EA versions the registry keys were just a handfull, but now there are tons. The EA relevant keys are found at

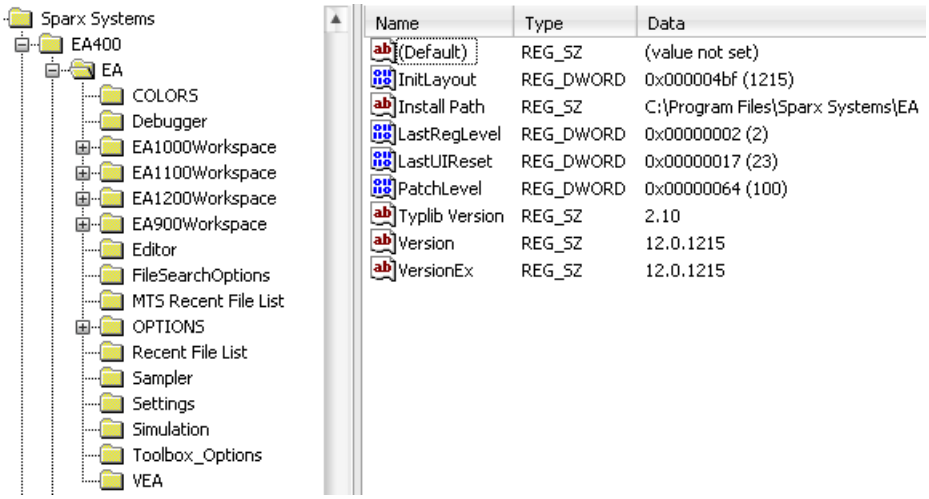
- `HKEY_CURRENT_USER\Software\Sparx Systems\EA400`
- `HKEY_CURRENT_USER\Software\Sparx Systems\EAAddins`
- `HKEY_LOCAL_MACHINE\SOFTWARE\Sparx Systems\EAAddins`

Obviously the `EAAddins` keys are used for addins and the usualy suspects will know how to deal with them.

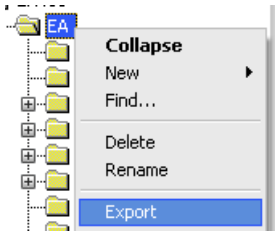
The first key contains all the marvelous settings which EA needs to show up. Quite some of the `Tools/Options` are hidden here besides all the layout stuff. So when you need to find certain settings and don't find any of the files being changed you should export the EA registry part, change the settings and export the registry again. Now you can compare the exports (e.g. with `WinMerge`).

If you need to change any of `Tools/Options` you likely have to go through the registry. And in order to take the changes effect you have to restart EA. I can't go into details for each single option but instead give you a simple receipt how to find the right one in case you need it.

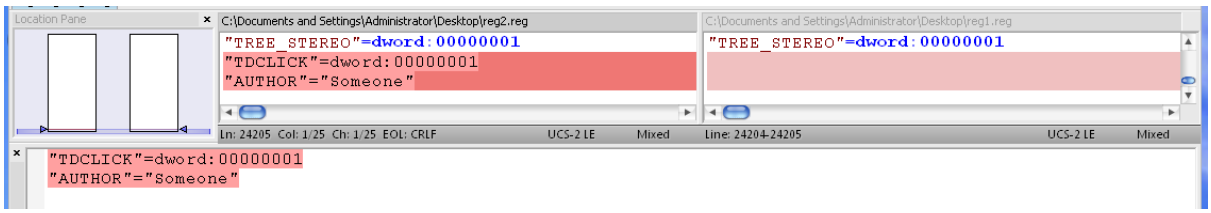
As an example we want to change the `General/General/Author` and `General/General/Double Click`. Per default they are set to *Administrator* and *Shows Properties* on my machine. So the first thing to do is to locate them. You can simply open the registry by entering `regedit` in the command line of the Windows `Start` menu. Next navigate to `HKEY_CURRENT_USER\Software\Sparx Systems\EA400\EA` which should look like this:



Now you need to export the current contents of the `EA` branch:



In order to spot the difference the options must be changed. Setting the author to *Someone* and checking *Opens Branch* will modify the registry. The `EA` branch must be saved once more to another file. When comparing the two exports with WinMerge[1] you can easily spot the differences.



Obviously the key `Author` holds the `Author` option as plain string and `TDCLICK` is an enumeration[2]. Since both values were added as new they will have defaults when not in the registry. So to get the default back you would need to remove the keys. In case of the `TDCLICK` you could also set it to *0*.

Using the above schema you should be able to figure out any option/key pair.

12.2 APPDATA

Quite a number of individual settings are stored in the folder `%appdata%\Sparx Systems\EA`. Here is a description of the most important ones:

[1] http://winmerge.org

[2] You would find that out when comparing after setting `Double Click` to *Open Branch & Diagram* which would result in a value of *2* for `TDCLICK`.

paths.txt
: This file stores the local paths you define from the context menu `Package Control/Version Control Settings` of packages and/or `Tools/Local Directories and Paths...`. The file itself is a plain text file that contains a semicolon-separated list for each definition.

  1. Fixed string *%PATH%*
  2. *type=<type>* where *<type>* is one of *Version Control, XMI* or any of the languages defined. See the `Type` dropdown.
  3. *id=<id>* where *<id>* is the `ID` from the dialog.
  4. *path=<path>* where *<path>* is the `Path` field from the dialog.

  Modifications of this file take direct effect as EA reads the file on each invocation (e.g. in version control or when working with code generation).

DBError.txt
: This file can (!) contain a message from the RDBMS giving a hint in the case EA fails for some reason (usually when you use scripting). E.g. the following content

```
23.11.2018 07:32:54
DAO.Database [3075]

Syntax error in string in query expression 'stereotype = '; ' " ()' AND AppliesTo\
 = 'class''.

Context:
        UPDATE t_stereotypes SET stereotype = 'a', AppliesTo = 'class' WHERE stereotype \
= '; ' " ()' AND AppliesTo = 'class'
```

was placed after trying to add a new stereotype with *; ' " ()* which caused EA to complain. Though the SQL does not help to find the cause (which is simply some badly coded error handling in EA) it can give you often some hints. Especially when you have coded some script which (in-)directly causes creation of invalid SQL.

Paths.txt
: contains settings for local paths and for version control. This file can be manipulated with any text processing. The format is pretty much straightforward:

  `%PATH%;type=<type>;id=<id>;path=<path>;`

  for local paths. Similarly the values for version control are stored here. Just setup version control for a package manually and observe what EA places in this file.

There are also lots of folders which might be interesting when distributing EA configurations.

- The `Workspace` folders simply appear to contain exports of the registry which replace settings to determine EA's workspace settings per user.
- `User Views` represents the `View/Model Views` in XML format.

- `Search Data` stores the user defined searches in XML format.

The rest of those folders is up to you if you feel the need for looking into them.

key.dat
: This contains the license which enables EA to run in full mode rather than evaluation mode.

***.tlx**
: Some spelling paraphernalia.

EAExample.eap
: The example model which is opened from `Help/Open Example Model`. This appears to be a dynamic copy of the same file in the `PROGRAMFILES` folder.

Undo.eaworkspace
: The workspace settings which are used for the `Undo` menu option.

UserPerspective*.xml
: Settings from `View/Perspectives/Perspective Sets`.

Editor Configuration
: This folder contains a number of *.properties* files which appear as some CSS-like configuration to highlight the syntax for different files in EA's internal code editor. You could easily alter the colors but it has tons of other options which can not be detailed here.

Forum History
: Guess what. I rarely use this feature so can not tell much about it except that it contains XML files named after GUIDs.

RTF Templates
: The file *normal.rtf* is used when you create a new linked document in EA and choose `None` as Template.

Search Data
: *EA_Search.xml* contains a XML export of the user defined searches. This is the same what you get when you use the `Documentation/Export` menu from the search view. This file can be used to distribute user defined searches.

User Views
: Definitions from `View/Model Views`.

Workspace Layouts*
: These folders contain proprietary registry exports of the `...\EA\EA*WORKSPACE` keys. It is not recommended to manipulate these files. But you can well distribute copies. The name of the file is visible in the selection menu `View/Perspectives/Workspaces`.

12.3 PROGRAMFILES

The files stored here are mostly meant for runtime and changes are not recommended — at least for most of them. I will only mention a few of the files and folders which might be of common interest.

EABase.eap
: Whenever you create a new EAP file it will be copied from this file. So if you intend to have some default glossary or other reference data you can replace this file with your own modification.

Books
: This folder contains the resources for the `Learning Center`.

DocTemplate
: All the RTF document template which are available under the `System Templates` selection. You can not modify any of the names since menu and file structure are connected hard coded. It's possible to alter the files themselves, though.

MDGTechnologies
: This folder contains all the MDGs you can see under `Extensions/MDG Technologies`. You may place your own MDG here, though that is not recommended. However, to get a clean install for your environment you may remove MDGs from here you do not want to appear for team members.

Sripts
: All the `Local` scripts you see under `View/Scrpiting`. You may add your own files here and they will be listed as any other file in the menu.

12.4 Multiple EAs in Parallel

When you navigate to `%programfiles%\Sparx Systems` you find the folder `EA` and eventually `EA Lite` which contains EA's executables in respective formats. A recurrent question found on Sparx' forum is: "How can I install two EA versions in parallel?". The answer is simply "By following these steps":

- Rename the `EA` folder to what the current release is (e.g. to `EA12`).
- Run the installer of the new version as if it were a normal installation.
- Now you have the new installation in `EA` and the Start menu will refer to this (since it looks in the `EA` folder.
- If you intend to use the new installation as experimental, just rename it to what you want (e.g. to `EA13`) and rename the other folder back to `EA`. Now your installation is back to normal. To start the newly installed EA version, just launch the `EA.exe` in the new folder.

So, is there any danger in doing so. Usually (!) not. Sparx is very conservative in database changes and most of it is backward compatible to V3 (we are now at 14 after almost 20 years). The worst that so far happend were a few things that just worked in the new version and as long as you did not touch them you were still able to run both versions concurrently without any issue. Even using those new features did not kill older installations. That being said: try it in a sandbox first, read the release notes and ask on the forum[3].

Inside the `EA` folder you will find the `EA.exe`, lots of `DLLs`, no longer EA help (that has been migrated to the all-knowing dump called Internet starting with V12) and a couple of folders (in contrast to help getting more over time). Most of the folders contain patterns used in various contexts and you should not have a need to manipulate them with the exception of `MDGTechnologies`. The files herein appear in the `MDG Technologies` settings and users can turn them on/off individually. In many cases you don't want them distributed in a specific environment (to not confuse users and to not polute models with elements that shall not be used). So if you have to create a local EA distribution you should remove all those profiles you will not use.

[3] I for myself have EA versions back to 9.3 and I know that the Sparx supporters still have even older versions installed.

13. GUI References

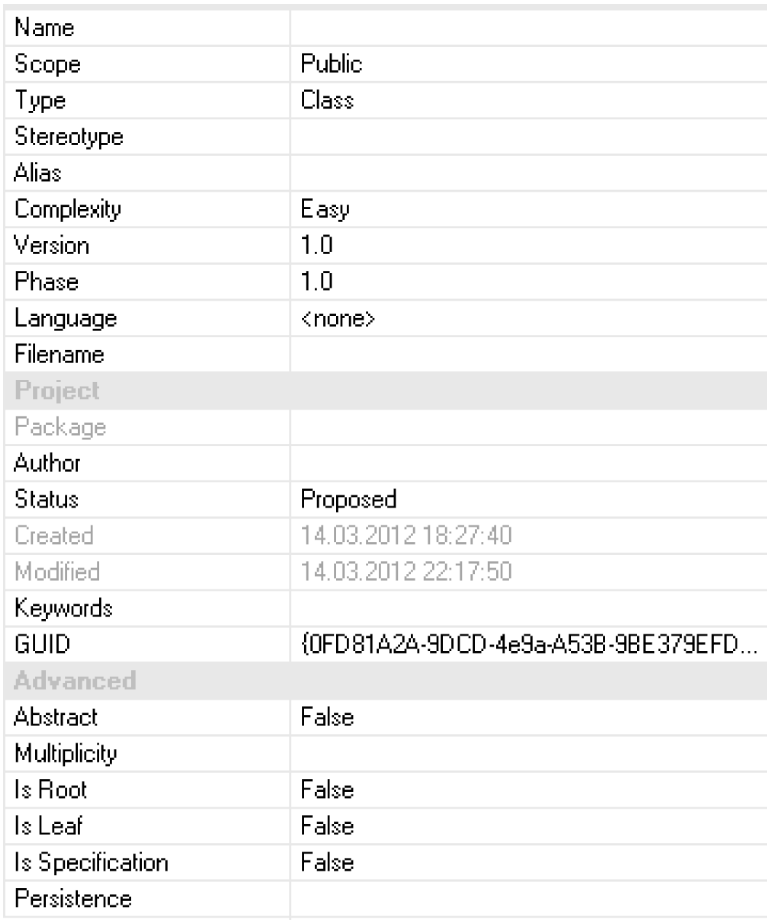
This section contains snapshots of various property windows. Many properties are highlighted with a red rectangle. For those a reference into the according table if given below the snapshot. In most cases the name of the property and the column name are identical. Some have just an additional blank for user readability. However, some properties differ from the column name. In that case the according user readable label is specified in parentheses right after the column name.

Recent versions of EA received a redesign of quite some property windows. I have updated the screenshots in this section to reflect the layout in EA V12. However, there are a few windows that have not changed substantially and are still shown as those from former EA versions. That is where e.g. only the window tabs were moved from top to left.

13.1 Element

Element related property windows also apply to packages due to the dualism of packages being also elements.

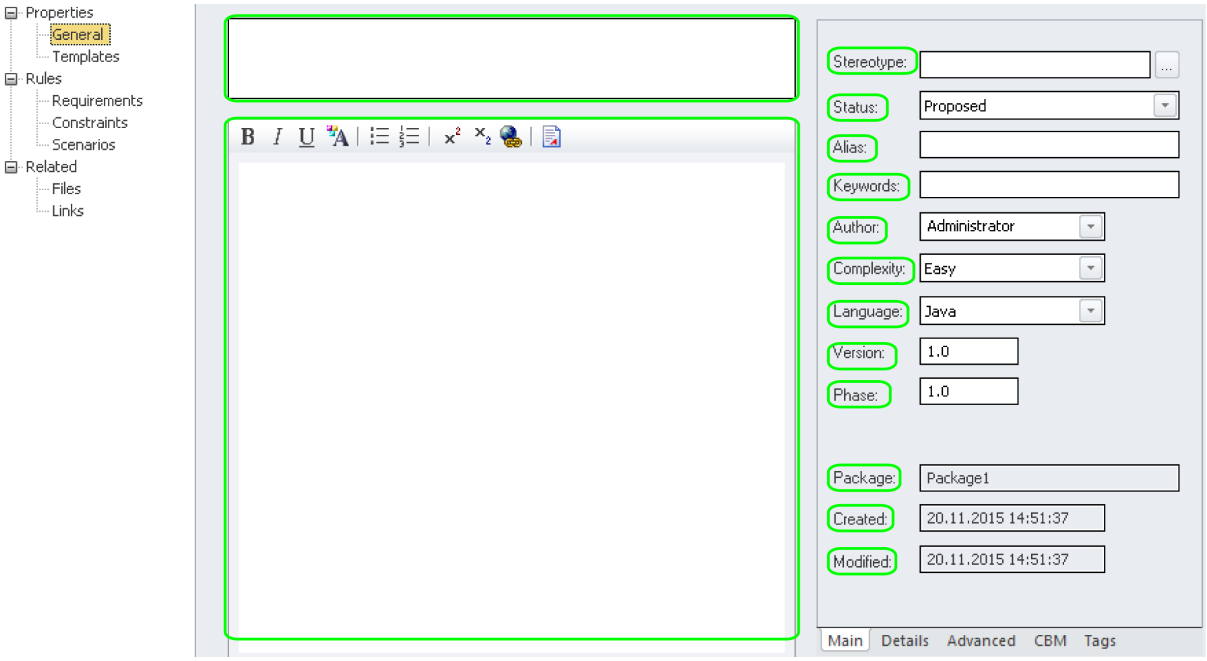
13.1.1 Dockable Properties Window



References:

Label	Column	Label	Column
`Name`	t_object.Name	`Status`	t_object.Status
	t_package.Name	`Created`	t_object.CreatedDate
`Scope`	t_object.Scope		t_package.CreatedDate
`Type`	t_object.Object_Type	`Modified`	t_object.ModifiedDate
`Stereotype`	t_object.Stereotype		t_package.ModifiedDate
`Alias`	t_object.Alias	`Keywords`	t_object.PDATA5
`Complexity`	t_object.Complexity	`GUID`	t_object.ea_guid
`Version`	t_object.Version		t_package.ea_guid
`Phase`	t_object.Phase	`Abstract`	t_object.Abstract
`Language`	t_object.GenType	`Multiplicity`	t_object.Multiplicity
`Filename`	t_object.GenFile	`Is*`	t_object.Is*
`Author`	t_object.Author	`Persistence`	t_object.Persistence
`Package`	t_object.Name (Name of parent package)		

13.1.2 General/Main Properties Window



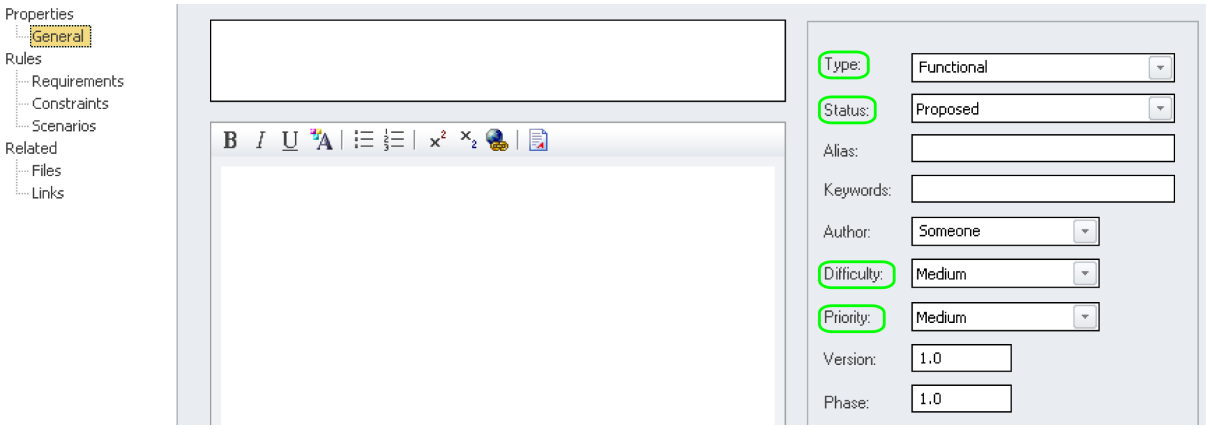
References:

Label	Column
`Name`	t_object.Name / t_package.Name
`Notes`	t_object.Notes / t_package.Notes
`Version`	t_object.Version / t_package.Version
`Alias`	t_object.Alias
`Author`	t_object.Author
`Version`	t_object.Version
`Stereotype`	t_object.Stereotype
`Keywords`	t_object.PDATA5
`Complexity`	t_object.Complexity
`Status`	t_object.Status
`GenType`	t_object.GenType
`Phase`	t_object.Phase
`Package`	t_object.Package_ID -> t_package.Name
`Created`	t_object.CreatedDate
`Modified`	t_object.ModifiedDate

Note that some properties are ambiguous as they appear in both t_object and t_package for package elements.

13.1.3 Requirement Element/General/Main Properties Window

The properties for requirement/issue elements appear almost identically to ‘normal’ element. But a few properties re-mapped differently. All non-marked properties are the same as above.

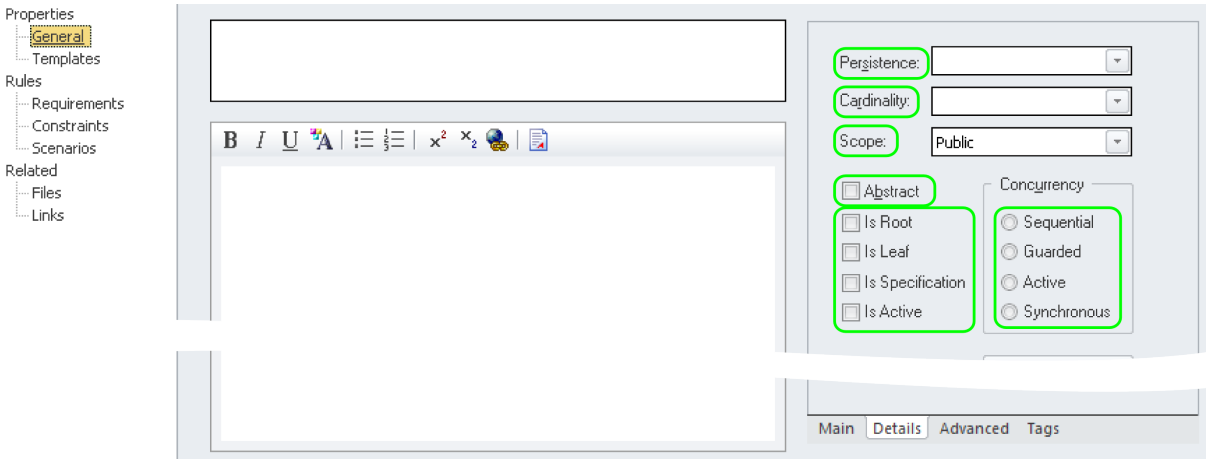


References:

Label	Column
`Type`	t_object.Stereotype
`Status`	t_object.PDATA1
	(appears duplicate in t_object.Status)
`Difficulty`	t_object.PDATA3
`Priority`	t_object.PDATA2

Other fields are identical to that in the Element General Properties Window[1].

13.1.4 General/Details Properties Window



References:

Label	Column
`Scope`	t_object.Scope
`Persistence`	t_object.Persistence
`Cardinality`	t_object.Cardinality
`Abstract`	t_object.Abstract
`Is*`	t_object.Is*
`Concurrency`	t_object.Concurrency

[1] Only the Great Sparx knowns why `Requirements` have a `Requirements` tab.

`Collection Classes` not yet decribed.

13.1.5 General Properties Window/Collection Classes

The properties in this window (left) will be mapped to tagged values (right) as soon as non-blank values are supplied. Any blank values result in deleting the according tagged value:

I have no idea what the `Language` property means or how to modify that.

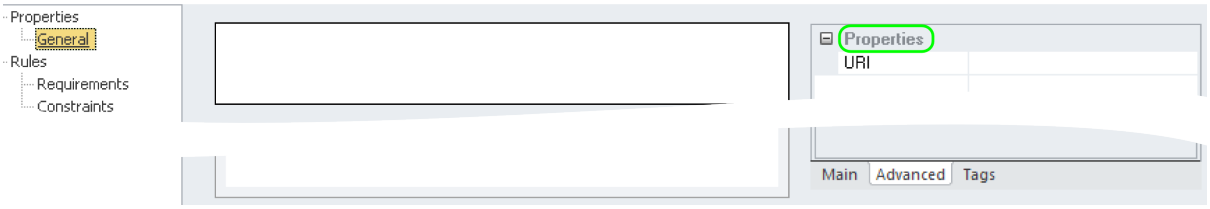
References:

Label	Column
`Scope`	t_object.Scope
`Persistence`	t_object.Persistence
`Cardinality`	t_object.Cardinality
`Abstract`	t_object.Abstract
`Is*`	t_object.Is*
`Concurrency`	t_object.Concurrency

`Collection Classes` not yet decribed.

13.1.6 General/Advanced Properties Window

The properties which appear in the `Advanced` tab for some elements are mapped to t_xref.



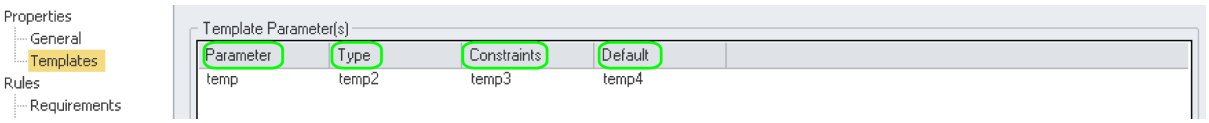
One basic row per property is created with the following mappings:

Column	Value
Name	_ CustomProperties_
Type	*element property*
Description	a CSV list starting with *@PROP* (see below)
Client	t_object.ea_guid of the element

@PROP Key	Value
@NAME	property name
@TYPE	*String, Boolean* and other values
@VALU	property value
@PRMT	empty

13.1.7 Templates Properties Window

The properties in this window do not have a direct property in the database. Instead they are mapped to t_xref.



One basic row is created with the following mappings:

Column	Value
Name	*OwnedMembers*
Type	*element property*
Behavior	some GUID
Description	a CSV list starting with *@ELEMENT* (see below)
Client	the GUID from t_object.ea_guid

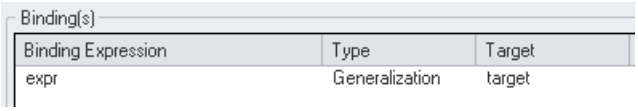
@ELEMENT Key	Value
GUID	the GUID from the Behavior column[2]
Name	the Parameter column
Type	*ClassifierTemplateParameter*
ParameteredElementType	the Type column
ConstrainingClassifierName	the Constraints column
DefaultName	the Default column

If you defined either Constraints or Default column with the ellipsis button, EA will create a link to the selected element also in t_xref.

Column	Value
Name	*MOFProps*
Type	*ownedelement property*
Behavior	*constrainingClassifier*
Description	the GUID from t_object.ea_guid of the linked element
Client	same as Description
Supplier	the GUID from the Behavior of *OwnedMembers*

[2] If you want to understand recursion you need to understand recursion.

The `Bindings` in the lower part of the `Templates` window maps to a CSV list in t_object.PDATA3.

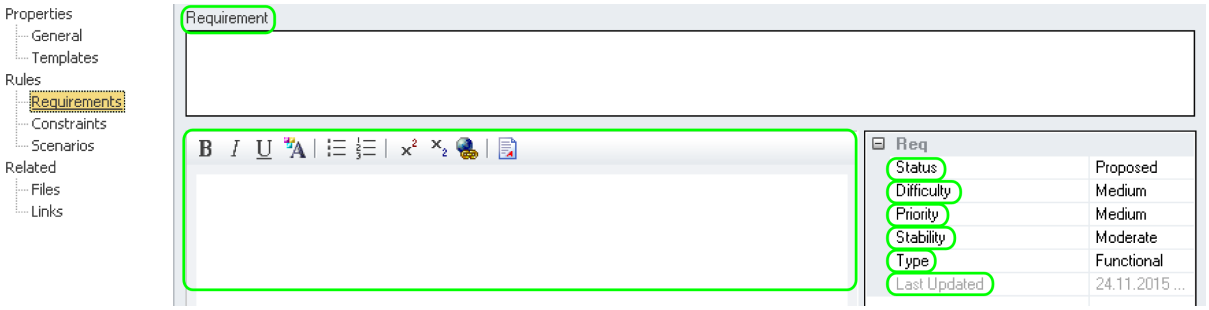


It appears to have only a single value named *ARGS* which defines a comma separated list. This list is repeated for as many entries are defined and simply separated by comma.

Key	Column
`Actual`	`Binding Expression`
`Parent`	`Target`
`BindType`	`Type` as mapped in table below

Key	Drop down
`Gen`	Generalization
`Real`	Realization
`Bind`	Template Binding

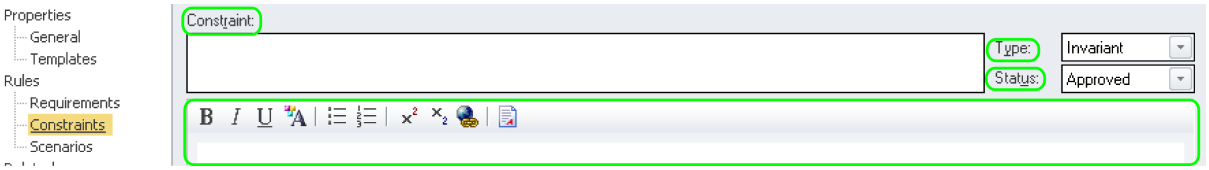
13.1.8 Requirements Properties Window



References:

Label	Column
`Name`	t_objectrequires.Name
`Notes`	t_objectrequires.Notes
`Status`	t_objectrequires.Status
`Difficulty`	t_objectrequires.Difficulty
`Priority`	t_objectrequires.Priority
`Stability`	t_objectrequires.Stability
`Type`	t_objectrequires.ReqType
`Last Updated`	t_objectrequires.LastUpdate

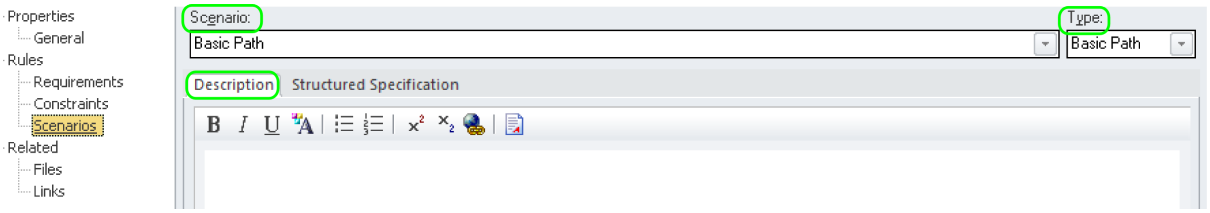
13.1.9 Constraints Properties Window



References:

Label	Column
Name	t_objectconstraints.Name
Notes	t_objectconstraints.Notes
Status	t_objectconstraints.Status
Difficulty	t_objectconstraints.Difficulty
Priority	t_objectconstraints.Priority
Stability	t_objectconstraints.Stability
Type	t_objectconstraints.ReqType
Last Updated	t_objectconstraints.LastUpdate

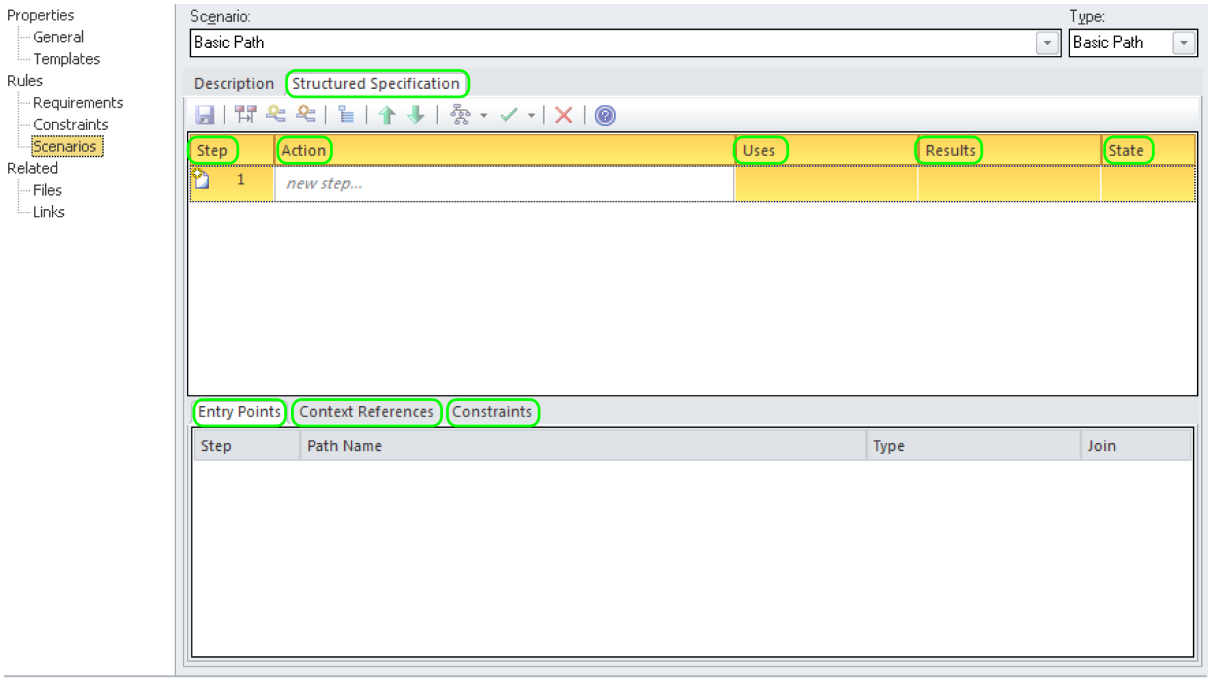
13.1.10 Scenarios/Description Properties Window



References:

Label	Column
Scenario	t_objectscenarios
Type	t_objectscenarios
Description	t_objectscenarios

13.1.11 Scenarios/Structured Properties Window



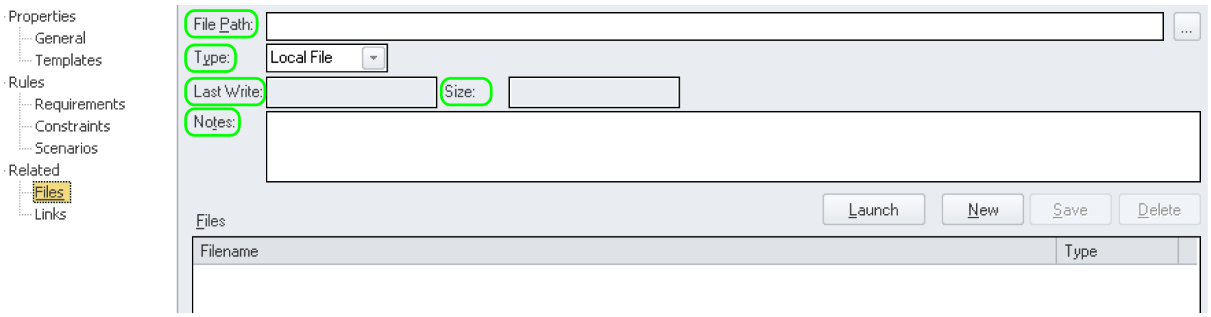
References:

Label	Column
`Structured Specification` and others except `Context References`	t_objectscenarios.XMLContent which maps to t_xref (see below)

Any `Context Reference` creates a row in t_xref with the following content:

Column	Value
`Name`	*Element*
`Type`	*reference*
`Client` and `Supplier`	t_object.ea_guid of the element

13.1.12 Files Properties Window

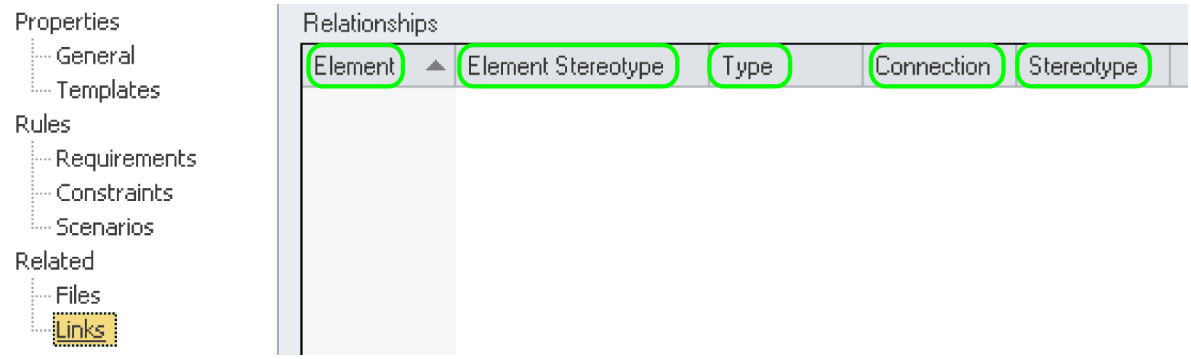


References:

Label	Column
File Path	t_objectfiles.FileName
Type	t_objectfiles.Type
Last Write	t_objectfiles.FileDate
Size	t_objectfiles.FileSize
Notes	t_objectfiles.Note

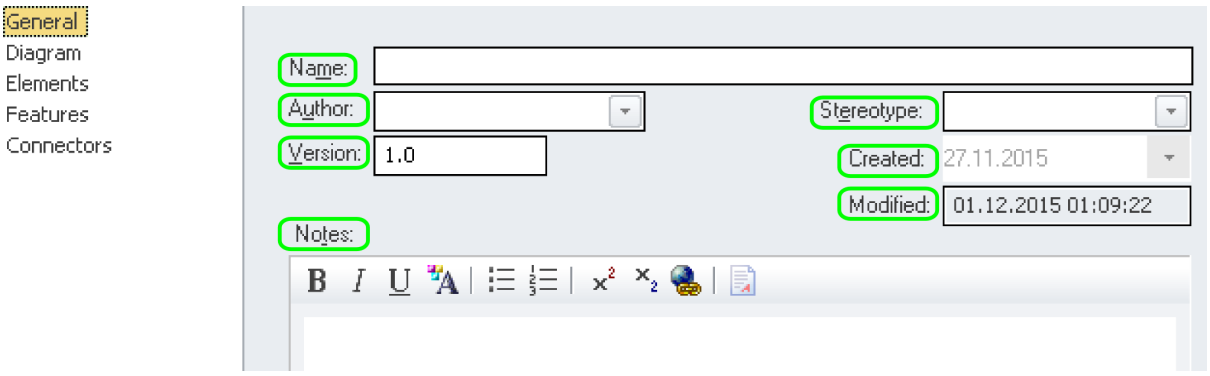
13.1.13 Links Properties Window

The connectors shown in this window have either t_connector.Start_Object_ID or t_connector.End_Object_ID equal to t_object.Object_ID or the element. Element and Element Stereotype are name and stereotype of the element at the other side of the connector. Connection equals t_connector. Connector_Type and Stereotype t_connector.Stereotype.



13.2 Diagram

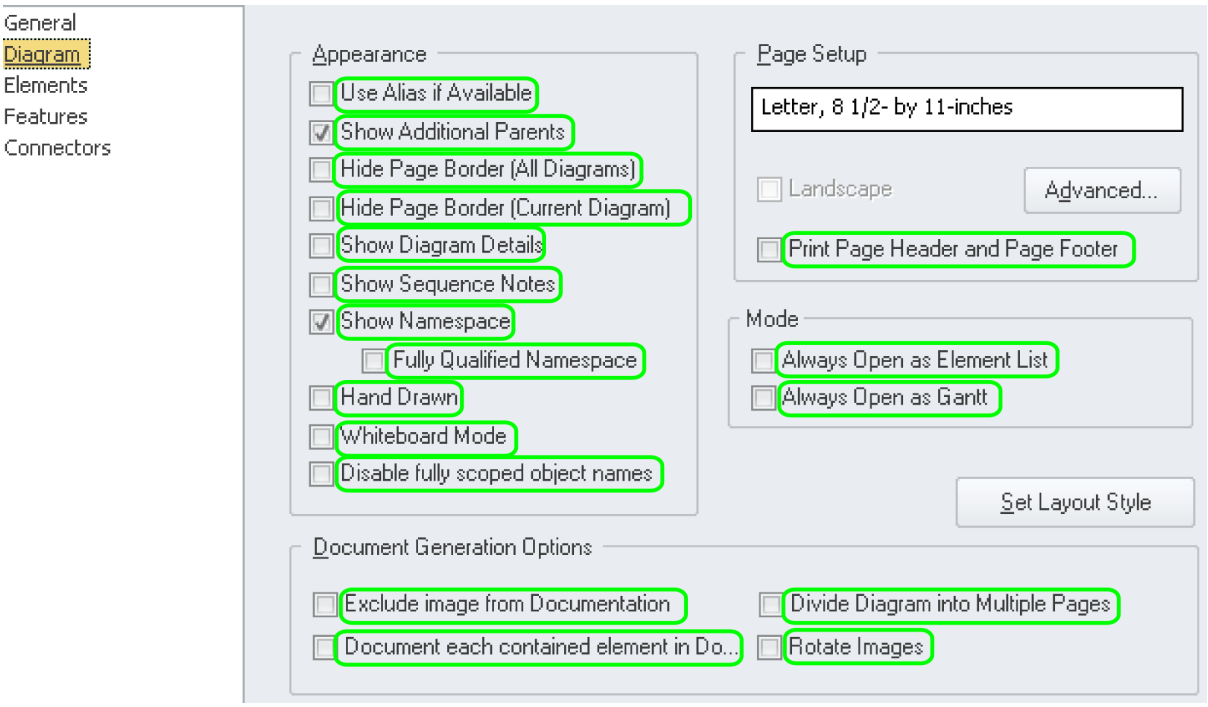
13.2.1 General Properties Window



References:

Label	Column
Name	t_diagram.Name
Author	t_diagram.Author
Version	t_diagram.Version
Notes	t_diagram.Notes
Stereotype	t_diagram.Stereotype
Created	t_diagram.CreatedDate
Modified	t_diagram.ModifiedDate

13.2.2 Diagram Properties Window



References:

Label	Column
Appearance/Show Page Border	t_diagram.ShowBorder
Appearance/Show Diagram Details	t_diagram.ShowDetails
Appearance/Show Namespace	t_diagram.ShowForeign

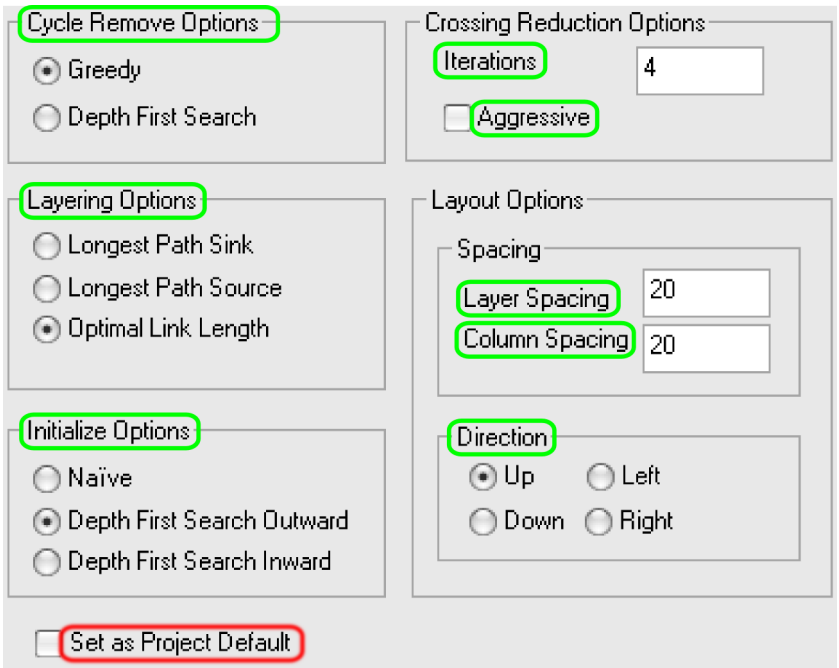
From t_diagram; for details see Diagram PDATA Property and Diagram StyleEx Property:

Label	Column	CSV Tag
`Use Alias if Available`	PDATA	*UseAlias*
`Show Additional Parents`	PDATA	**not** *HideParents*
`Show Sequence Notes`	PDATA	*ShowSN*
`Fully Qualified Namespace`	StyleEx	*ShowFQN*
`Hand Drawn`	StyleEx	*HandDraw*
`Whiteboard Mode`	StyleEx	*Whiteboard*
`Disable fully scoped object names`	StyleEx	*NoFullScope*
`Print Page Header and Page Footer`	StyleEx	*PrintPageHeadFoot*
`Always Open as Element List`	StyleEx	*ShowAsList*
`Exclude image from RTF Documents`	StyleEx	*ExcludeRTF*
`Document each contained element in RTF`	StyleEx	*DocAll*
`Divide Diagram into Multiple Pages`	StyleEx	*ShowDiagramInPages*
`Rotate Images`	StyleEx	*RotateImages*

`Hide Page Border (All Diagrams)` goes to the registry `HKEY_USERS\\Software\\Sparx Systems\\EA400\\EA\\OPTIONS\\HIDEBORDERS`.

`Always Open as Gantt` somehow goes to the registry. If you **really** need this: good luck!

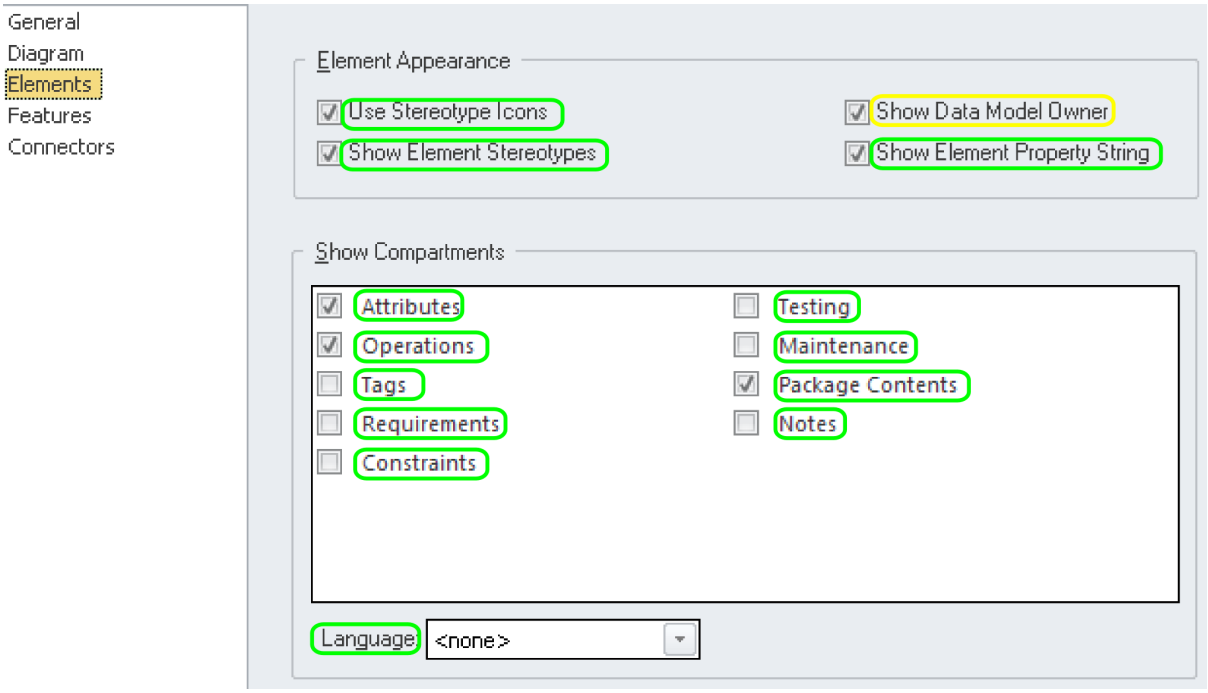
13.2.3 Layout Properties Window



References: Values are stored in the *Layout* attribute of t_diagram.StyleEx.

Label	Tag	
Cycle Remove Options	*cr*	
	Greedy=0	
	Depth...=1	
Layering Options	*la*	
	Longest Path Sink= 0	
	Longest Path Source= 1	
	Optimal Link Length= 2	
Initialize Options	*i*	
	Naïve = 0	
	Depth First Search Outward = 1	
	Depth First Search Inward = 2	
Iterations	*it*	
Aggressive	*a*	0, 1
Layer Spacing	*l*	
Column Spacing	*c*	
Direction	*d*	
	Up = 0	
	Down = 1	
	Left = 2	
	Right = 3	

13.2.4 Elements Properties Window



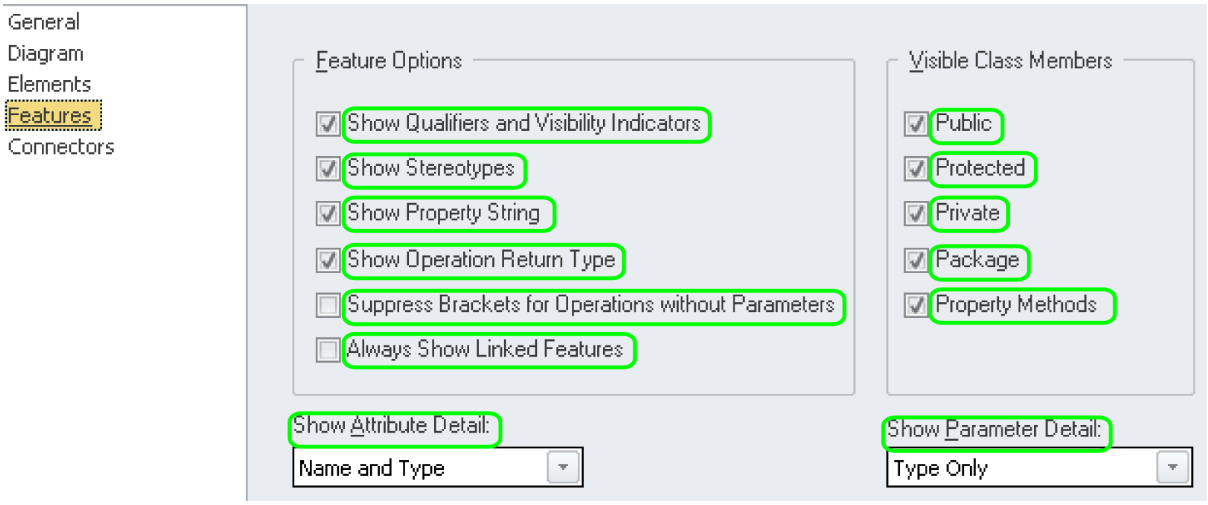
References: t_diagram.ShowPackageContents (Package Contents)

From t_diagram

Label	Column	CSV Tag
Use Stereotype Icons	PDATA	*ShowIcons*
Show Element Stereotypes	PDATA	**not** *HideEStereo*
Show Table Owner	?!	
Show Element Property String	PDATA	**not** *AdvancedElementProps*
Show Compartments/Attributes	PDATA	**not** *HideAtts*
Show Compartments/Operations	PDATA	**not** *HideOps*
Show Compartments/Tags	PDATA	*ShowTags*
Show Compartments/Requirements	PDATA	*ShowReqs*
Show Compartments/Constraints	PDATA	*ShowCons*
Show Compartments/Testing	StyleEx	*ShowTests*
Show Compartments/Maintenance	StyleEx	*ShowMaint*
Show Compartments/Package Contents	ShowPackageContents	
Show Compartments/Notes	StyleEx	*ShowNotes*
Language	StyleEx	*DefaultLang*

Show Table Owner goes to the registry HKEY_USERSSoftwareSparx System-sEA400EAOPTIONSSHOW_TABLE_OWNER.

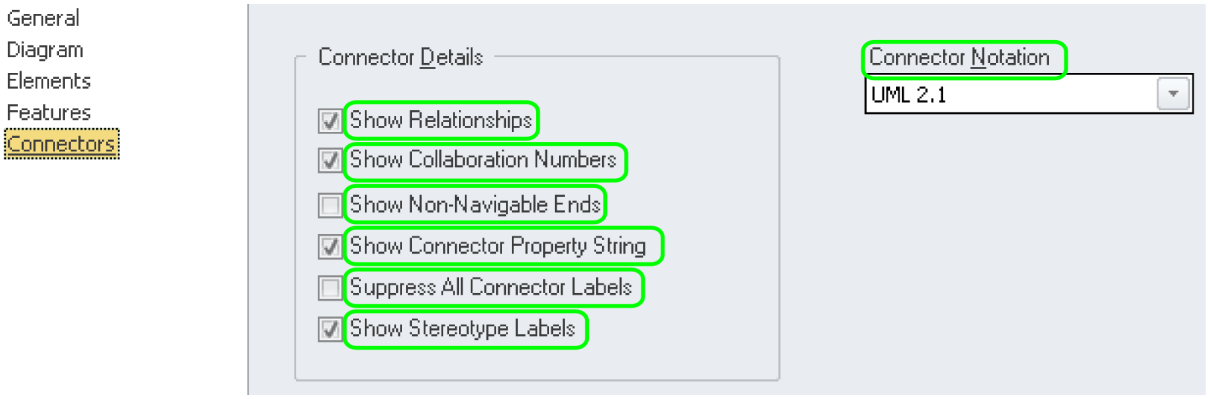
13.2.5 Features Window



References: From t_diagram

Label	Column	CSV Tag
Show Qualifiers and ...	StyleEx	**not** *HideQuals*
Show Stereotypes	PDATA	**not** *HideStereo*
Show Property String	StyleEx	*AdvancedFeatureProps*
Show Operation Return Type	StyleEx	*ShowOpRetType*
Suppress Brackets for ...	StyleEx	*SuppressBrackets*
Always Show Limked Features	StyleEx	*OverrideLinkedF*
Public	AttPub	
Protected	AttPro	
Private	AttPri	
Package	StyleEx	*AttPkg*
Property Methods	PDATA	**not** *HideProps*
Show Attribute Detail	StyleEx	*VisibleAttributeDetail*
		Name and Type=0
		Name Only=1
Show Parameter Detail	PDATA	*OpParams*
		None=0
		Type Only=1
		Full Details=2
		Name Only=3

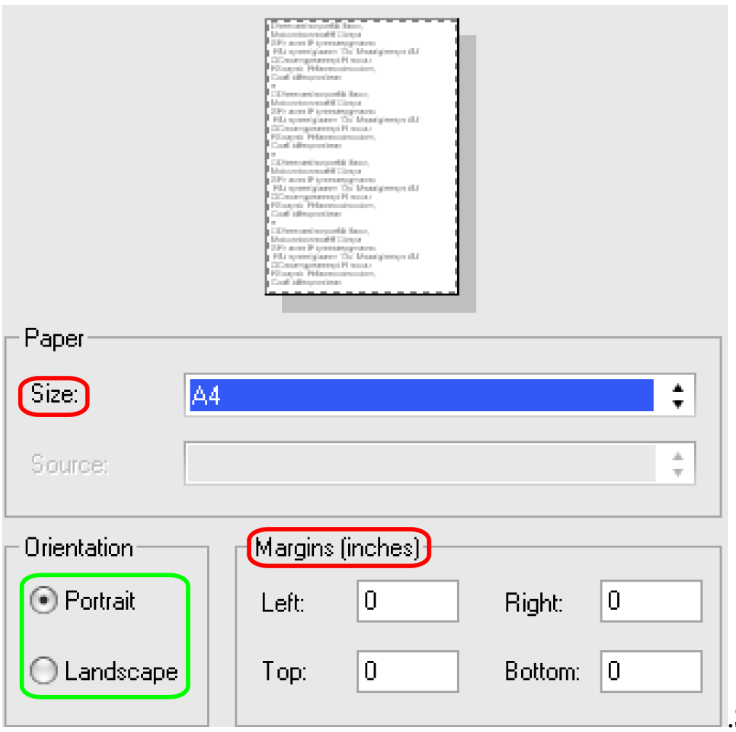
13.2.6 Connectors Window



References: From t_diagram

Label	Column	CSV Tag
Show Relatiosnships	PDATA	**not** *HideRel*
Show Collaboration Numbers	PDATA	**not** *SuppCN*
Show Non-Navigable Ends	StyleEx	*TExplicitNavigability*
Show Connector Property String	StyleEx	**not** *AdvancedConnectorProps*
Suppress All Connector Labels	StyleEx	*SuppConnectorLabels*
Show Stereotype Labels	StyleEx	**not** *HideConnStereotype*
Connector Notation	StyleEx	*TConnectorNotation*

13.2.7 Page Setup Window



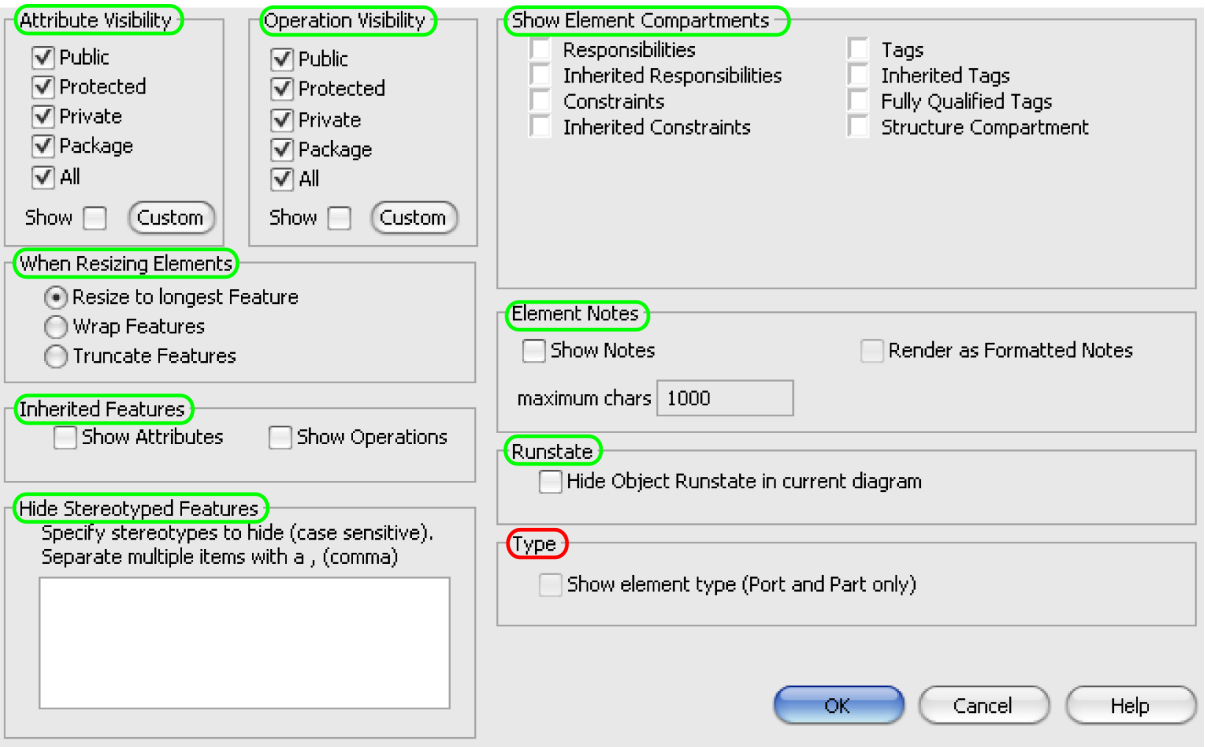
.StyleEx

References:

Label	Column
`Orientation`	t_diagram.Orientation

13.3 Diagram Element Context

13.3.1 Feature Visibility



References: From t_diagramobjects.ObjectStyle

Attribute Visibility	CSV Tag
`Public`	*AttPub*
`Protected`	*AttPro*
`Private`	*AttPri*
`Package`	*AttPkg*

The above are present in the format e.g. *AttPub=0;* if the `Public` is unchecked. When the checkmark is set the CSV tag does simply not appear in the list.

For the `Custom` button see next chapter.

Operation Visibility	CSV Tag
`Public`	*OpPub*
`Protected`	*AOpro*
`Private`	*OpPri*
`Package`	*OpPkg*

The `Show` checkmark is simply a GUI representation for the above four checkmarks.

For the `Custom` button see next chapter.

Show Element Compartment	CSV Tag
`Responsibilities`	*Responsibility*
`Inherited Responsibilities`	*ResInh*
`Constraints`	*Constraint*
`Inherited Constraints`	*ConInh*
`Tags`	*Tag*
`Inherited Tgs`	*TagInh*
`Fully Qualified Tags`	*FQ*
`Structure Compartment`	*SC*

The above are present in the format e.g. *Responsibility=1;* if the `Responsibilities` is checked. When the checkmark is not set the CSV tag does simply not appear in the list. This applies also for the following CSV tags.

When Resizing Elements	CSV Tag
`Resize to longest Feature`	*RzO=1* (default if missing)
`Warp Features`	*RzO=2*
`Truncate Features`	*RzO=3*

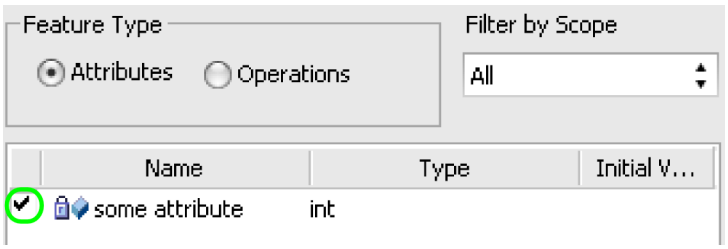
Inherited Features	CSV Tag
`Show Attributes`	*AttInh*
`Show Operations`	*OpInh*

Element Notes	CSV Tag
`Show Notes`	*Notes=*`<maximum chars>`
`Render as ...`	*Formatted*

Runstate	CSV Tag
`Hide...`	*Runstate*

Hide Stereottyped Features	CSV Tag
Input field	*HideSType*

13.3.2 Features & Properties/Feature and Compartment Visibility.../Custom

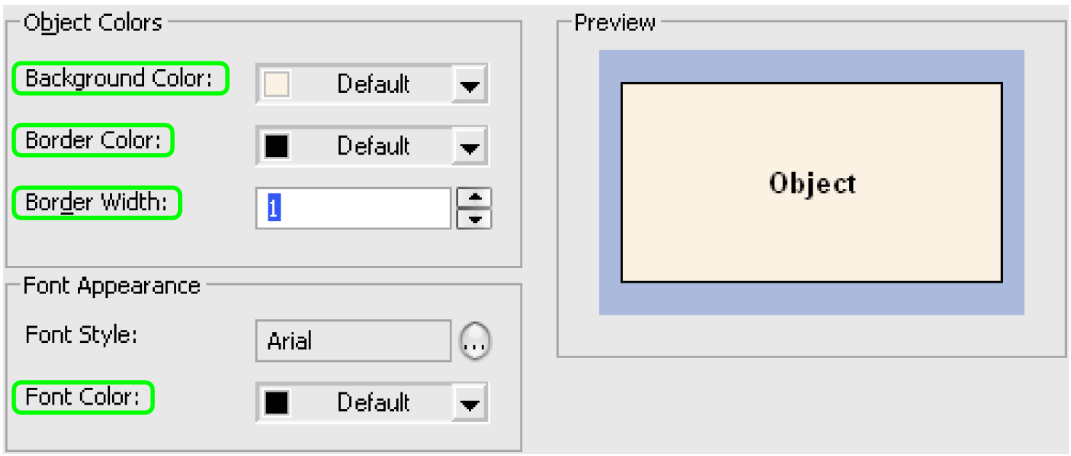


References: From t_diagram.StyleEx

For any attribute or operation you suppress via this dialog EA creates a CSV tag named `SPL`. The value of the tag is a colon (":") separated list of GUID-constructors. Those are build in the format `S_<obj_sguid>=<feat_sguid>` where `<obj_sguid>`are the first 6 chars of the according object GUID and `<feat_guid>` the first 6 GUID chars of the according attribute or operation. E.g. such an entry looks like `SPL=S_566CE9=0F67F5:;` for a single suppressed feature.

If you accidentally have two objects starting with the same chars in the GUID and features which are also not unique in the first 6 chars you will notice that the second object also suppresses one (or more) features (if more features are not unique within the first 6 chars). Well, it's quite unlikely so EA hazards the consequences.

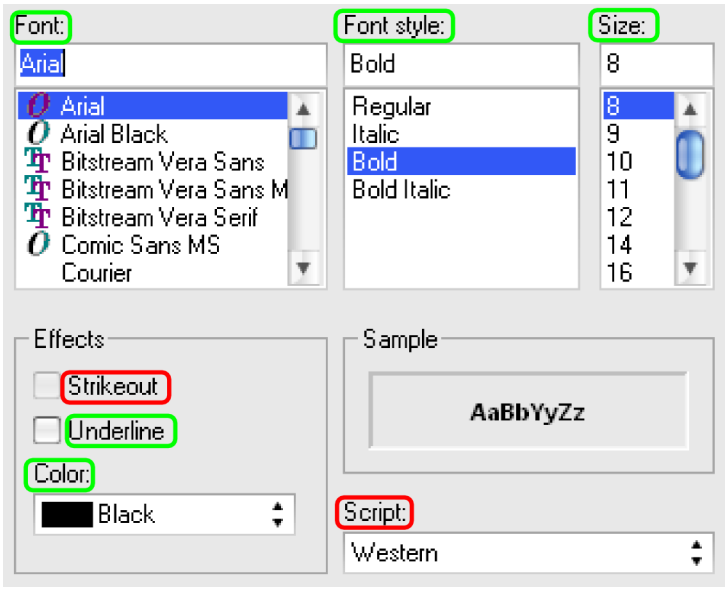
13.3.3 Default Appearance Window



References:

Label	Column
`Background Color`	t_object.Backcolor
`Border Color`	t_object.Bordercolor
`Boder Width`	t_object.BoderWidth
`Font Color`	t_object.Fontcolor

13.3.4 Default Appearance/Font Window

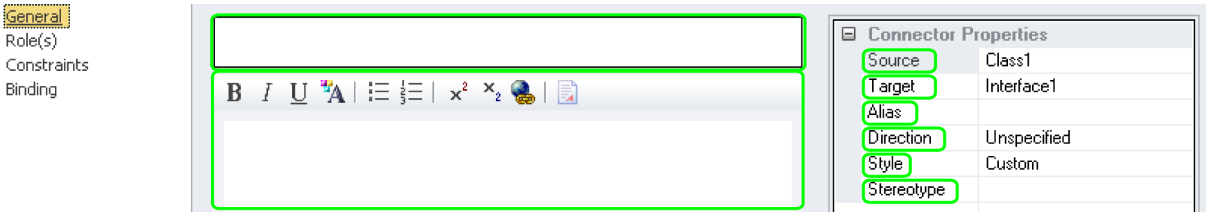


References: t_object.StyleEx

See StyleEx property attributes for details.

13.4 Connector

13.4.1 General Properties Window



References:

Label	Column
Source	t_object.Name of t_connector.Start_Object_ID
Target	t_object.Name of t_connector.End_Object_ID
Name	t_connector.Name
Alias	t_connector.StyleEx
Direction	t_connector.Direction
Stereotype	t_connector.Stereotype
Notes	t_connector.Notes
Style	t_connector.LineStyle

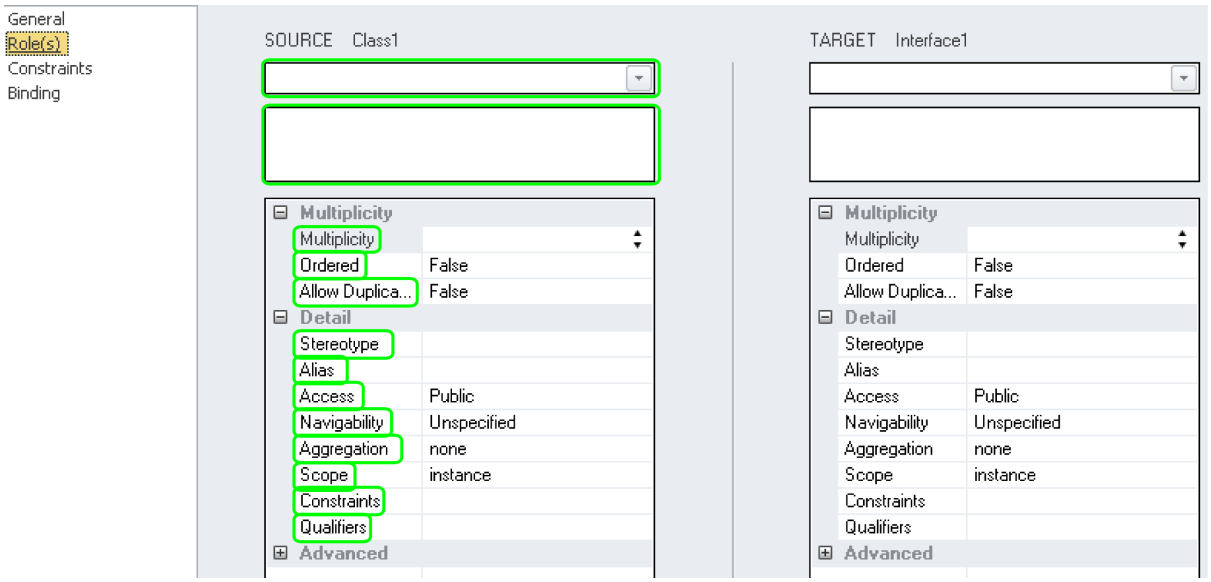
13.4.1.1 Advanced Properties

The `kind` property in the `General/Advanced` window will be mapped to t_xref.Description in a `@PROP` entry where t_xref.Client holds t_connector.ea_guid.

@PROP Key	Value
`@NAME`	*kind*
`@TYPE`	*TransitionKind*
`@VALU`	property value
`@PRMT`	always empty

13.4.2 Source/Target Properties Window

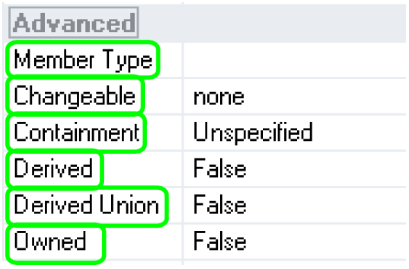
The Source Role windows looks identically to that for the Target Role. Thus all `Source*` references also apply to `Target*` resp. `Dest` in the tables pendants.



References:

Label	Column
`SOURCE`	t_connector.SourceRole
Notes	t_connector.SourceRoleNote
`Multiplicity`	t_connector.SourceCard
`Ordered`	t_connector.SourceIsOrdered
`Allow Duplicates`	t_connector.SourceStyle *AllowDuplicates*
`Stereotype`	t_connector.SourceStereotype
`Alias`	t_connector.SourceStyle *alias*
`Access`	t_connector.SourceAccess
`Navigability`	t_connector.SourceContainment
`Aggregation`	t_connector.SourceIsAggregate
`Scope`	t_connector.SourceTS
`Constraint(s)`	t_connector.SourceConstraint
`Qualifier(s)`	t_connector.SourceQualifier

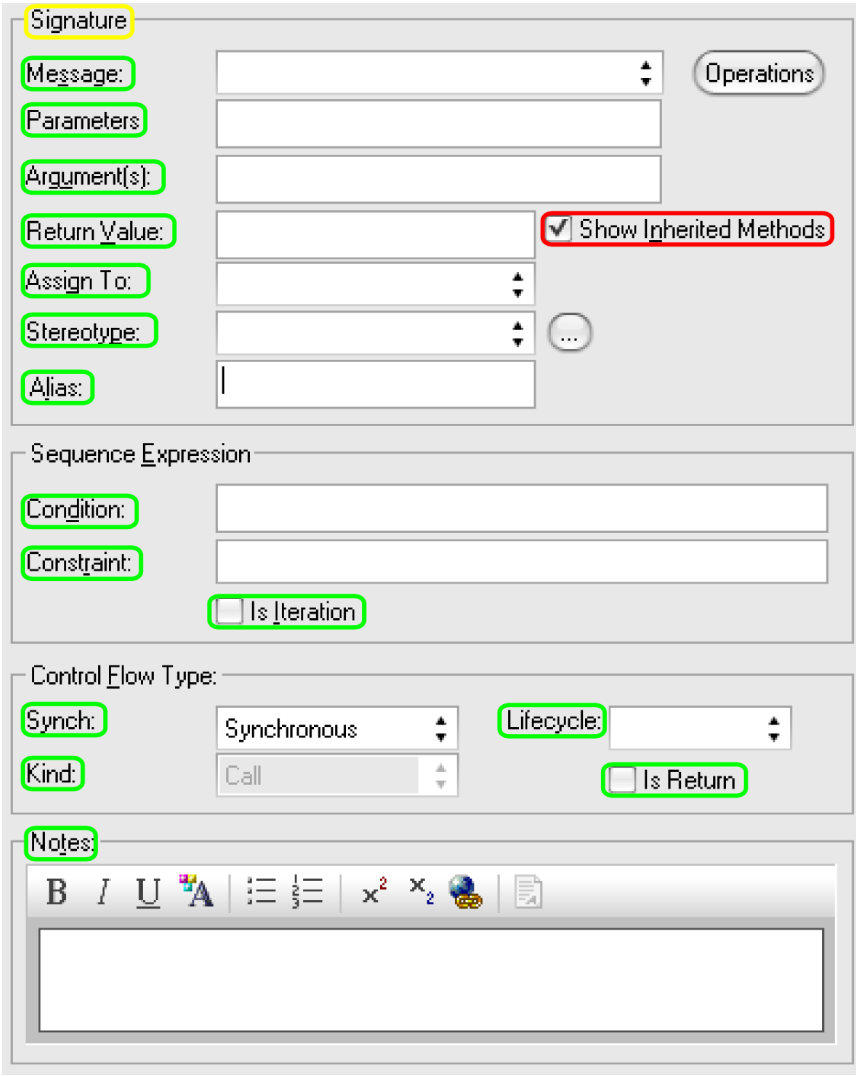
13.4.2.1 Advanced



Label	Column
`Member Type`	t_connector.SourceElement
`Changeable`	t_connector.SourceChangeable
`Containment`	t_connector.SourceContainment
`Derived`	t_connector.SourceStyle *Derived*
`Derived Union`	t_connector.SourceStyle *Union*
`Derived`	t_connector.SourceStyle *Owned*

13.4.3 Message Properties Window

Some of the properties map directly to properties in the t_connector table. But some are encoded in CSV list in the `PDATA2` and `StyleEx` columns.



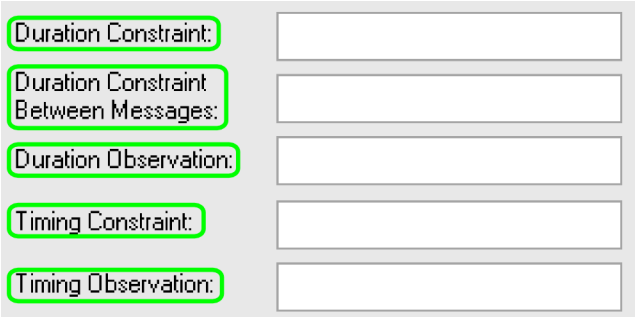
References:

Label	Column
`Message`	t_connector.Name Drop down from methods of classifier
`Parameters`	t_connector.PDATA2 *paramsDlg*
`Arguments(s)`	t_connector.StyleEx *paramvalues*
`Return Value`	t_connector.PDATA2 *retval*
`Assign To`	t_connector.PDATA2 *retatt* Drop down from attributes of classifier
`Stereotype`	t_connector.Stereotype
`Alias`	t_connector.StyleEx *alias*
`Condition`	t_connector.Btm_Mid_Label
`Constraint`	t_connector.SourceConstraint
`Is Iteration`	t_connector.SourceCard either *Iteration* or blank
`Synch`	t_connector.PDATA1 (as in the drop down)
`Lifecycle`	t_connector.SubType (as in the drop down) <*none*> is the same as an empty value
`Kind`	t_connector.PDATA3 (as in the drop down)
`Is Return`	t_connector.PDATA4 *1* if return, else *0*

Label	Column
Notes	t_connector.Notes

13.4.4 Timing Properties Window

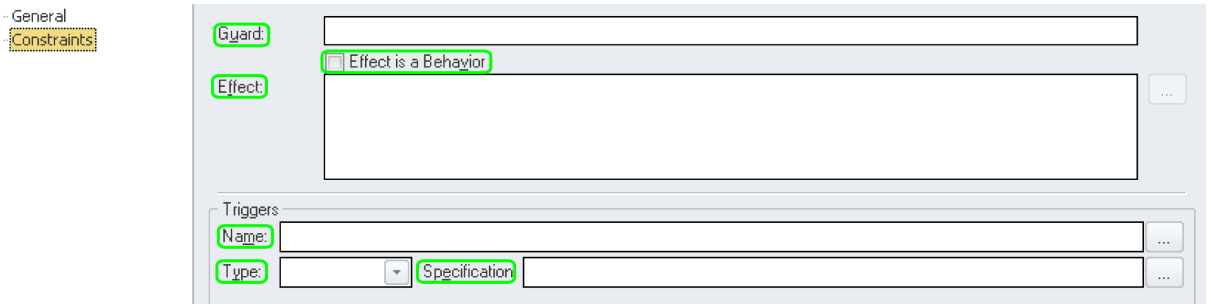
The values in this window will be mapped to keys in the CSV list in t_connector.StyleEx according to the table below.



References:

Key	Value
Duration Constraint	SEQDC
... Between ...	DCBM
Duration Observation	SEQDO
Timing Constraint	SEQTC
Timing Observation	SEQTO

13.4.5 Transition/Constraints Properties Window



References:

Label	Column
Guard	t_connector.PDATA2
Effect	t_connector.PDATA2
Effect is a Behavior	Entry in t_xref; see below

13.4.5.1 Triggers

The names of the triggers for the transition are stored in t_connector.PDATA1 as comma-blank separated list. Additionally a `Trigger` element is created in t_object as well as a number of records in t_xref. The `Trigger` element basically receives the name of the trigger and its t_object.ea_guid is used as reference in the t_xref records.

13.4.5.1.1 Plain trigger

A trigger with `Name` and `Specification` results in three t_xref records.

Record 1

`Name`	*CustomProperties*
`Type`	*element property*
`Description`	`@PROP` list below
`Client`	t_object.ea_guid of the `Trigger` element

@PROP Key	Value
`@NAME`	*kind*
`@TYPE`	*TriggerKind*
`@VALU`	empty
`@PRMT`	empty

Record 2

`Name`	*MOFProps*
`Type`	*element property*
`Behavior`	*event*
`Description`	*RefGUID=<some guid>;RefName=spec;*
`Client`	t_object.ea_guid of the `Trigger` element

Record 3

`Name`	*MOFProps*
`Type`	*connector property*
`Behavior`	*trigger*
`Description`	t_object.ea_guid of the `Trigger` element
`Client`	t_connector.ea_guid of the `Transition`

13.4.5.1.2 Trigger based on class operation

When you connect a trigger with a class operation via the `Type` dropdown the above records have following modification:

Record 1

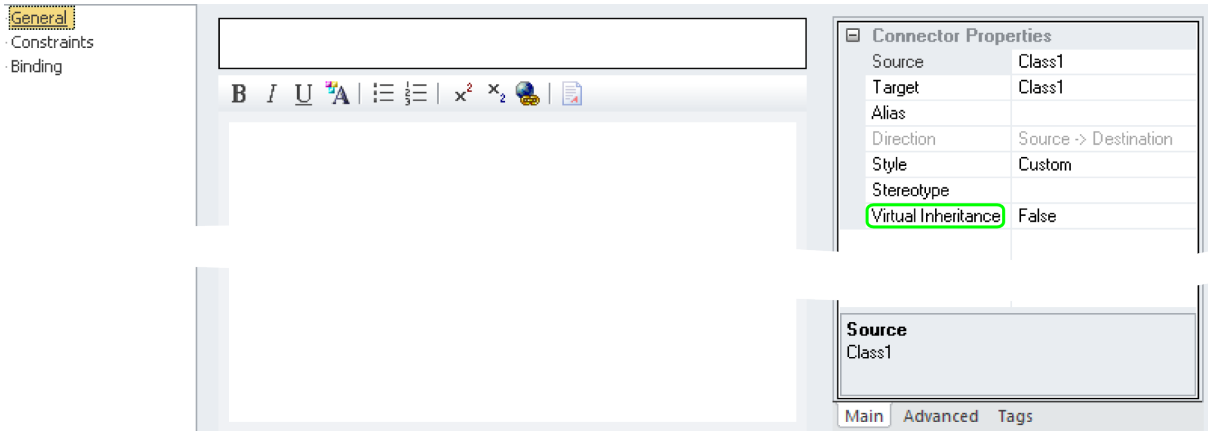
`VALU` contains the `Type` value (*Call, Change, Signal* or *Time*)

Record 2

<some guid> is the t_operation.ea_guid of the referenced class operation.

RefName defines a string composed as *<class>.<operation>*

13.4.6 Generalization Properties Window

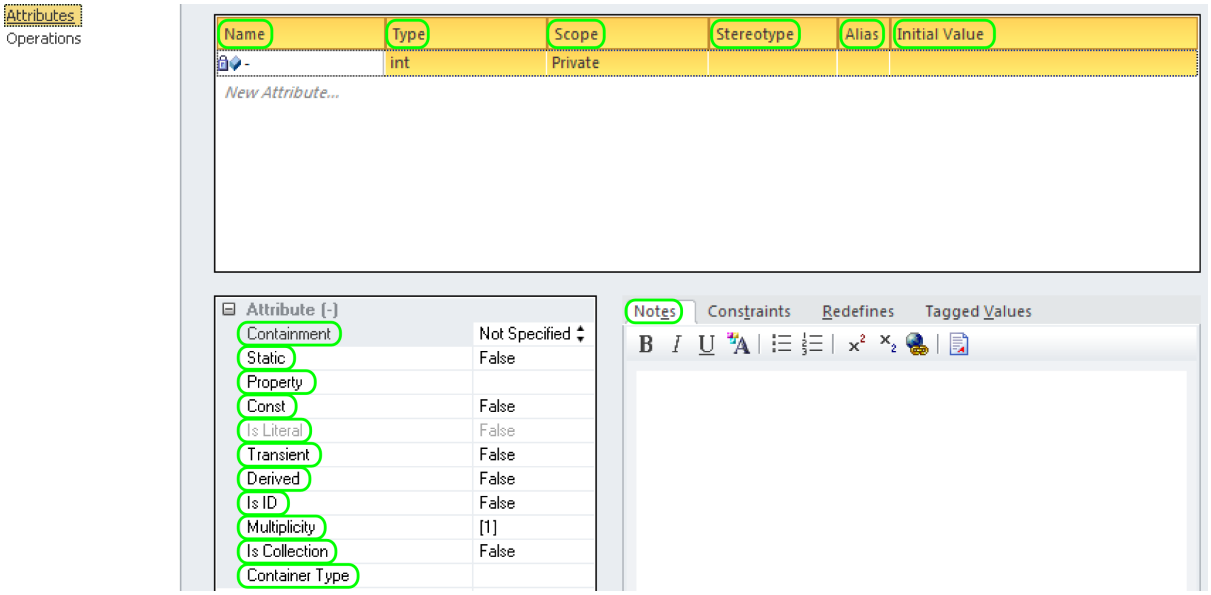


References:

Label	Column
Virtual Inheritance	t_connector.VirtualInheritance

13.5 Attributes

13.5.1 General Properties Window



References:

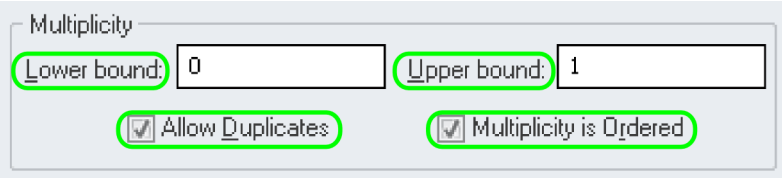
Label	Column
Name	t_attribute.Name
Type	t_attribute.Type
Scope	t_attribute.Scope
Stereotype	t_attribute.Stereotype
Alias	t_attribute.Style
Initial Value	t_attribute.Default
Containment	t_attribute.Containment
Static	t_attribute.IsStatic
Property	t_attribute.GenOption
Const	t_attribute.Const
Is Literal	t_attribute.StyleEx *IsLiteral*
Transient	t_attribute.StyleEx *volatile*
Derived	t_attribute.Derived
Is ID	t_xref entry see below
Multiplicity	see below
Is Collection	t_attribute.IsCollection
Container Type	t_attribute.Container
Notes	t_attribute.Notes

Is ID

Column	Value
Name	*CustomProperties*
Type	*attribute property*
Description	a CSV list starting with *@PROP* (see below)
Client	t_attribute.ea_guid of the attribute

@PROP Key	Value
@NAME	*isID*
@TYPE	*Boolean*
@VALU	property value
@PRMT	empty

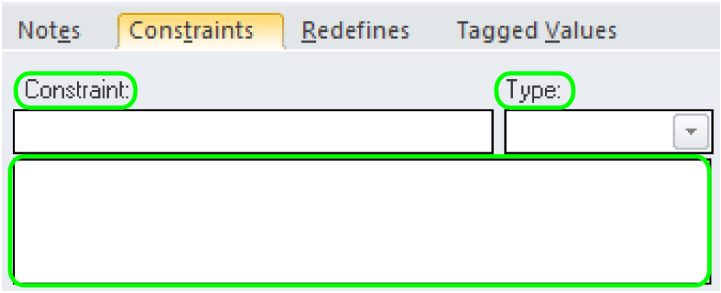
13.5.1.1 Multiplicities



References:

Label	Column
Lower bound	t_attribute.LowerBound
Upper bound	t_attribute.UpperBound
Allow Duplicates	t_attribute.AllowDuplicates
Multiplicit is Ordered	t_attribute.IsOrdered

13.5.1.2 Constraints



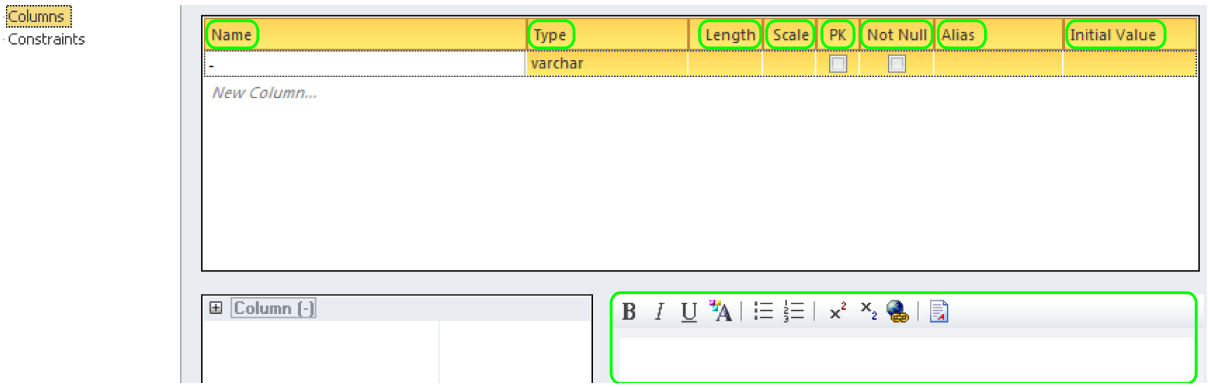
References:

Label	Column
`Constraint`	t_attributeconstraints.Constraint
`Type`	t_attributeconstraints.Type
Notes	t_attributeconstraints.Notes

13.5.2 Column General Properties Window

This special attribute property dialog appears only for `<<column>>` stereotyped attributes in a `<<table>>` class.

Note that for the table class and the columns EA will create `EAUML` MDG stereotype entries in t_xref.



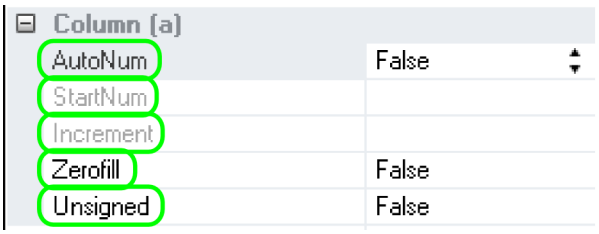
References:

Label	Column
Name	t_attributetag.Name
Type	t_attribute.Type
Length	t_attribute.Length
Scale	t_attribute.Scale
PK	t_attribute.IsOrdered
Not Null	t_attribute.AllowDuplicates
Alias	t_attribute.Style
Initial Value	t_attribute.Default
Notes	t_attribute.Notes

Length maps to t_attribute.Precision in case Type is *Decimal*. Yippie!

13.5.2.1 Advanced properties for some properties

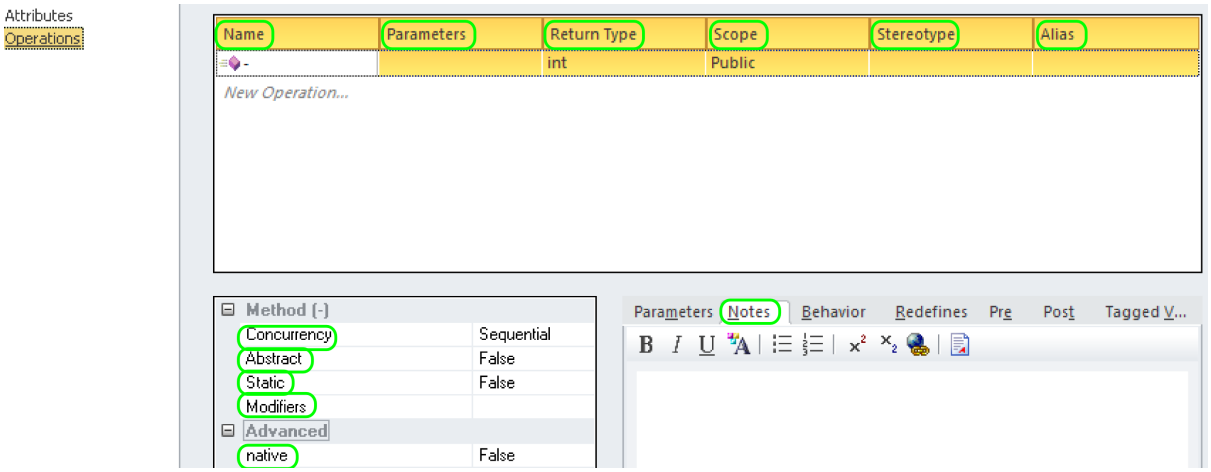
For Float and some other datatypes the properties pane contains a couple of different values:



These will be mapped to t_attributetag with the column Property having the name of the property and VALUE holding the value. Additionaly (for *Autonum*) an additional row is created where Property equals *property* and VALUE repeats *AutoNum=1;StartNum=1;Increment=1;*.

13.6 Operations

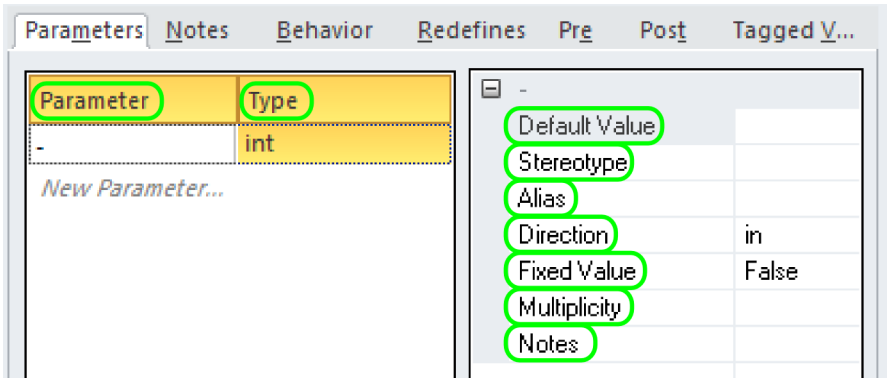
13.6.1 General Properties Window



References:

Label	Column
Name	t_operation.Name
Parameters	Edited data from t_operationparams
Return Type	t_operation.Type
Scope	t_operation.Scope
Stereotype	t_operation.Stereotype
Alias	t_operation.Style
Concurrency	t_operation.Concurrency
Abstract	t_operation.Abstract
Static	t_operation.IsStatic
Modifiers	t_operation.Const *0/1*
	t_operation.ReturnArray *0/1*
	t_operation.Synchronized *0/1*
	t_operation.Pure *True/False*
	t_operation.IsQuery *True/False*
native	t_operationtag with *native* as Property
Notes	t_operation.Notes

13.6.1.1 Parameters Properties Window



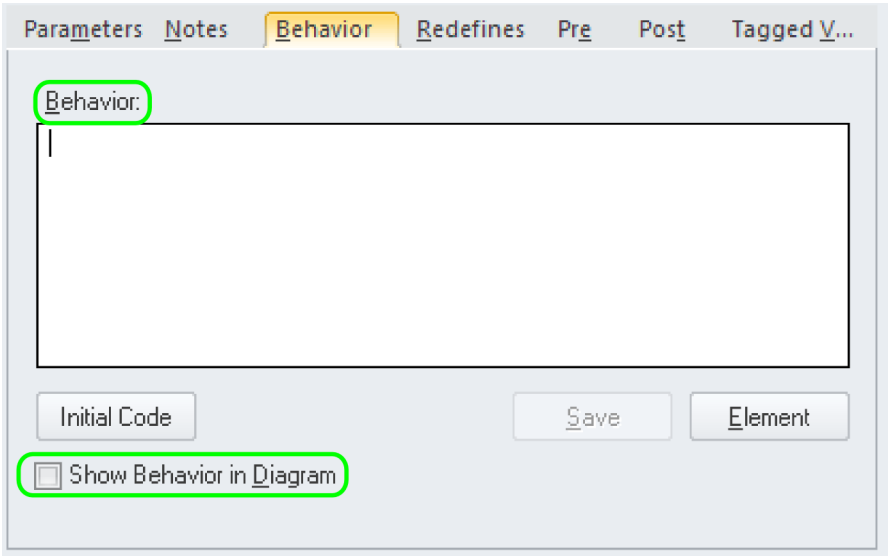
References:

Label	Column
Parameter	t_operationparams.Name
Type	t_operationparams.Type
Default Value	t_operationparams.Default
Stereotype	see t_xref table reference below
Alias	t_operationparams.StyleEx *alias*
Direction	t_operationparams.Kind
Fixed Value	t_operationparams.Const *True/False*
Multiplicity	see t_xref table reference below
Notes	t_operationparams.Notes

Stereotype Column	Value
Name	*Stereotypes*
Type	*parameter property*
Description	a CSV list starting with *@STEREO*
Client	t_operationparams.ea_guid of the operation parameter

Multiplicity Column	Value
Name	*CustomProperties*
Type	*parameter property*
Description	a CSV list starting with *@PROP*
Client	t_operationparams.ea_guid of the operation parameter

13.6.1.2 Behavior Properties Window



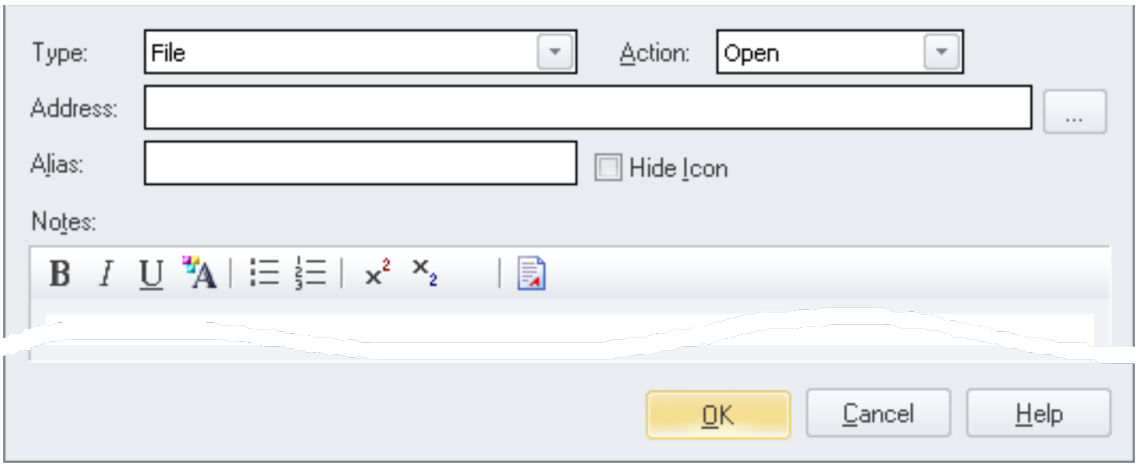
References:

Label	Column
Behavior	t_operation.Behavior
Show Behavior in Diagram	t_operation.StyleEx
Initial Code	t_operation.Code

13.7 Miscellaneous

13.7.1 Hyperlinks

Meanwhile the hyperlink allow a lot more variations than former EA versions did. So it can not only link to files or URLs, but also to attributes, diagrams, etc. and even can execute EA commands. Basically any hyperlink is (besides its visible part in the diagram) stored in t_object with `Object_Type="Text"` and `NType=19`.



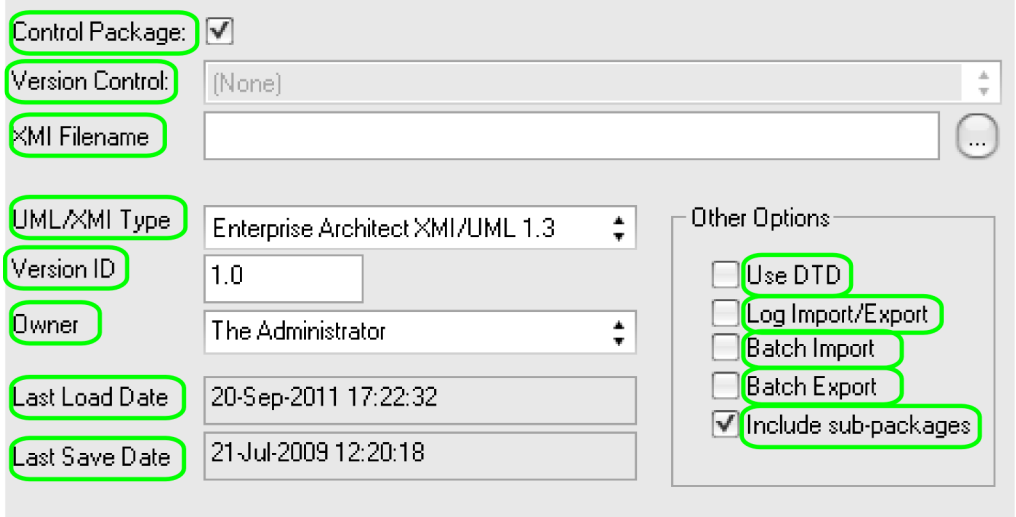
I'm not gonna show screen shots for each variant of the hyperlink.

Type (drop down)	Name	Remark
Attribute	`$feature://`GUID (of attribute)	or operation (see below)
Diagram	`$diagram://`GUID (of diagram)	`NType` contains 0
Diagram Image	`$diagramming://`GUID (of diagram)	
EA Command	`$uicmd=`<cmd>`;param1=`<first>`;`	`param` only where appropriate
Element	`$element://`GUID (of element)	
File	<name of the file>	`Style` contains `LinkOpen=`<action>`;` where <action> is either `open` or `edit`
Help	`$help://`<contents of Address>	
Image Manager	`$imageman://`<details>	you need to find out details yourself
Learning Center	`$learning://`<some guid>	
Matrix	`$matrix://`<name of matrix>	
Operation	`$feature://`GUID (of operation)	or attribute (see above)
Package	`$package://`GUID (of package)	
Search	`$search://Name=`<name of search>`;Term=`<search term>	
Web Site	`$inet://`<contents of Address>	

Attribute/Operation can contain either GUID

The `Alias` in case of one of the Diagram links is taken from the `Notes` input field.

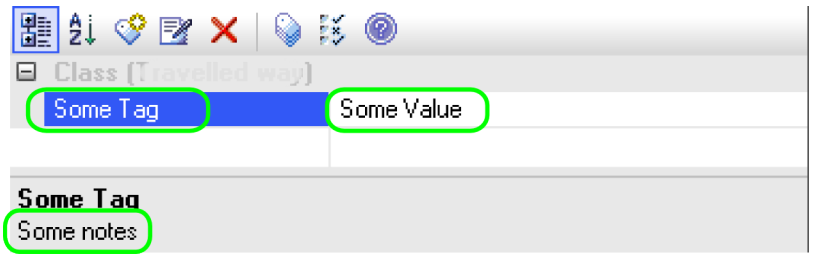
13.7.2 Package Control Properties Window



References:

Label	Column
`Control Package`	t_package.IsControlled
`Version Control`	The `id` column from paths.txt
`XMI Filename`	t_package.XMLPath
`UML/XMI Type`	t_package.UMLVersion
`Version ID`	Tagged value *version* written to the XMI
`Owner`	t_package.PkgOwner,
`Last Load Date`	t_package.LastLoadDate
`Last Save Date`	t_package.LastSaveDate
`Use DTD`	t_object.UseDTD
`Batch Export`	t_object.BatchSave
`Batch Import`	t_object.BatchLoad
`Log Import/Export`	t_object.LogXML
`Include sub-packages`	t_package.PackageFlags value *Recurse=<val>;* where *<val>* is either *1* (include) or *0* (dont include)

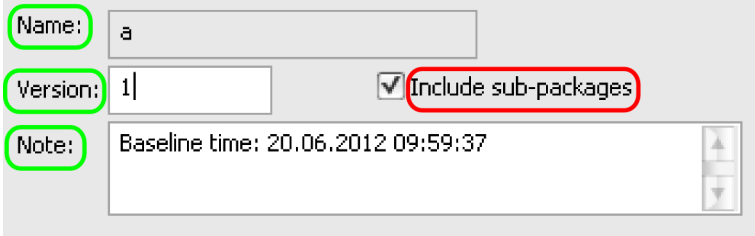
13.7.3 Tagged Values Docked Window



References:

Label	Column
Name of the tag	t_objectproperties.Property
Value of the tag	t_objectproperties.Value
Notes of the tag	t_objectproperties.Notes

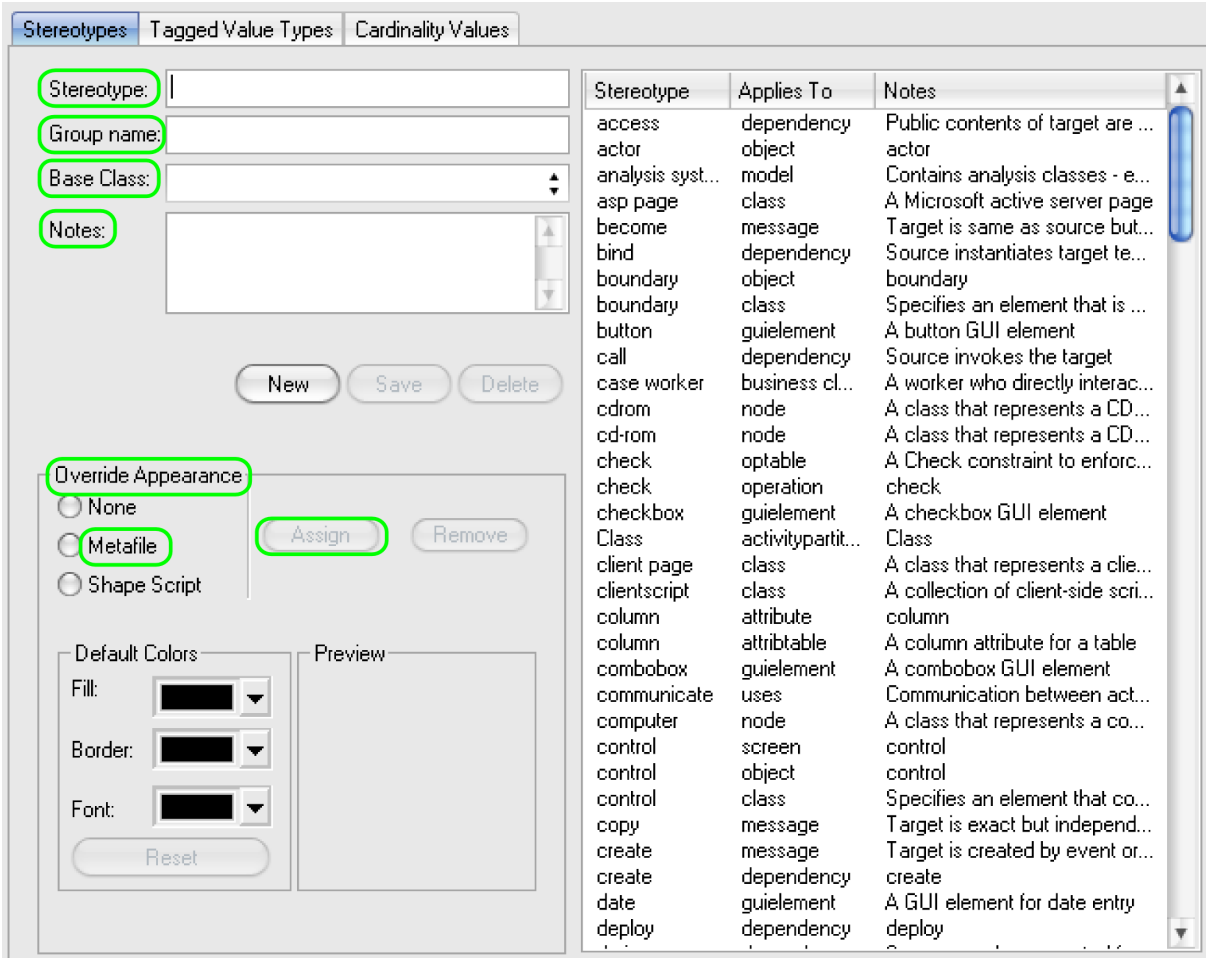
13.7.4 Baseline Creation Window



References:

Label	Column
Name	t_document.Docname
Version	t_document.Version
Note	t_document.Note

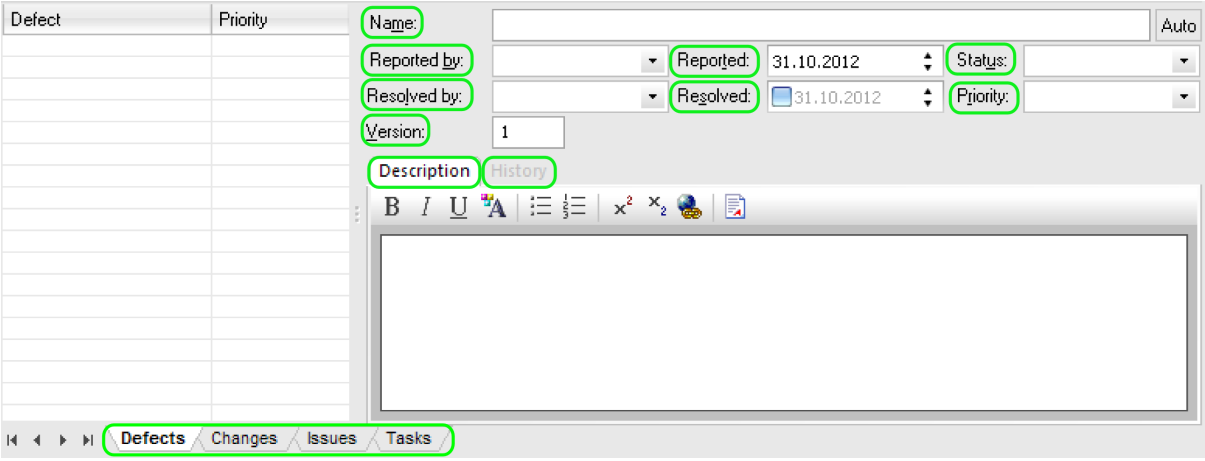
13.7.5 Stereotypes Definition Window



References:

Label	Column
`Stereotype`	t_stereotype.Name
`Base Class`	t_stereotype.AppliesTo
`Notes`	t_stereotype.Description
`Metafile`	t_stereotype.MFEnabled
`via Assign`	t_stereotype.MFPath
`Override Appearance`	t_stereotype.Style

13.7.6 Maintenance Window



References:

Label	Column
`Name`	t_objectproblems.Problem
`Defects/.../Tasks`	t_objectproblems.ProblemType
`Reported`	t_objectproblems.DateReported
`Status`	t_objectproblems.Status
`Description`	t_objectproblems.ProblemNotes
`ReportedBy`	t_objectproblems.ReportedBy
`ResolvedBy`	t_objectproblems.ResolvedBy
`Resolved`	t_objectproblems.DateResolved
`Version`	t_objectproblems.Version
`History`	t_objectproblems.ResolverNotes
`Priority`	t_objectproblems.Priority

14. Query Caveats

Once you start using direct SQL you need to know:

> **All SQL are equal**
>
> But some SQL are more equal than others

And that's unfortunately true. A SQL for MS Access (EAP) is different to that of its 'big brother' MS SQL Server.

I can not go into detail since there are so many different SQL flavors out there and I'm not a DBA. But a few should be lined out below. Also note that entering SQL in the search builder is treated specially by EA. Some #-tags are interpreted due to exactly that reason. So EA supports you a bit when dealing with different SQL dialects.

- **Wild Cards**: EAP uses '*', most other SQL derivates use '%'. EA search translates `#WC#` into the appropriate wild card.
- **Dates**: EAP uses a `#mm/dd/yyyy#` format. Here the #-tags are not interpreted by EA. In most other SQL you can use a string `"yyyy-mm-dd"` to delimit a date search.
- **Syntax in General**: Ouch. The best advice is to stick to the most simple syntax possible. Of course you can do very fancy things with an ORACLE database which you even can't dream of in EAP. But then you're fixed with that database. For minor syntax differences the EA Search Builder offers the `#DB=<db>` tags.

When you use

```
1         res = Respository.SQLquery (sql);
```

you need to respect the database type in constructing the SQL. Rather than placing `#DB=` parentheses you need to individually construct the according SQL. Usually you need to distinguish between EAP (for user local use) and the SQL server of choice.

14.1 Debugging SQL

As already explained in the introduction you can use the embedded SQL editor. This is convenient in respect to the auto-completion. The drawback here is that you have only a single edit window. You might consider to additionally use a tool like `Toad`[1] or `Navicat` which have a load of features to support SQL development. Of course - if you have access - you can use the DB maintenance tools that comes with the RDBMS you deploy. But then you would need two tools (MS Access

[1] Just google for "toad sql" or "navicat" and you will find the right source.

for EAP and that for your RDBMS). So using `Toad` is preferable as it supports different RDBMS including MS Access with a single user interface.

Whenever you deploy your own SQL on EA and it comes to errors EA just passes the error message from the underlying system. This will usually show a popup window which eventually is closed too soon. In that case you can look into `%APPDATA%\Sparx Systems\EA\DBError.txt`. Here you will find a bit more information. A bit of intuition is needed to decode this error, though.

15. SQL Search Builder

Once you worked out the previous chapters you are ready to provide your EA users with some fancy searches (available as download[1]). You find the search builder when navigating from `Ctrl-F/Builder` and pressing the left `New Search` icon.

Now you have to name the search (e.g. `Diagram by name`) and select `SQL Editor`. Here you can enter any query that is supported by the database(s) you utilize.

15.1 Search Results

For most queries you are interested in elements, packages, diagrams, connectors, operations and attributes. Here the search builder interprets two result columns:

- **CLASSGUID** which must contain the respective ea_guid of the t_object, t_package, t_-diagram, t_connector, t_operation or t_attribute tables. This column will not display in the results. Instead it is interpreted by EA in order to target the right element for a double click.
- **CLASSTYPE** depends on the result table. For t_object you can simply specify `Object_Type AS CLASSTYPE` and for t_diagram `Diagram_Type AS CLASSTYPE`. Like the previous column it will not appear textual in the results. Instead it is used to show an icon corresponding to its value. Other possibilities are
- `'Package' AS CLASSTYPE` for t_package,
- `'Operation' AS CLASSTYPE` for t_operation,
- `'Attribute' AS CLASSTYPE` for t_attribut.

If you are after the right diagram icon you need to add

- `'t_diagram' as CLASSTABLE`

as first result column (yeah, why be consistent?).

If you have specified the above two columns as part of the result set EA will render an according icon in front of each result line. Elements, Operations and Attributes can directly be opened via dbl-click in the result set. Diagrams and Packages however do not open directly. Instead you have to locate them in the project browser via `Alt-G` and open from there.

Another nice feature is the ability to copy/paste the result to a spreadsheet. Simply select the result lines and copy them into the paste buffer. You can paste them e.g. into OpenOffice Calc by selecting a semi-colon as separator. Here's a drop of bitterness, though: columns like

[1] https://liquit.biz/query.zip

t_diagram.StyleEx do contain semi-colons themselves. And EA does not escape these (that is putting quotes around the according column and doubling quotes inside). So currently you should not include these columns if you intend to use them for copy/paste.

Another way to export the results is the `Export to CSV` option hidden in the `Options` button (see here top right). You first have to select any rows you want to export or you will never see the file selection dialog for saving the `csv`. The output format is just the same as that you will get in the paste buffer.

So guess why EA never includes the `Notes` in the export. If you need this column then specify something like

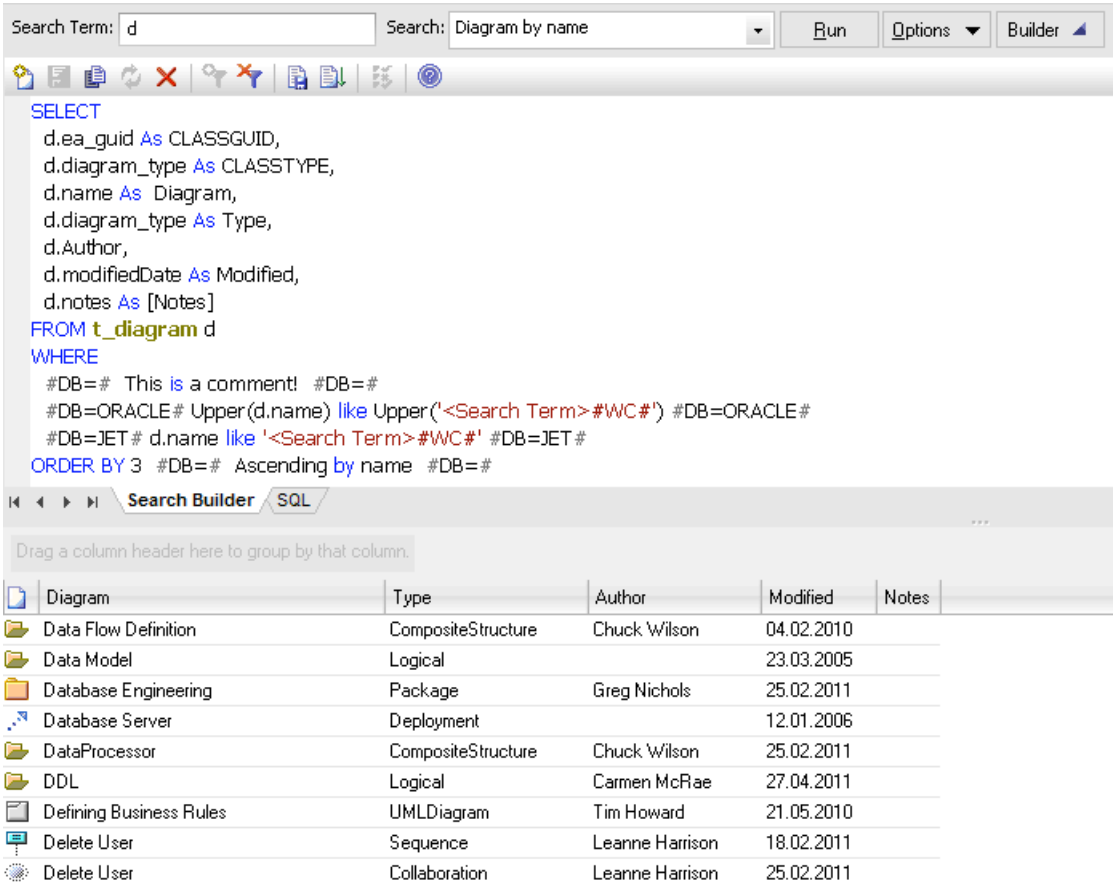
```
t_object.Note as note_
```

This again will put you in the bad situation that line breaks and field separators (semi-colon) are not escaped and the export is useless in most cases. You could write some tricky SQL which places double quotes around the notes and doubles all the double quotes inside, but I'm not sure if you really want to do that. Most likely you are better served with a little script that performs the query and writes the XML result to disk that comes directly from the `Respository.SQLSearch()`[2].

15.2 Search Tagging

Anticipating you have issued the Diagram by name search below you should see something like the following:

[2]Don't ask me why this is not offered as option. It's obviously implemented and probably a candidate for a feature request.



Search Window

While the SQL query window passes the SQL string directly to the underlying database the EA search processes the SQL string before executing it.

In this sample EA will replace the substring `<Search Term>` with whatever has been entered in the Search Term. The appropriate wild card is appended to the string so the comparison is matched against any name beginning with what had been entered in the Search Term. Since Oracle is case sensitive in its search and EAP is not, the comparison for the ORACLE DB is made to compare upper case strings. Therefore the `#DB=<db>#` parentheses are used.

The replacement tags are as follows[3]:

- `<Search Term>` - quote-escaped contents of the Search Term input field (see example).
- `#WC#` - the wild card for string comparison for the active database (see example).
- `#Author#` - value from `Tool/Options/General/Author`
- `#UserName#` - The Windows login or in case of enabled EA security the user from the EA login dialog.
- `#DB=<db>#` - where `<db>` is one of MYSQL, JET, ORACLE, SQLSVR, ASA, OPENEDGE or POSTGRES. The tags used in pairs with identical `<db>` specifier allow definition of database specific queries (see example). A bit strange but useful is the use of `#DB=#` parentheses. EA obviously does only expand those parentheses were the used database matches the specified one. And if you omit the `<db>` specification EA will always ignore the text inside

[3]Note that all tags are case sensitive!

the parentheses. So you can use the to comment your queries. The EA help specifies a list of `<db>` names. Currently it does not warn if you choose an invalid/empty string. You could also choose a `<db>` name which you do not use in your environment. Or simply use `#DB=ORACLE# /* any comment */ #DB=ORACLE#` as Oracle can process comments. *Caveat:* It seems that EA reacts strange to such comments appearing at the very beginning of a SQL. That means it does not issue any complaint but silently ignores anything. No error message, no result!

- `#Package#` - holds the t_package.Package_ID of the currently selected package.
- `#Branch#` - yields a collection of t_package.Package_ID of the current package and all its sub-packages (see example).

15.3 Some Sample Queries

As a warning please note that when pasting the queries EA often gets confused and does not execute any line. The result list stays silently empty. To be safe manually type the `SELECT` keyword and just paste the rest of the string. That seems to work always.

15.3.1 Diagram by name

Retrieve all diagrams which name start with a specific (case independent) string.

Only available in the SQL Query builder unless you strip off the `#DB#` tags according to your repository.

```
SELECT
  d.ea_guid As CLASSGUID,
  d.diagram_type As CLASSTYPE,
  d.name As  Diagram,
  d.diagram_type As Type,
  d.Author,
  d.modifiedDate As Modified,
  d.notes As [Notes]
FROM t_diagram d
WHERE
  #DB=#  This is a comment!  #DB=#
  #DB=ORACLE# Upper(d.name) like Upper('<Search Term>#WC#') #DB=ORACLE#
  #DB=JET# d.name like '<Search Term>#WC#' #DB=JET#
ORDER BY 3  #DB=#  Ascending by name  #DB=#
```

15.3.2 Recursive elements in package

List all elements which are contained in a package along with all its sub-packages.

Only available in the SQL Query builder.

```
SELECT
  o.ea_guid AS CLASSGUID, o.Object_type AS CLASSTYPE,
  o.name, o.Object_type AS Type, o.Stereotype, o.Author,
  pkg.name AS [Package Name],
  o.modifiedDate AS Modified, o.note AS [Notes]
FROM
  t_object o, t_package pkg
WHERE
  pkg.Package_id in (#Branch#)
  AND o.Package_ID = pkg.package_id
#DB=# add more types if needed to the following list #DB=#
  AND o.Object_Type NOT IN ('Package', 'Text', 'Merge')
  AND o.name <> ''
  AND o.name like '#WC#<Search Term>#WC#'
ORDER BY 3,4,5
```

15.3.3 GUID

Find and list all elements/diagrams/attributes/operations which match a certain GUID. The result set is always one as GUIDs are unique over all kind of elements. This query is an alternative to what I described in my article Create a "hyperlink" for EA elements[4].

Only available in the SQL Query builder unless you strip off the `#DB#` tags according to your repository.

[4] https://liquit.biz/brain/enterprise.html

```
SELECT
  o.ea_GUID AS CLASSGUID, o.Object_Type As CLASSTYPE,
  o.Name, o.Object_Type, o.stereotype, o.Author,
#DB=ORACLE#
  Cast(o.ModifiedDate As Varchar(20)) As Modified,
  Cast(SubStr(o.note,1,200) As Varchar(200)) As Notes
#DB=ORACLE#
#DB=JET# o.ModifiedDate As Modified, Mid(o.note,1,200) As Notes #DB=JET#
FROM
  t_object o
WHERE
  o.ea_guid = '<Search Term>'

UNION SELECT
  d.ea_GUID, d.Diagram_Type, d.Name,
  d.Diagram_Type, d.stereotype, d.Author,
#DB=ORACLE#
  Cast(d.ModifiedDate As Varchar(20)) As Modified,
  Cast(SubStr(d.notes,1,200) As Varchar(200))
#DB=ORACLE#
#DB=JET# d.ModifiedDate As Modified, Mid(d.notes,1,200) #DB=JET#
FROM
  t_diagram  d
WHERE
  d.ea_guid = '<Search Term>'

UNION SELECT
  a.ea_GUID,  'Attribute' , a.Name, a.type, a.stereotype, ' ', ' ',
  #DB=ORACLE# Cast(SubStr(a.notes,1,200) As Varchar(200)) #DB=ORACLE#
  #DB=JET# Mid(a.notes,1,200) #DB=JET#
FROM
  t_attribute  a
WHERE
  a.ea_guid = '<Search Term>'

UNION SELECT
  op.ea_GUID,  'Operation' , op.Name, 'Operation', op.stereotype, ' ', ' ',
  #DB=ORACLE# Cast(SubStr(op.notes,1,200) As Varchar(200)) #DB=ORACLE#
  #DB=JET# Mid(op.notes,1,200) #DB=JET#
FROM
  t_operation  op
WHERE
  op.ea_guid = '<Search Term>'
```

15.3.4 Replace

If you really need this, here's a way to replace line-breaks with human readable text. You might also use this to replace semi-colons with commas. There's a good chance your Access version will not be able to do that as the `Replace` function is only available from Access 2000 on.

Only available in the SQL Query builder.

```
#DB=SQLSVR#
  Replace(Replace(Cast(t_object.note As varchar(max)), ';',':'),
    CHAR (13) + CHAR(10),'-->') As Notes,
#DB=SQLSVR#
```

15.3.5 Tags

This query finds all tagged values for all elements. The result set is obviously very large so you need to apply some `WHERE` clause to reduce it.

```
SELECT
  o.ea_guid AS CLASSGUID, o.Object_Type As CLASSTYPE,
  o.Name, p.property, p.value
FROM t_object o
INNER JOIN t_objectproperties p
ON o.object_id = p.object_id
```

15.3.6 Methods

This rather complex query will find the usage of a method in all diagrams (like in sequence messages).

Only available in the SQL Query builder unless you strip off the `#DB#` tags according to your repository.

```
SELECT
  o.ea_guid As CLASSGUID, o.Object_type As CLASSTYPE,
  'Behaviour' As Usage, o.name As ElementName,
  o.Object_Type As ElementType, o.stereotype As ElementStereotype,
  '' As Diagram, o.ea_guid
FROM t_operation op,  t_object o
WHERE
  op.EA_GUID = '<Search Term>'        AND
#DB=JET# op.Behaviour = o.EA_GUID #DB=JET#
#DB=ORACLE# Cast(op.Behaviour As Varchar2(38)) = o.EA_GUID #DB=ORACLE#

#DB=ORACLE#
/* --- Find State Operation from Class operation (do,entry,exit) --- */
#DB=ORACLE#
UNION
SELECT
  op.ea_guid, 'Operation', 'Operation Class<--> State', op.name,
  Type, op.stereotype, '', op.ea_guid
FROM  t_operation op
WHERE
#DB=JET# op.Behaviour = '<Search Term>' #DB=JET#
#DB=ORACLE#
  Cast(op.Behaviour As Varchar2(38))  = '<Search Term>'
#DB=ORACLE#

#DB=ORACLE#
/* --- Find Class Operation from State operation (do,entry,exit) --- */
#DB=ORACLE#
UNION
SELECT
  op.ea_guid, 'Operation', 'Operation Class<--> State', op.name,
  op.Type, op.stereotype,'', op.ea_guid
FROM  t_operation opState, t_operation op
WHERE
  opState.ea_guid = '<Search Term>'  AND
#DB=JET# opState.Behaviour = op.ea_guid #DB=JET#
#DB=ORACLE# Cast(op.Behaviour As Varchar2(38)) = op.ea_guid #DB=ORACLE#

#DB=SQLSVR#
/* --- Find Call Action ------------------------------------------- */
#DB=SQLSVR#

UNION
SELECT
  o.ea_guid, o.Object_Type, 'Call Action', o.name,
  o.Object_Type, o.stereotype,'', o.ea_guid
```

```
47  FROM t_operation op,  t_object o
48  WHERE
49    o.Classifier_GUID = '<Search Term>'
50  AND  o.Classifier_GUID = op.ea_GUID
51
52  #DB=SQLSVR#
53  /* --- Find return type of method---------------------------------- */
54  #DB=SQLSVR#
55  UNION
56  SELECT
57    o.ea_guid, o.Object_Type, 'ReturnType', o.name,
58    o.Object_Type, o.stereotype,'', o.ea_guid
59  FROM t_operation op,  t_object o, t_object o1
60  WHERE
61      op.EA_GUID = '<Search Term>' AND
62  #DB=JET# Format(o.Object_ID) = op.Classifier #DB=JET#
63  #DB=ORACLE# o.Object_ID = op.Classifier #DB=ORACLE#
64  #DB=SQLSRV# Usage in Sequence Diagram #DB=SQLSRV#
65  AND op.object_id = o1.object_id
66
67  UNION
68  SELECT
69    c.ea_guid, c.connector_type, 'Sequence', c.name, 'Operation',
70    o.stereotype,d.name, c.ea_guid
71  FROM
72    t_connector c, t_object o, t_operation op,
73    t_diagram d, t_diagramlinks dl
74  WHERE
75    c.end_object_id = o.object_id AND o.object_id = op.object_id AND
76    '<Search Term>' = op.ea_guid AND dl.diagramID = d.diagram_ID AND
77    dl.connectorID = c.connector_id
78
79  UNION
80  SELECT
81    c.ea_guid, c.connector_type, 'Sequence', c.name,
82    'Operation',  o.stereotype, d.name, c.ea_guid
83  FROM
84    t_connector c, t_object o1, t_object o, t_operation op,
85    t_diagram d, t_diagramlinks dl
86  WHERE
87    c.end_object_id = o1.object_id AND o1.object_id = o.object_id AND
88    o.object_id = op.object_id AND '<Search Term>' = op.ea_guid AND
89    dl.diagramID = d.diagram_ID AND dl.connectorID = c.connector_id
90
91  Order By 3,4
```

15.4 Combine Script with Search

Here's an even more advanced way to use searches. That is, if you combine it with a script. I'm definitely not going into scripting details here as it would go beyond the scope of this book. However, here is a small sample. It starts with a SQL to list all elements conveyed by an information flow. As you will see this search requires the Connector_ID of the information flow in question. But this is not directly available to a user. So the only way to retrieve it is via a script. This script can retrieve any context element (here the information flow being selected in a diagram) and pass it to the search. In order to do this you have to create the script and the search once. Now you can select an information flow (conveying some classes) in a diagram and execute the script from the scripts window (see below).

15.4.1 Information Flow Conveyed Query

Only available in the SQL Query builder unless you strip off the `#DB#` tags according to your repository.

```
SELECT DISTINCT o.ea_guid as CLASSGUID, o.Object_Type as CLASSTYPE,
  o.name As Item, o.Object_Type As ItemType,
  o.stereotype As 'ItemStereotype', "Connector" As ConnectorType,
  c.Name, c.Stereotype

FROM t_object o, t_xref xCon, t_xref xFlow, t_connector c, t_connector flow

WHERE
  c.connector_ID = <Search Term> AND c.ea_guid = xCon.Client AND
  flow.ea_guid = xFlow.client AND xCon.Behavior = 'abstraction' AND
  flow.ea_guid IN (
#DB=SQLSVR#
  substring(x.description,0,39),
  substring(xCon.description,39,39),
  substring(xCon.description,78,39),
  substring(xCon.description,117,39),
  substring(xCon.description,156,39),
  substring(xCon.description,195,39),
  substring(xCon.description,234,39),
  substring(xCon.description,273,39),
  substring(xCon.description,312,39),
  substring(xCon.description,351,39)
#DB=SQLSVR#
#DB=Other#
  left(xCon.description,38),
  mid(xCon.description,40,38),
```

```
  mid(xCon.description,79,38),
  mid(xCon.description,118,38),
  mid(xCon.description,157,38),
  mid(xCon.description,196,38),
  mid(xCon.description,235,38),
  mid(xCon.description,274,38),
  mid(xCon.description,313,38),
  mid(xCon.description,352,38)
#DB=Other#
) AND o.ea_guid IN (
#DB=SQLSVR#
  substring(x.description,0,39),
  substring(xCon.description,39,39),
  substring(xCon.description,78,39),
  substring(xCon.description,117,39),
  substring(xCon.description,156,39),
  substring(xCon.description,195,39),
  substring(xCon.description,234,39),
  substring(xCon.description,273,39),
  substring(xCon.description,312,39),
  substring(xCon.description,351,39)
#DB=SQLSVR#
#DB=Other#
  left(xCon.description,38),
  mid(xCon.description,40,38),
  mid(xCon.description,79,38),
  mid(xCon.description,118,38),
  mid(xCon.description,157,38),
  mid(xCon.description,196,38),
  mid(xCon.description,235,38),
  mid(xCon.description,274,38),
  mid(xCon.description,313,38),
  mid(xCon.description,352,38)
#DB=Other#
)

UNION
SELECT DISTINCT o.ea_guid , o.Object_Type ,
  o.name,  o.Object_Type, o.stereotype, "Information Flow",
  c.Name, c.Stereotype

FROM t_object o, t_xref x, t_connector c

WHERE
  x.client = c.ea_guid AND
  x.Behavior = 'conveyed' AND
```

```
  c.connector_ID = <Search Term> AND
  o.ea_guid IN (
#DB=SQLSVR#
  substring(x.description,0,39),
  substring(xCon.description,39,39),
  substring(xCon.description,78,39),
  substring(xCon.description,117,39),
  substring(xCon.description,156,39),
  substring(xCon.description,195,39),
  substring(xCon.description,234,39),
  substring(xCon.description,273,39),
  substring(xCon.description,312,39),
  substring(xCon.description,351,39)
#DB=SQLSVR#
#DB=Other#
  left(xCon.description,38),
  mid(xCon.description,40,38),
  mid(xCon.description,79,38),
  mid(xCon.description,118,38),
  mid(xCon.description,157,38),
  mid(xCon.description,196,38),
  mid(xCon.description,235,38),
  mid(xCon.description,274,38),
  mid(xCon.description,313,38),
  mid(xCon.description,352,38)
#DB=Other#
)


order by 3,4,5
```

15.4.2 Information Flow Conveyed Script

To create the script follow these steps:

- Open the scripting window via `View/Scripting`.
- Create a new normal group (left icon) and name it e.g. 'Searches'.
- Add a new VB script (2nd icon) and name it e.g. 'Information Flow'.
- Double click the new entry to open the editor.
- Copy/Paste the following script into the editor and save the result (the famous diskette[5] symbol).

Now when you select the appropriate connector in a diagram just click the run button (4th icon) to execute the script. This will open the search window with the listed resulting conveyed classes.

[5] Yes, most of those diskettes were blue. And you could store more than one million bytes on it! Interestingly they still live on as icons.

```
option explicit

!INC Local Scripts.EAConstants-VBScript

sub main()
  dim selectedConnector as EA.Connector
  set selectedConnector = Repository.GetContextObject()

  if selectedConnector is nothing OR
     selectedConnector.ObjectType <> otConnector then
    Session.Prompt "You must select an Information Flow", promptOK
    exit sub
  end if
  dim id
  id = CStr(selectedConnector.ConnectorID)
  Repository.RunModelSearch "Elements on Flow", id, "", ""
end sub

main()
```

16. Further Reading

Finally I would like to add a few links where you will get further help.

16.1 Feedback

Any questions you have are important. So you should ask them. Feedback is important for you, me and of course all the other readers. So you are encouraged to send these to one of the below links:

http://leanpub.com/InsideEA
: The page where you bought the book has a small discussion forum enabled. Here you can also post.

thomas.kilian@me.com
: You can mail me directly if you have specific questions. Or maybe you have information regarding the ?! markers.

16.2 Scripting Enterprise Architect

I have not touched scripting much in this book. Basically because it would simply break the scope of this book. However, you might be interested in EA's automation interface. My book **Scripting Enterprise Architect** which is available at http://leanpub.com/ScriptingEA will introduce you in that matter. Even if you are already firm with the API its references and a couple of not well known hints will make this a valuable guide for you.

16.3 Sparx Forum

Probably not necessary to name this source, but anyway:

> https://www.sparxsystems.com/forums/smf/index.php

When you post your questions here you should select the right forum and only post once. Getting help here is most likely the fastest lane you can find. I also post regularly.

16.4 Sparx Community

A not so well known source as Sparx itself does not link this on its forum pages:

> http://community.sparxsystems.com

Here you will find a variety of articles and resources around EA.

16.5 SQL in General

Of course: Google is your friend. But sometimes it's nice to have a direct link:

> http://www.1keydata.com/sql/sql-commands.html

This is nice place to get a quick reference to most of SQL. This info is available in different languages if you enter via the main site.

> http://www.w3schools.com/sql/sql_syntax.asp

is a bit more responsive and condensed.

I once downloaded a compact HTML page with all information in it, but that site has ceased. So to get something similar see my remark above…

16.6 Geert Bellekens

Geert has become a kind of institution at Sparx' forum. He is a regular poster. But beyond that he also has an excellent blog dealing with UML in general and Enterprise Architect in particular. Check it out:

> https://bellekens.com

You'll also find entries dedicated to SQL usage in EA

> https://bellekens.com/#SQL

with some more advanced SQL stuff.