Industry Models and Assets

Industry Models Multi-Model Mapper (MMM) User's Guide

V3.0.1

V3.0.1 (July 2010)

References in the publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM deliverable in this publication is not intended to state or imply that only IBM's deliverables may be used. Any functionally equivalent deliverables may be used instead.

Publications are not generally available from the address given below. Requests for IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

IBM may have patents or pending applications covering subject in this document. The furnishing of this document does not give you any license to these patents.

Licensed Materials – Property of IBM

NOTICES

Trademarks and Service Marks

The following terms are trademarks or service marks of the IBM Corporation in the United States or other countries or both:

IBM, IAA, HPDM, Rational, Websphere

ERwin® is a registered trademark of Computer Associates International, Inc.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

ABOUT THIS BOOK

This MMM User's guide provides a general description of the Multi Model Mapper tool provided with the Insustry Models. It covers both the usage of MMM for an end-user perspective, and for a model management perspective.

Who Should Read This Book

This User's guide is intended for persons who want to gain a general understanding of the MMM tooling. The information in this guide should be especially useful to business users and analysts who want to manage the models and their inter-dependence (traceability) in a usable format.

Related Materials

IAA MMM and Product Model – Usage Guide
IAA MMM and CBM – Usage Guide
IAA Roadmap
IIW Roadmap
HPDM Roadmap

Table of Contents:

CHAPTER 1: OVERVIEW

Introduction

MMM is an application that is used to manage the common repository for all the models that make up an Industry Models edition.
It provides the following functionality:

- Meta-Model Editor
- Model Editor
- Domain Editor
- Package Editor
- View Editor
- Type and Type Property Editor
- Traceability across models and re-use of definitions
- Bridges to XML, Rational Software Architect® and ERwin®
- Generation of the documentation (Reference Manuals)
- Generation of the Hyperlinks (see below)
- Extended Copy Model
- Project Scope Propagator
- Compare Models

How does MMM help?
Having all the models in one consistent (relational) repository saves maintenance time and guarantees consistency. A relational environment also makes it possible to easily extract information from the models through queries and reports, as well as maintain CASE-tool independence.

What MMM is not?

- MMM is not a CASE tool, as it has no diagramming capabilities and therefore needs to be seen as a complementary tool to use with a CASE tool of choice.
- MMM is not an IBM product, as the source code is delivered with the tool and can be extended by the customer. The code is delivered on an “AS-IS” base.

The **hyperlinks** are a common HTML front-end across all the models. They are generated from the Multi-Model Mapper. They also include a complete index of the terms found in all of the models, providing an easy entry point into the models.

How do the Hyperlinks help?
The hyperlinks provide a consistent way of browsing through the models without the need for a licensed tool. By removing the dependency on a licensed tool, the hyperlinks provide an easy way to share models across an organisation.
In addition, the hyperlinks are an easy way to trace business content, from requirements to design constructs.

What was new in 2005

The main MMM new features that have been released, improved / enhanced in 2005 include:

Meta-Model customization has been extended for defining Association types, default Roles, and Type's default Type Property.

Project Scope view concept has been extended to easily and quickly define project boundaries.

Scope Propagator is a powerful feature that starts from a *Project Scope* view and navigates through the mapping (traceability) to define the equivalent scope in a target model.

Copy Model has been extended to support filtering based on *Project Scope* views with *Resulting Scope* button.

Reference Manual has been extended to support filtering based on *Project Scope* views with *Resulting Scope* button.

Import/Export XML (basic version) is a generic import/export mechanism based on a cross-reference table between XML tags and MMM constructs. This cross-reference table allows the definition of different XML structures such as XMI, MDX for XDE, PrimeEntity, etc. It currently covers basic information such as Name, Definition, and Stereotype.

Import/Export ERwin® has been extended to export the Subject Areas and to represent a Project Scope (colorize entities, setup the Logical-Only flag).

Import/Export SRI has been extended to support the Process Model (WBI Modeler)

Package Editor has been extended to support Package Mapping

View Editor has been extended to support View Links (associations between different view elements)

Type Editor has been extended to support filtering based on *Project Scope* views with *Resulting Scope* button.

Hyperlinks Generator has been extended to provide Tree views, left navigation panes, and new index facilities. Dedicated tree views are also available to represent the Process flows, the Product structures and the Business Solution Templates.

Free Format BID flag in the Object Type Editor makes it possible to import pre-allocated BID that do not comply to the Industry Models algorithm for allocating sequential numbers.

What was new in 2006

The main MMM new features in 2006 include:

Import/Export SRI
- supports the “Import as View” option
- supports the new naming conventions for RSA models

Hyperlinks
- Style Sheet (style.css) makes it possible to customize fonts, colors, background, in a very easy and efficient way.
- <Tag> in definitions are more sophisticated: look (in that order) for: Type, Type property (assuming unique), View, or Package. Also look if BID provided as tag (could be cross-model)
- Examples are kept in input sequence and not in alphabetical order
- mapping that provides definition inheritance is in italic
- generic section title "Source Mapping From" replaced by the actual Mapping Type (one section by mapping type)
- sequence nr not shown anymore in views
- support for free-format BID
- shows Processes in BST

Type Editor and Property Editor
- refresh problem fixed in Mapping Tab for recently added properties

Home page:
- Product Model home page has an entry for editing the Relationships
- Requirements Model home page has an entry for editing the Business Item
- Enterprise Component Blueprint home page is new

Import / Export ERwin®:
- handles the Optional attribute

BID Generator
- if numbering range reached (9999), seeks for unused BIDs (wholes in numbering)
- if numbers are already provided with another (non compliant) format, BID is reformatted with correct prefixes and length, trying to reuse provided numbers.

What is new in 2007

The main MMM new features in 2007 include:

Hyperlinks
- adaptations to 2007 model content

BID Generator
- Automated BID conversion when customizing prefixes and length.

Import / Export ERwin®:
- Support for Erwin® v7

Software pre-requisites

- MMM is a MS-Access application that requires MS-ACCESS® 2000 or higher.
- For performing the “Export ERwin 3.5” function, the ERwin® API (er2api32.dll) is needed. It is available with ERwin® 3.5.2 SP2 or higher.
- Erwin® 4.1.4 (build 4033) is the recommended version for MMM Erwin® 4 Import/Export functions while Erwin® 7.3.5.1945 is the recommended version for MMM Erwin® 7 Import/Export functions.
- For the Hyperlinks generator and XML bridge, the XML API (msxml.dll or msxml2.dll or msxml3.dll) is needed. Usually installed with Windows, but if missing, is available from the Microsoft Support site.
- For the Hyperlinks tree views supported by applets, Sun Java JRE 1.5 with latest updates.

CHAPTER 2: MODEL CONTENT MANAGEMENT

2.1 MMM Basic concepts

MMM mainly deals with four constructs:

- Package
- View
- Type
- Type property

For each of them, Object Types can be defined in order to extend their concept.

Because the MMM holds different kinds of models (requirement, business, and design), and since the MMM is CASE tool and notation independent, the term ***Type*** will be used to refer to an *Entity* or to a *Class*.
The term *Type* can also be used as a generic term for additional concepts such as *Business Process*, *Business Activity*, etc…, since the user can also define customised ***Object Type****s*.

The term ***Property*** will be used as a generic term to refer to an *Attribute* or to an *Operation*.
As with Types, the user can define additional *Object Types* for *Properties*, such as *Atomic Data Element, Business Service*, etc…

*The term **View*** will be used as a generic term to refer to a *Use Case, E/R Diagram, Component View*, etc…

Business Identifier (BID)
The BID is an identifier that is unique across all the models, and is independent of any CASE tool. It is used (among other things) to generate the Hyperlinks. The BID is composed of an object type prefix, a model prefix, and a number (see below, and refer to the *Standards and Naming Conventions* Appendix for more detail). The BID can be assigned manually, or via a batch allocation which will assign all BIDs for a given model at one time (discussed further in the Model Management chapter).

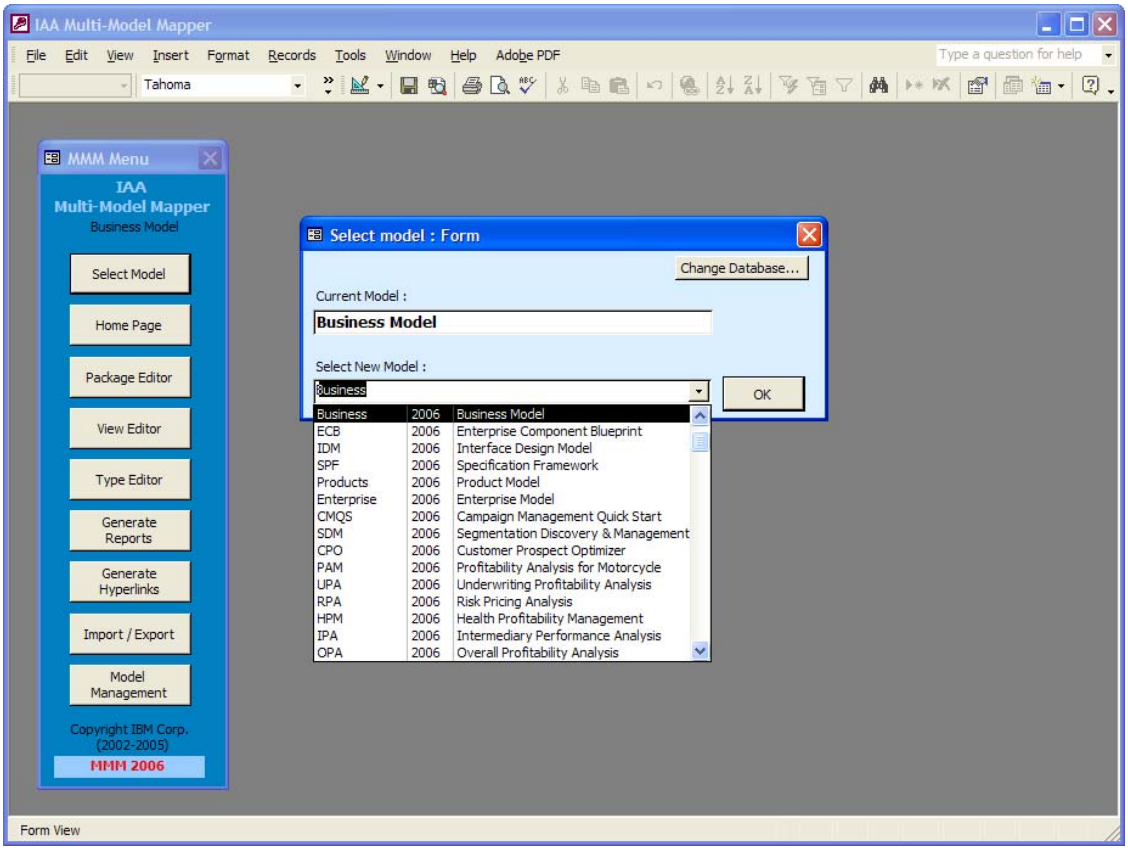
The BID will be formed by **XX**Ynnnn, where:

- **XX is the Object Type prefix**
- Y is the model prefix (defined in the Model Editor)
- nnnn is a meaningless sequential number

This list of prefixes can be updated for a specific project, according to the *Standards and Naming Conventions* appendix.

2.2 Getting Started

Since the MMM Repository contains a complete model set, the first step when starting MMM is to select the model to work with. The drop-down list displays all the models in the repository. Select one, and press OK.

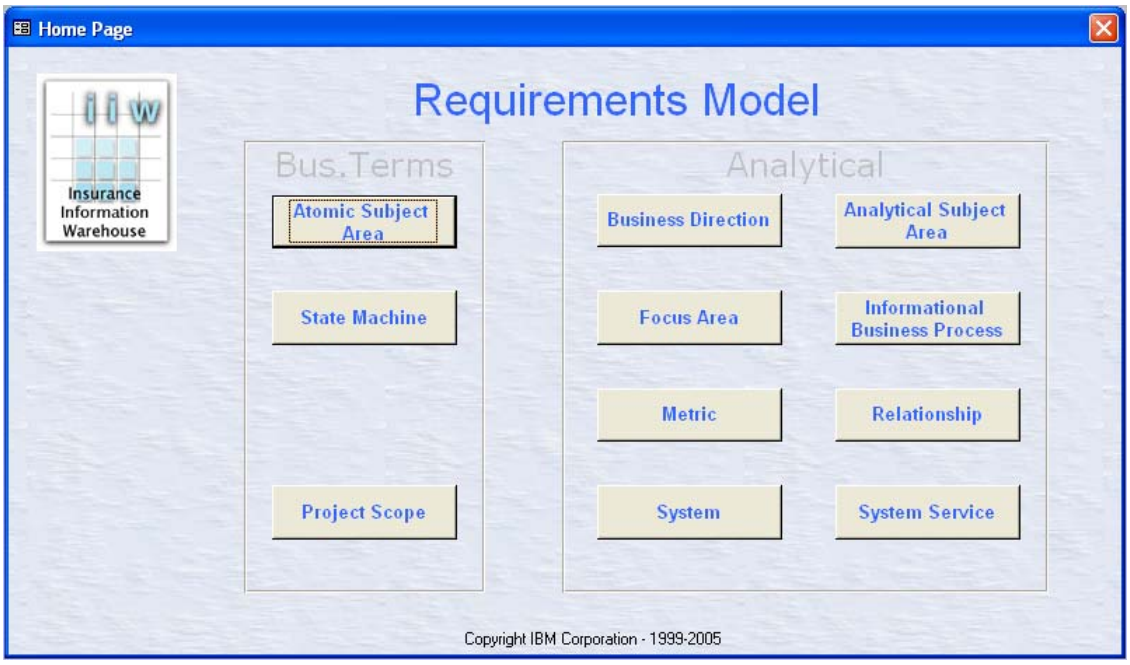


At this point, all further actions (Package Editor, View Editor, Type Editor, Import/Export, Reports, …) will be related to the current model.

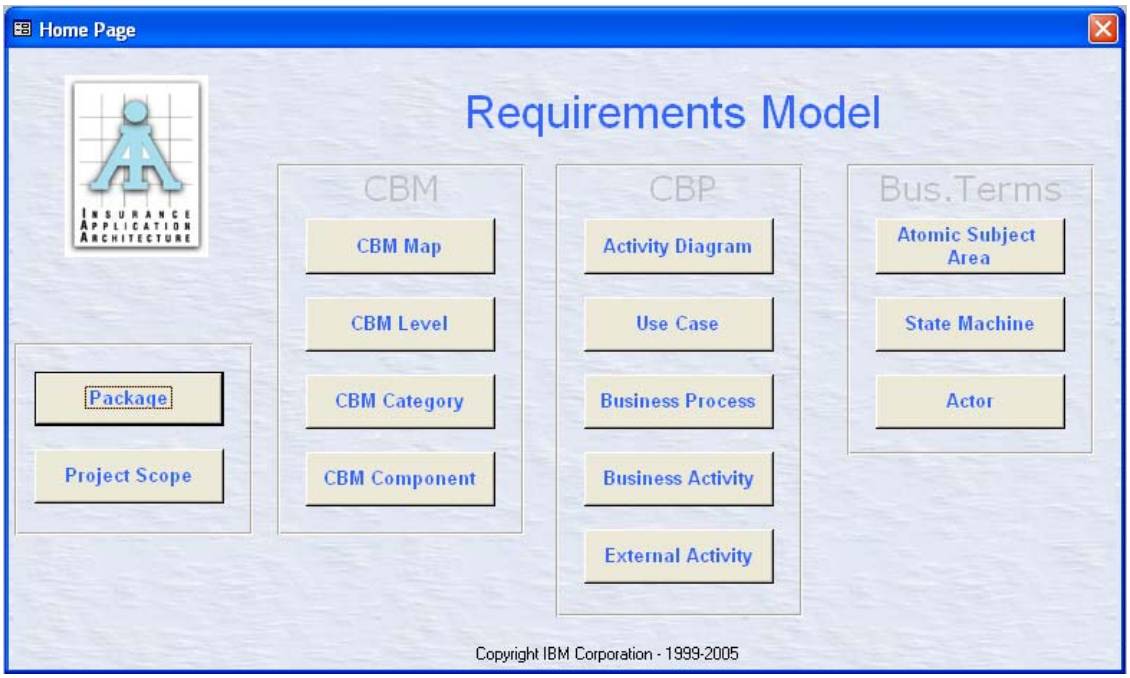
2.2 Home Page

Once the model has been selected, a Home Page can be displayed, presenting a simplified entry-point for the MMM end-user.
It shows the different object types used in the model as buttons that open the corresponding editor with the appropriate filters.

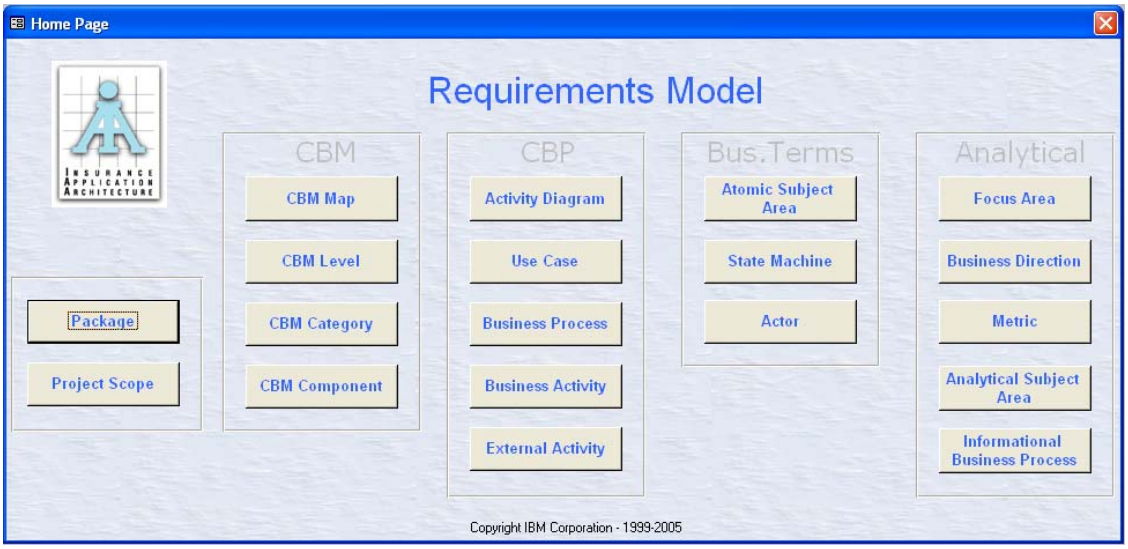
Here is an example of a customised Home Page for the Requirements Model in the context of an IIW implementation:



Here is an example of a customised Home Page for the Requirements Model in the context of an IAA implementation:



Here is an example of a customised Home Page for the Requirements Model in the full INDUSTRY MODELS CD:



The Home Page forms can be customised or created by opening the forms in *design mode* within MS-Access. The form name should contain the “Home Page” keyword.

The reference to a Home Page is specified in the Model Editor. (See section 3.1)

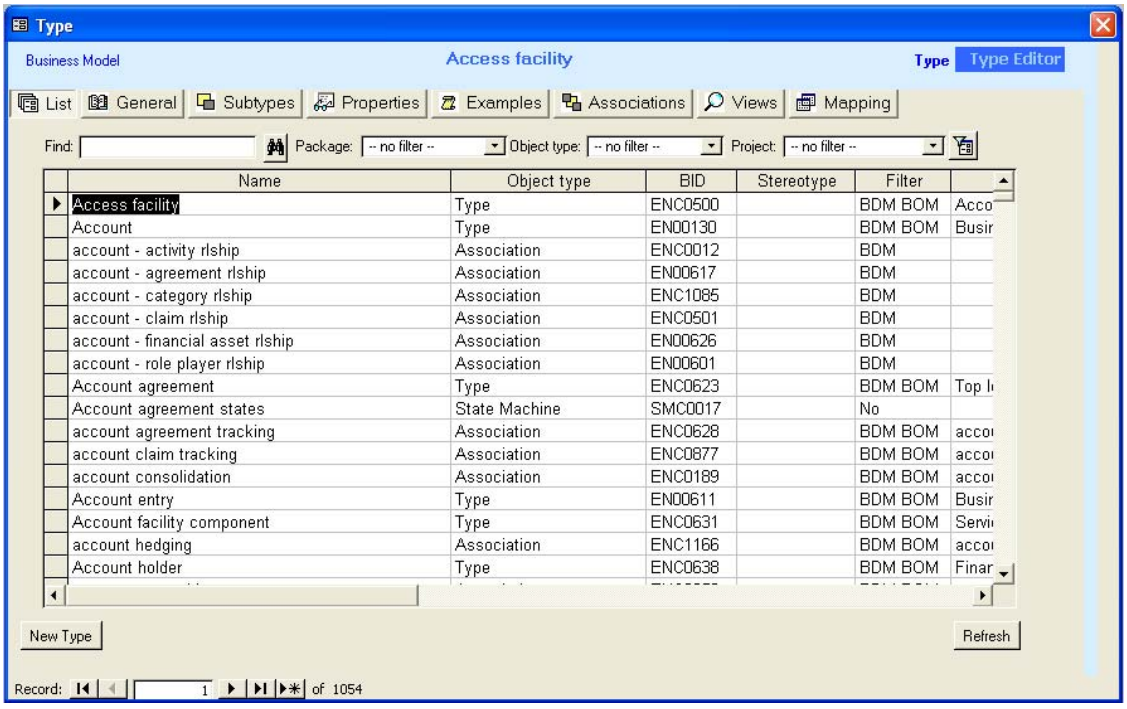
2.3 Type Editor

Within the MMM, the Type Editor is the primary focal point. In addition to listing all of the Types in the current model, the Type Editor can be used to edit the definitions, examples, comments, business identifier (BID), super type, associations and mappings of a given Type.

The Type Editor is a generic editor, used to manage the various kinds of Types that have been defined via the Object Type Editor in the Model Management tool such as: "**Type**", "**Association**", "**Relationship**", "**Atomic Subject Area**", "**Business Process**", "**Business Activity**", "**Business Function**", and so on.

The functions within the Type Editor are organised in a series of tabs: **List**, **General**, **Subtypes**, **Properties**, **Examples**, **Associations, Views, Mapping**, and the optionally displayed, **Requirements** tab.
Below is an explanation of each of the Type Editor tabs.

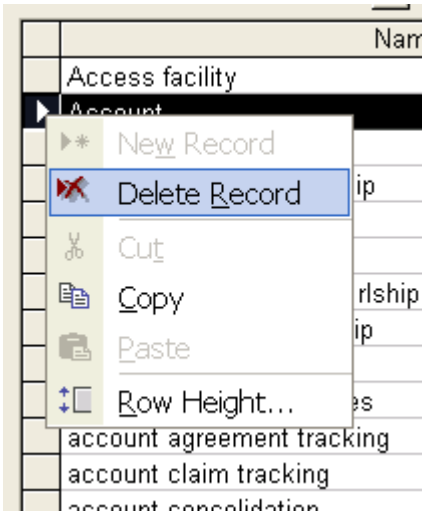
Type Editor: List Tab



From the **List** tab, single-click on the Type name to open the **General** tab where the Type's details can be edited, or a new Type can be added.

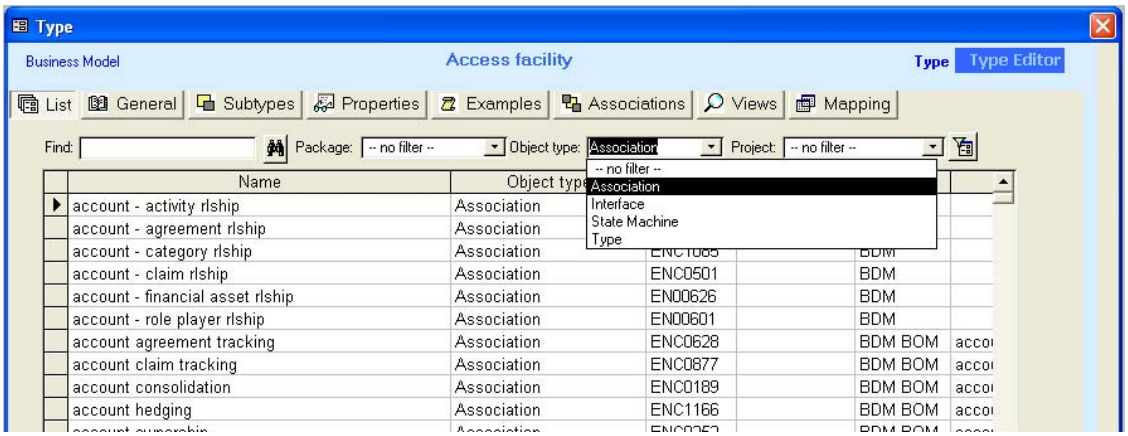
For help retrieving a certain Type in a long list, there is a **Find** function. No wild-card character is needed if you want to type only the few first letters of a name.

The **List** tab also allows a Type to be deleted: right-click on the row margin (where a black arrow appears in front of the selected row) and choose *Delete Record.* This is captured in the screenshot below.

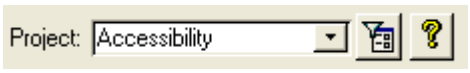


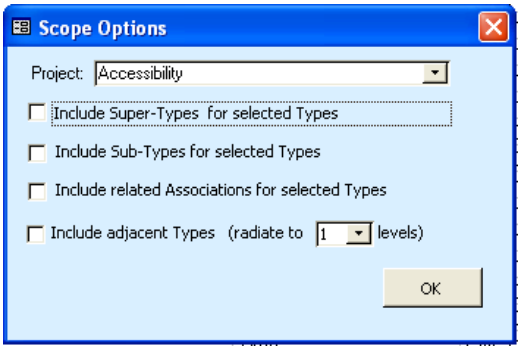
The type list can be sorted on any column of the List tab by right-clicking on the column header, then choose *Sort Ascending* or *Sort Descending*. You can therefore **sort** by Name, by BID, by Association flag, and so on.

Filters:
There are three exclusive filtering mechanisms: by **Package**, by **Object type**, or by **Project**. The Project filter can actually filter on other views than Project Scope view, by selecting -more-
.



When filtering by Project, the Project scope options can be specified and the resulting scope log can be shown. This log explains the ”why” of every instance in the scope.

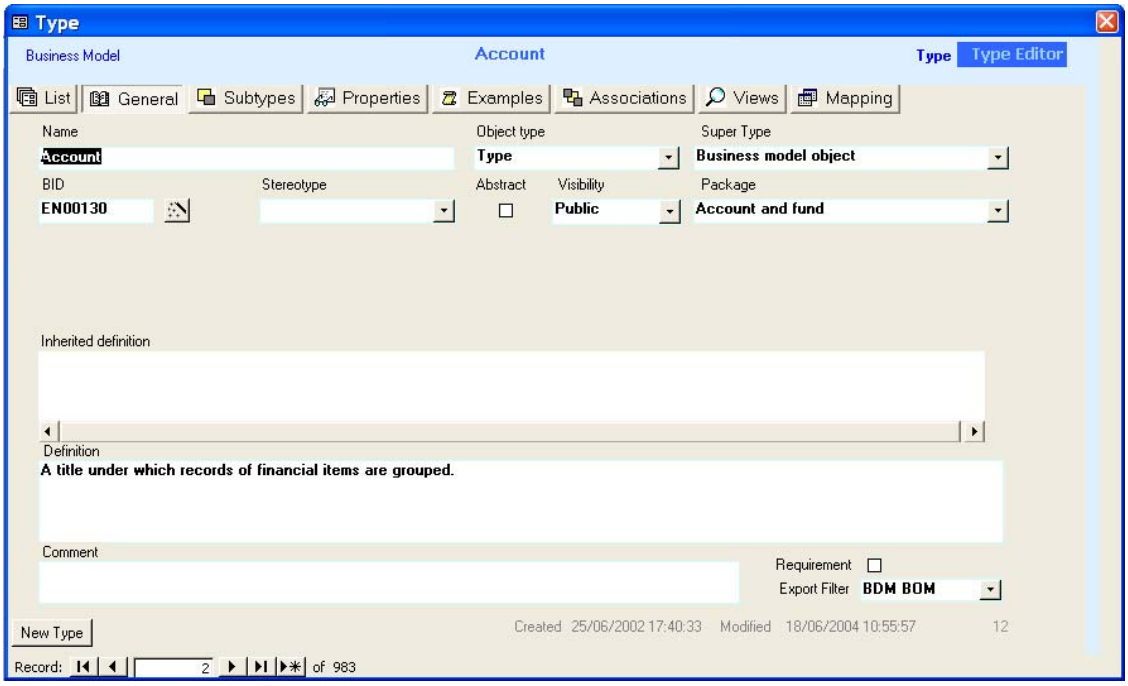




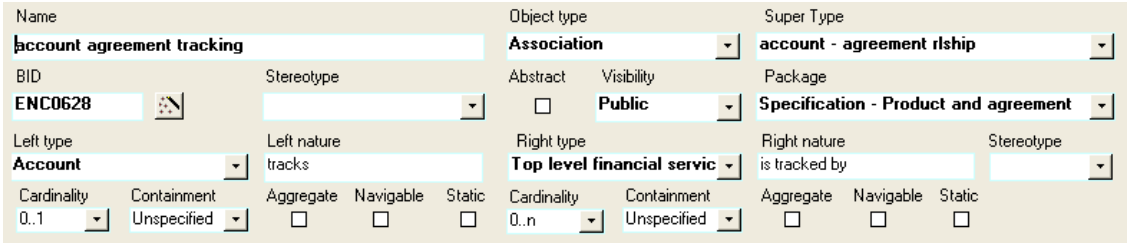
Type Editor: General Tab

The General tab is a common screen for editing both Types and Associations.

To create a new Type or an Association, click on the New Type button, and complete the entry fields. The minimum information required is the Type's **name**. If a name is not provided, the MMM will default the name to the BID number.



If the Object Type **Association** has been selected, an additional set of entry fields is displayed:



These additional fields capture the Association's specifications:

- Both the Left Type and the Right Type can be selected using a drop-down list
- In addition, both the Left Nature and the Right Nature can be specified, as well as cardinality, containment, static and navigability rules.

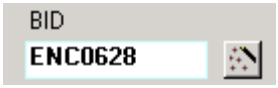
You can double-click on the name of the Left or Right Type to open the Type Editor for it.

Package
A Type or Association can belong to (only) one Package. The drop-down list displays all packages defined in the current model.

Super Type
The Super Type drop-down list is used to build the Type Hierarchy. By specifying a super type here, the current Type will then be shown (via derived information) in the Type Editor's

Subtypes tab of the given Super Type. To view the Super Type, double-click on the Super Type name, and the MMM will open the Type Editor for it.

Business Identifier (BID)
As mentioned earlier, the BID is composed of an object type prefix, a model prefix, and a number.
This field can be used to allocate the BID while editing the Type, either manually, or using the Wizard button (see below). If using the Wizard, simply provide the prefix (for example **ENC**) and press the button. If a prefix is not provided, the wizard will use the default prefix (refer to the Model Editor Chapter).



Cardinality
Indicates the number of links between parent and child(ren): 0..1 0..n 1..n

Abstract
Flag to specify whether a Type is abstract or not.

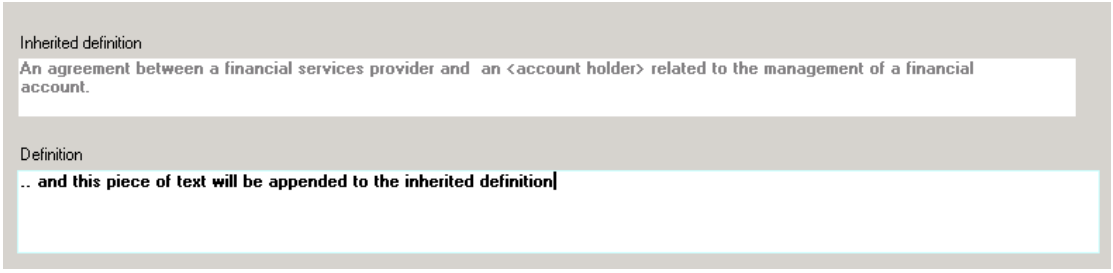
Static
Flag to specify whether a Type is static or not.

Aggregate and **Containment**
Flag and code to specify aggregation *byReference*, *byValue* or *Unspecified*.

Navigable
Flag to specify whether an association end is navigable or not.

Visibility
Also known as “Access specifier” can be either Public, Private, or Protected.

Definition and Inherited Definition
There are two ways to define a Type: by inheritance from another model, or by providing a definition at this level in the **Definition** field. A definition is inherited via the mapping information captured on the **Mappings** tab (discussed below in more detail in the **Mapping Tab** section); therefore, the **Inherited Definition** field is read-only. The **Definition** field can be used to further qualify the **Inherited Definition**. It is an optional field and is considered an addendum to the inherited definition.



If the text of the definition includes a Type name, by inserting the **<**type name**>** between angle brackets, the given Type will appear as hypertext in the Hyperlinks Navigator.
A formula in the definition is expressed between square brackets. For example [A+B-C] where the letters correspond to mapping labels. (See the *Mapping Tab* section). The formula will appear as hypertext in a frame in the Hyperlinks Navigator.

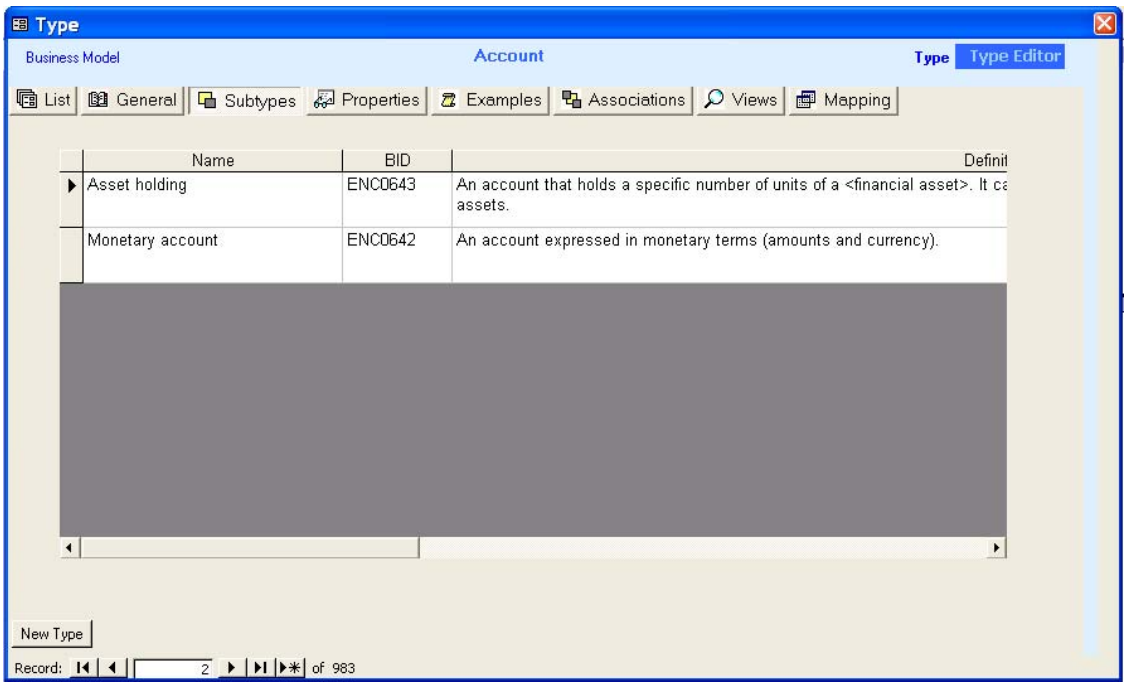
Export Filter

This field can be used to supply a value to be used in the Export functions to filter out what needs to be exported. By default, the keyword “All” specifies that the Type will be exported in every case. For example, if the current Type should be exported for XXX and YYY, ”*XXX YYY*” should be specified in the Export Filter.
See *the Hints and Tips* appendix for additional explanation on how to use the Filter

Comment
This field can be used to store a comment about the Type. Note however, that this information is not exported to any CASE tool or report, and does not appear in the Hyperlinks Navigator.

Type Editor: Subtypes Tab

The **Subtypes** tab lists the sub-types of the current Type and presents the information as read-only because this information is derived from the Super-Type field on the **General Tab** (discussed above).



Single-clicking on the subtype name (for example, “Asset holding” in the screenshot above), will open the subtype in the Type Editor.

Type Editor: Properties Tab

The Properties Tab contains the Property Editor. The **Property Editor** lists the properties (attributes / operations) of a selected Type and allows the user to edit the Property’s details. The Property editor is discussed in detail, below, in section 2.3.

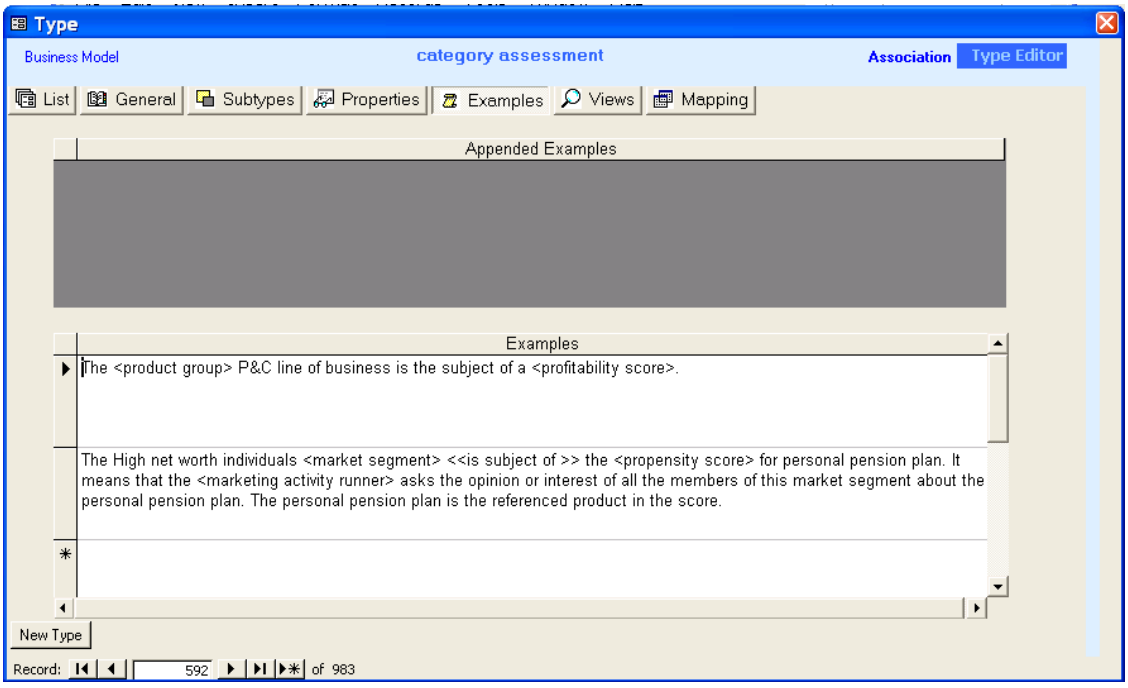
Type Editor: Examples Tab

As the name indicates, the **Examples Tab** provides examples of the Type.

As with the **Definition** field on the General Tab of the Type Editor, Examples can be both inherited from another model and specified at this level.

An **Appended Example** is inherited via the same process as inherited definitions, through the mapping information captured on the **Mappings** tab (discussed below in more detail in the **Mapping Tab** section). And because they are inherited, **Appended Examples** cannot be modified from this Tab; they are displayed as read-only text.
Additional **Examples** however can be specified at this level to extend any inherited ones if required.

Each Example is a record in MS Access, so new Examples should be entered on the row with the * in the left margin.



To delete an example, right-click on the row margin, and select *Delete Record*.

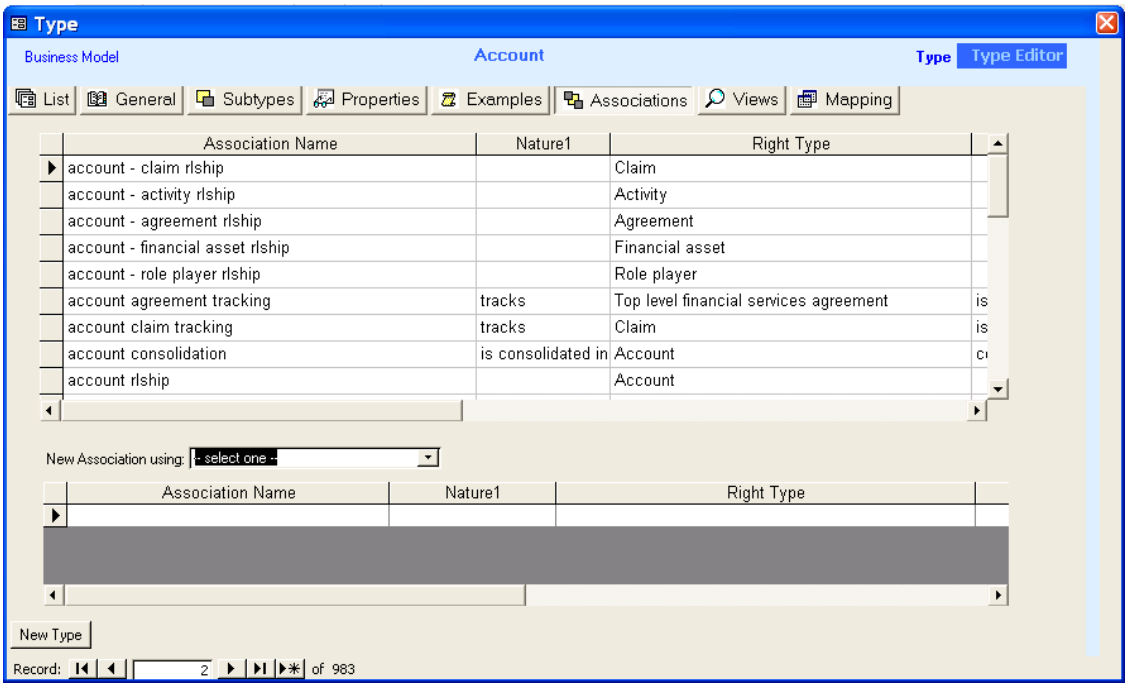
Type Editor: Associations Tab

The Associations tab is only shown in the Type Editor for a Type, and not for an Association.

The first section provides read-only information, listing the associations and their natures, for the current Type. This information is derived from the Left and Right Type fields in the **General** tab of an Association.

The second section provides a data entry sheet, which is preformatted with the default natures (role names) according to the selected target Type.

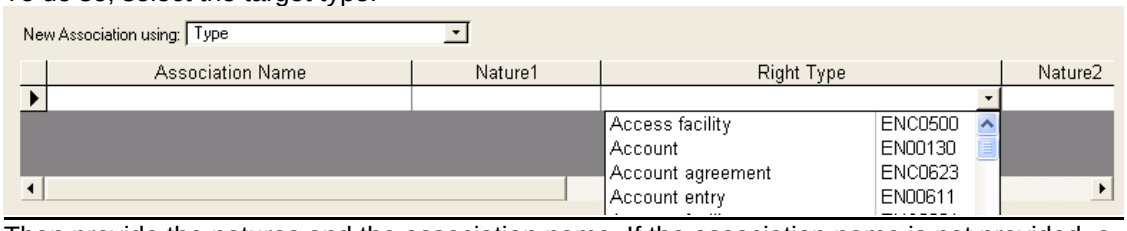
If more information is desired about any of the entities listed in the **Association Name** column, a single-click on the Association name, or Right/Left Type name will open the Type Editor for that association or Type.



In order to create an Association, you can use the Type Editor as explained on *page 15* (which consists in pressing  and selecting Object Type “Association”),

but you can also use this quick way of creating an Association from the current Type (left side of the association) and select a target (right side of the association).

To do so, select the target type:

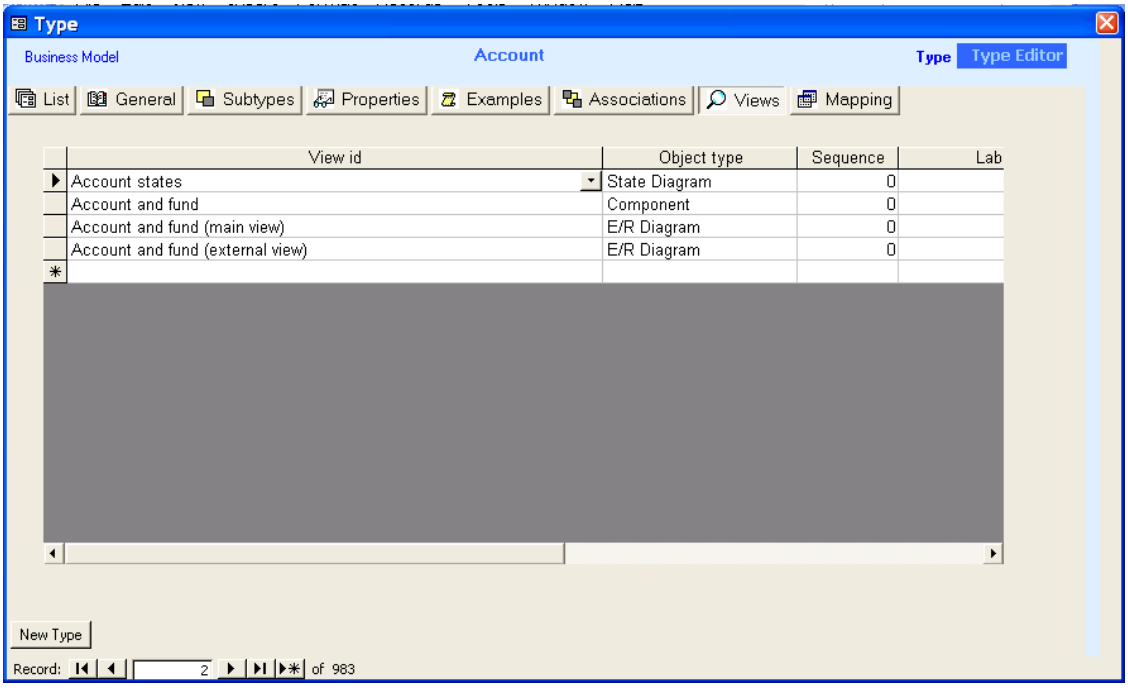


Then provide the natures and the association name. If the association name is not provided, a generated name will be supplied with the following pattern:
left type name + nature1 + right type name.

Note: The pre-defined targets and natures can be customised with the **Association types Editor** (see *section 3.9*)

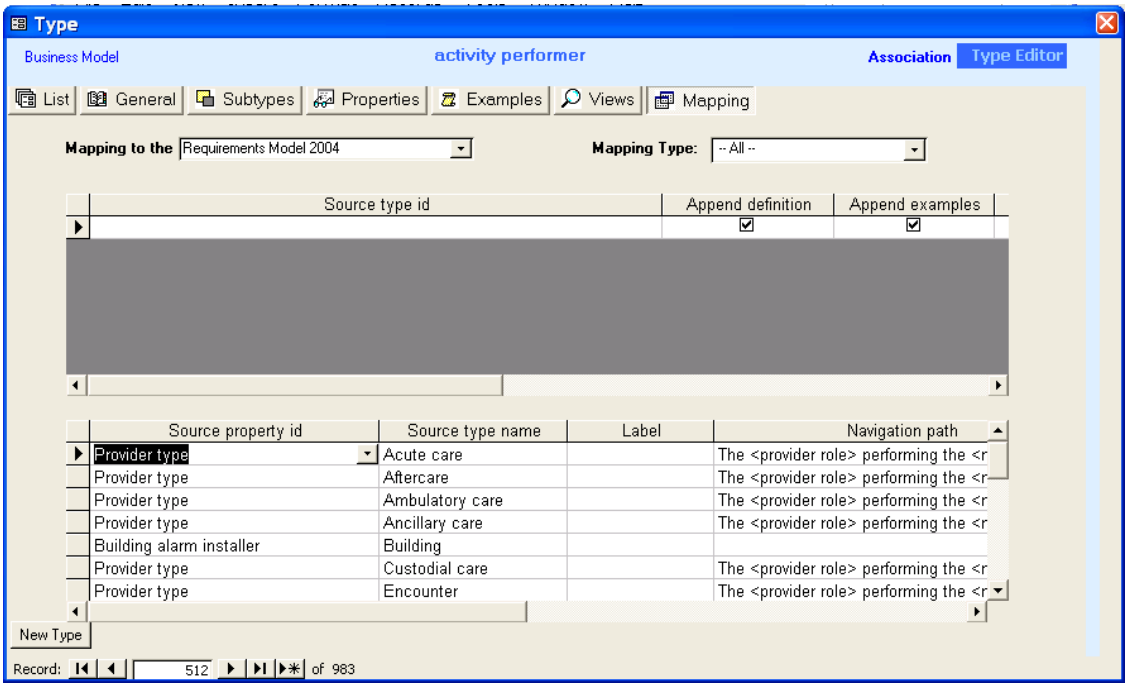
Type Editor: Views Tab

A Type can be included in one or more views. This reference to a view can be done using this Tab, as well as by using the View Editor.



Type Editor: Mapping Tab

A Type can be mapped to one or more Types or Properties, from one or more models. Select the Source model for the mapping first. Next select the Type or the Property from the selected model.
Once specified in the Mapping Tab, the navigation will be mapped across the appropriate models in the Hyperlinks Navigator.



Append definition
Checking this flag specifies that the Type should inherit the definition(s) from source Type(s).

Append examples
Checking this flag specifies that the Type should inherit the examples from source Type(s).

Mapping type
This drop-down field is populated with the following values (if exist for the current model):

- **Transformation**: traces information between models as they evolve from Requirements to Business to Design to Implementation. (e.g. Enterprise Model versus Business Model)
- **Version**: tracks changes between different versions of the same model (e.g. Business Model 2004 versus Business Model 2005)
- **Reference**: provides a single link between two related models (e.g. AcmeBM versus IAA_BM)
- **Translation**: provides a single link between two models that differ by language (e.g. the English version of a model and a translated version of the model)
- **Scoping**: provides a single link between a full model and a subset of a model (e.g. a project which is a subset of an Enterprise model)
- **Population**: provides a single link between a Source model and a Target model, in terms of population, for documenting an extract-transform-load (ETL) process (e.g. Operational data to an Enterprise Data Warehouse
- **Alias**: provides a single link between two terms that are synonyms.
- **User-defined**: you can enter your own Mapping type

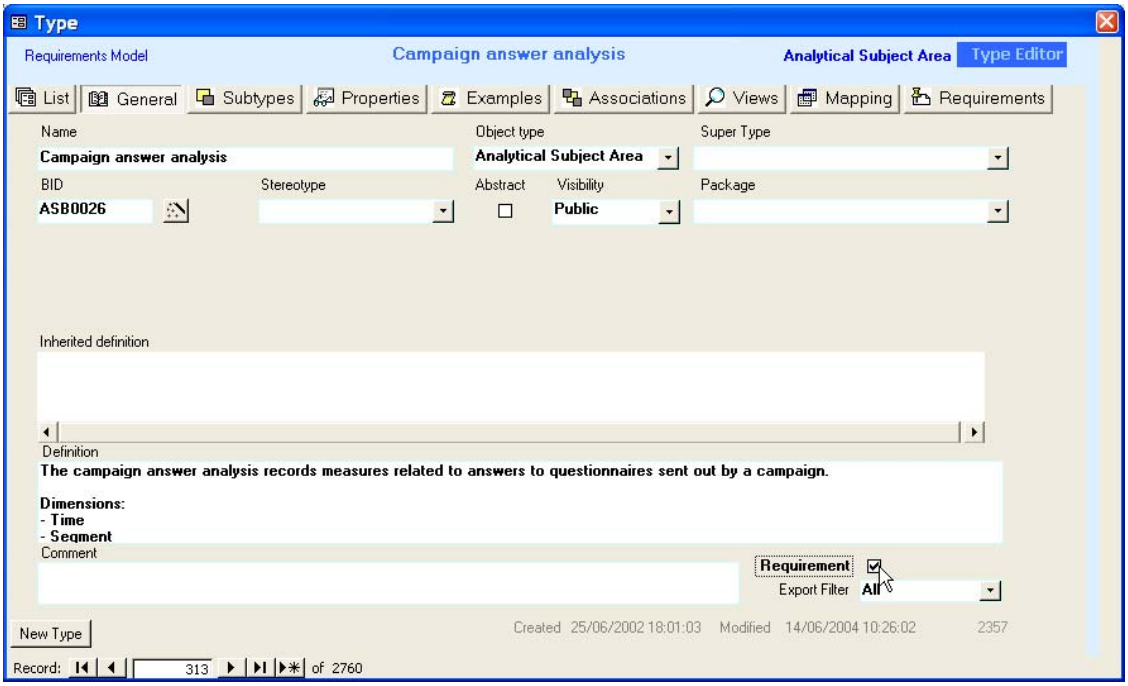
Label
This field can be used to provide a unique label to multiple similar mappings. It can be used for example to differentiate several mappings that have the same source and the same target. This label can also be referenced in a formula expressed in a definition. (See Type Definition)

Navigation Path
This field can be used to document the mapping more precisely. In addition, the navigation path can utilise the Hyperlinks navigation by inserting Type names between angle brackets (e.g. **<**Party>), which will create a hypertext link to the given Type if it exists within the same model.

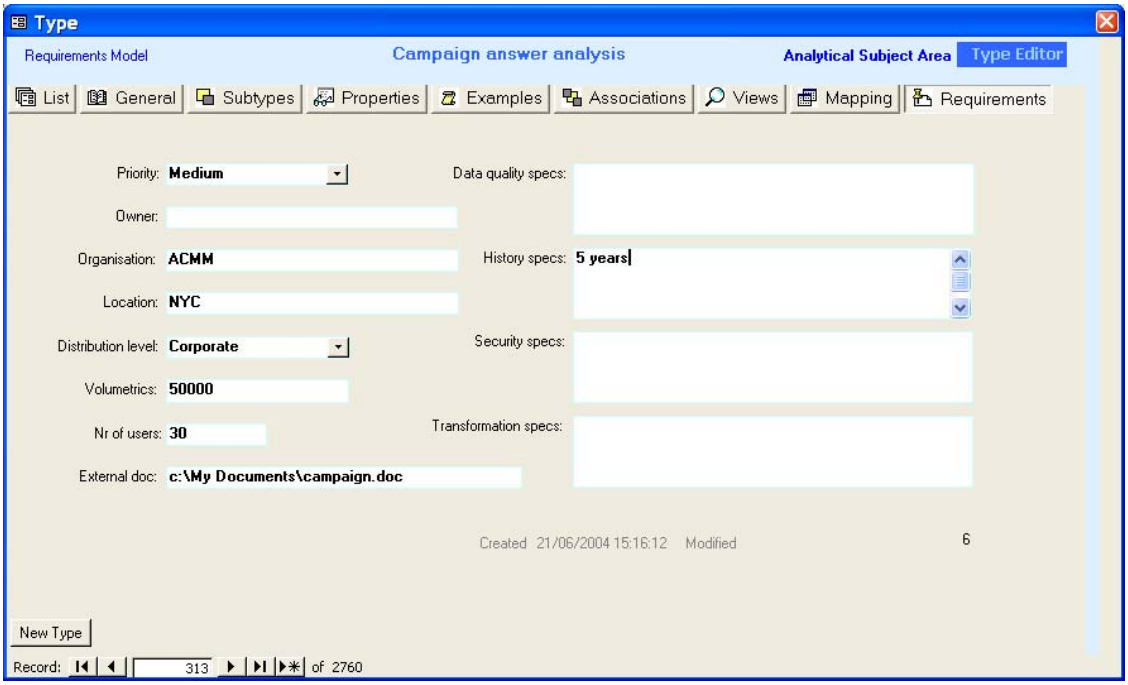
Transformation Rules
This field can be used to document the transformation applied more precisely. In addition, the transformation rules can utilise the Hyperlinks navigation by inserting Type names between angle brackets (e.g. **<**Party>), which will create a hypertext link to the given Type if it exists within the same model.

Type Editor: Requirements Tab

The Requirements Tab captures additional information such as volumetrics, data quality specifications, and history specifications, etc… that is needed by the Requirements Model. This is an optionally displayed Tab of the Type Editor, triggered by the Requirement flag on the General Tab. In case of a *Requirements* model, this tab is always visible.



If the **Requirement** flag has been selected in the **General** tab, an additional tab can be seen in the Type Editor:



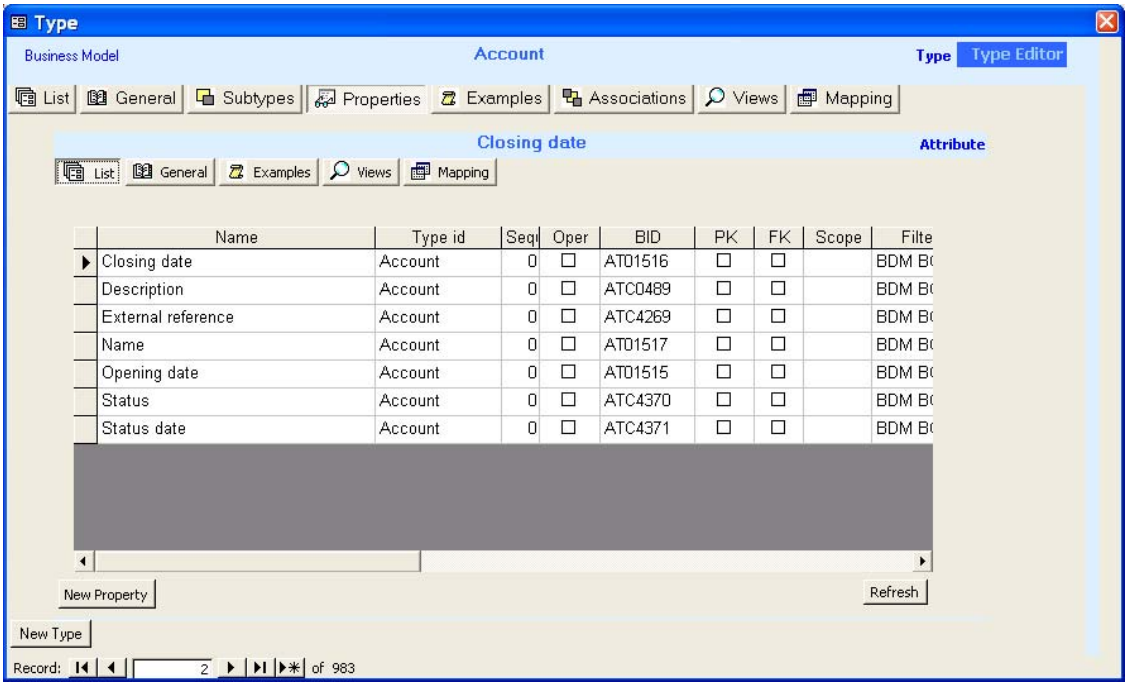
2.4 Property Editor

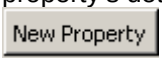
Included within the Type Editor is the Property Editor, which lists the properties (attributes / operations) of a selected Type. The Property Editor allows the user to edit the property definitions, examples, comments, business identifier (BID) and mappings.

Clicking on the **Properties** tab from the Type Editor opens the Property Editor.

The design of the Property Editor is very similar to the Type Editor; it includes a series of tabs, that organize its functions: **List**, **General**, **Examples** and **Mapping** tabs, as well as an optionally displayed **Requirements** tab.

Property Editor: List Tab

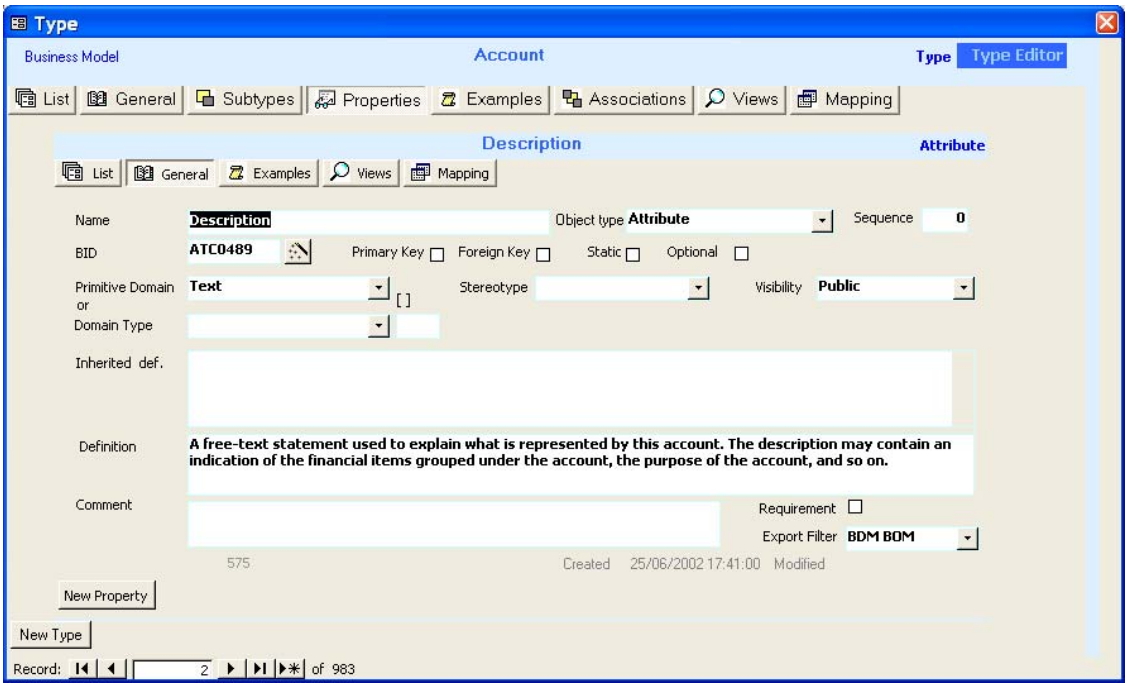


A property can either be created, modified, or deleted from the **List** tab of the Property Editor. To edit an existing property, single-click on the Property name to open the **General** tab where the property's details are displayed, and can be modified. To create a new property, click on the 

button.

To delete a Property, right-click on the row margin (where a black arrow appears in front of the selected row) and choose *Delete Record.*

Property Editor: General Tab



The Property Editor is used to create and maintain both Attributes and Operations.

Name: The name of the property.

Object type: Shows the object types defined in the Object Type Editor for Properties, such as: Attribute, Operation, Atomic data element, and so on.

Sequence: makes it possible to force the sequence in the list of properties in the Hyperlinks Navigator.

BID: The unique Business Identifier. The wizard has the same behaviour as for Types.

Visibility: also known as “Access specifier” can be either Public, Private, Protected.

Primary Key: a flag to indicate that the attribute is part of the primary key.

Foreign key: a flag to indicate that the attribute is a foreign key.

Static: a flag to specify the attribute being static.

Optional: a flag to specify the attribute being optional.

Primitive Domain or **Domain Type**: Some models, such as the Interface Design Model (IDM) do not use the MMM Primitive Domain types, but rather define their own “Common Data Types” as Types in the model itself.
This is why, in the Property Editor, both entry fields are available:
- Primitive Domain (a drop-down list of all domains defined in the MMM Domain Editor)
- Domain Type (a drop-down list of all Types defined in the current model)

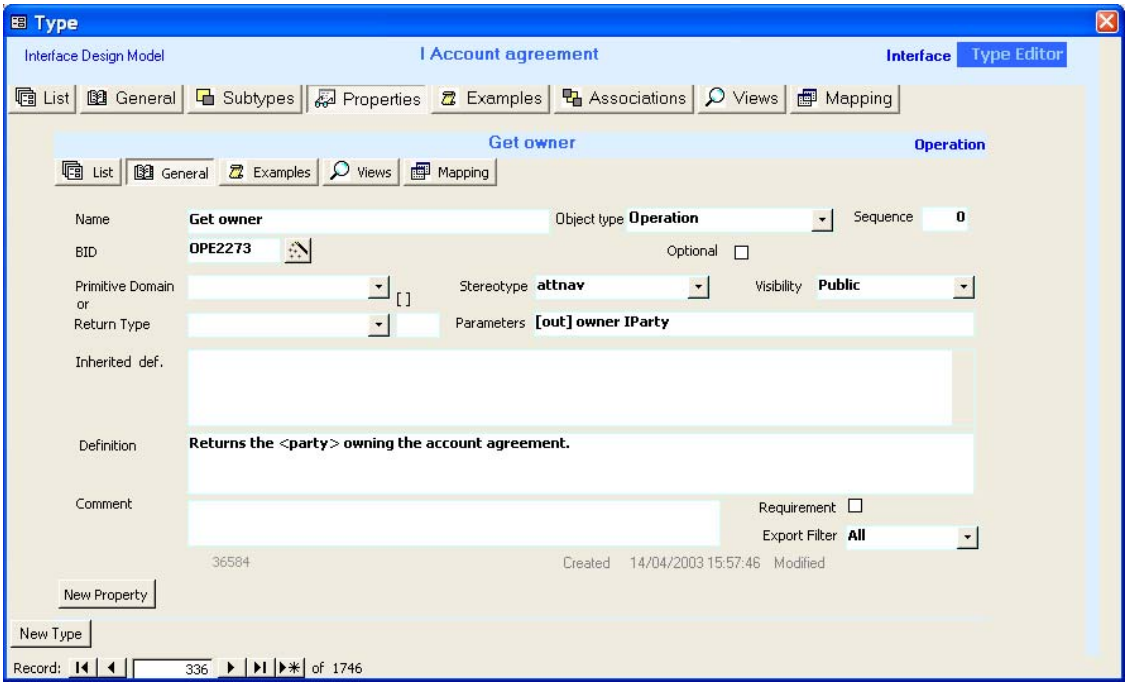
Inherited Definition and **Definition**: The inheritance mechanism for definitions is also similar to the Type Editor: a property (an attribute or an operation) can inherit the definition from an attribute or an operation to which it is mapped. Likewise, specifying a **<**Type name**>** between angle brackets in the definition will create an active hypertext in the Hyperlinks Navigator..

When the Object Type is **Operation**, additional entry fields are displayed. The **Return Type** field replaces the **Domain Type** field, and a **Parameters** field is displayed.

If the **Return Type** is a collection of Return types, specify **[ ]** (square brackets) in the field next to the Return type.

Parameters are to be written in the following format: [*directionality1*] *paramName1 Type1,* [*directionality2*] *paramName2 Type2,and so on.*
The directionality can be **[in]**, **[out]**, or **[in out]**.

The screenshot below, for example, shows an output parameter named *owner*, of Type *IParty*. A blank separates a name and a Type, and a comma separates each name/Type pair.



Property Editor: Examples Tab

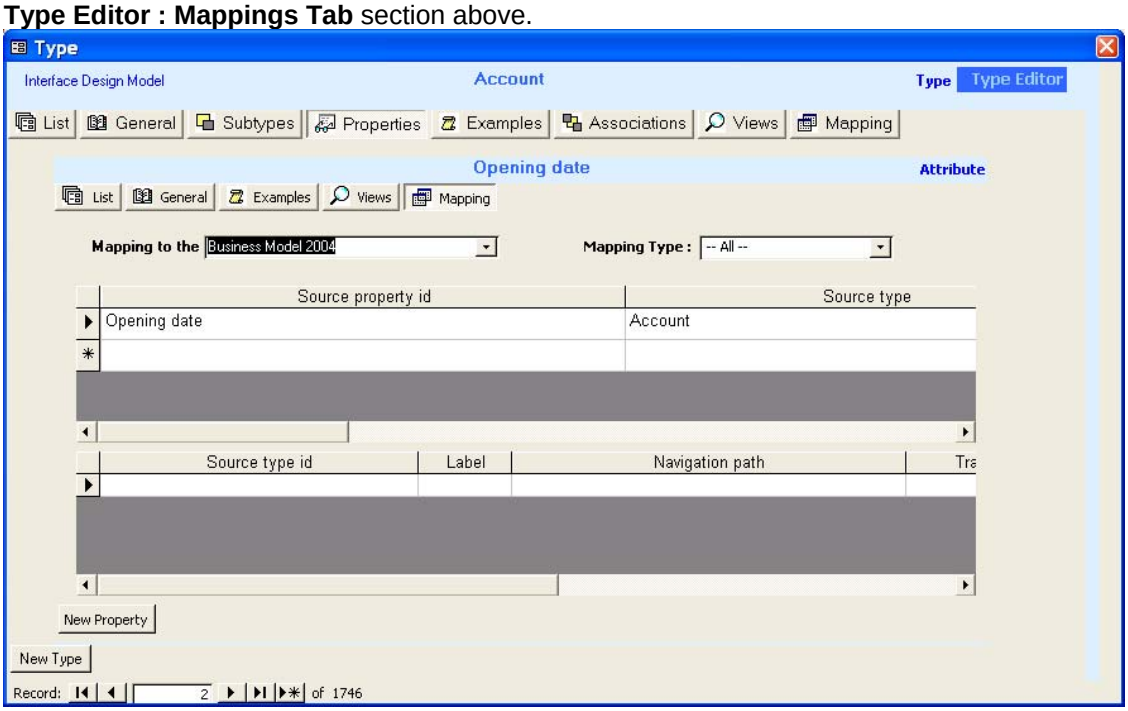
The **Examples Tab** in the Property Editor is nearly identical to the **Examples Tab** in the **Type Editor** except that it provides examples of properties. The Examples can be both inherited from another model, according to the mapping, or specified at this level. Both are displayed on this tab, with Inherited Examples shown as read-only. For more information, refer to the discussion above, of the **Examples Tab** in the **Type Editor.**

Property Editor: Views Tab

The **Views Tab** in the Property Editor is nearly identical to the **Views Tab** in the **Type Editor** except that it allows inclusion of properties in a View. The same functionality is available from the View Editor**.**

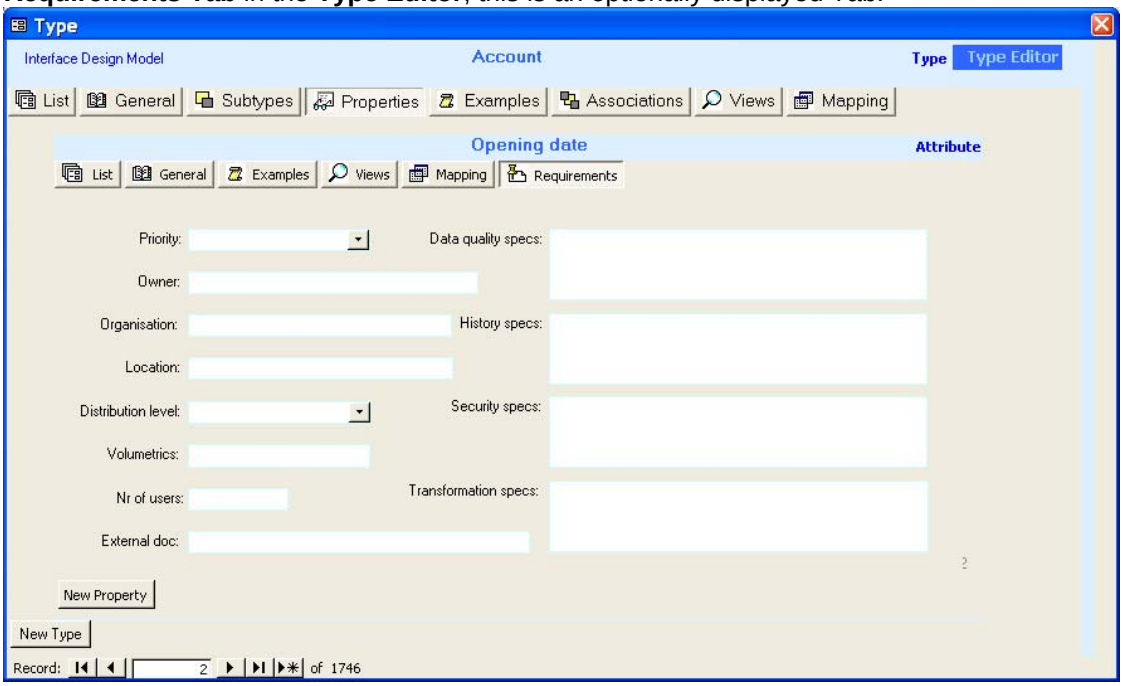
Property Editor: Mappings Tab

As seen in the screenshot below, the **Mappings Tab** in the Property Editor is nearly identical to the **Mapping Tab** in the **Type Editor,** except that it maps Properties instead of Types. As with the mapping of Types, a Property can be mapped to one or more Properties or Types, from one or more models. For a more detailed discussion of the **Mappings Tab**, refer to the **Type Editor : Mappings Tab** section above.



Property Editor: Requirements Tab

As with the **Examples** and **Mapping Tabs**, the Property Editor's **Requirements Tab** is very similar to the Requirements tab in the Type Editor. It makes it possible to express specific business requirements at the level of an Atomic Data Element or Measure Element. Like the **Requirements Tab** in the **Type Editor**, this is an optionally displayed Tab.



2.5 Package Editor

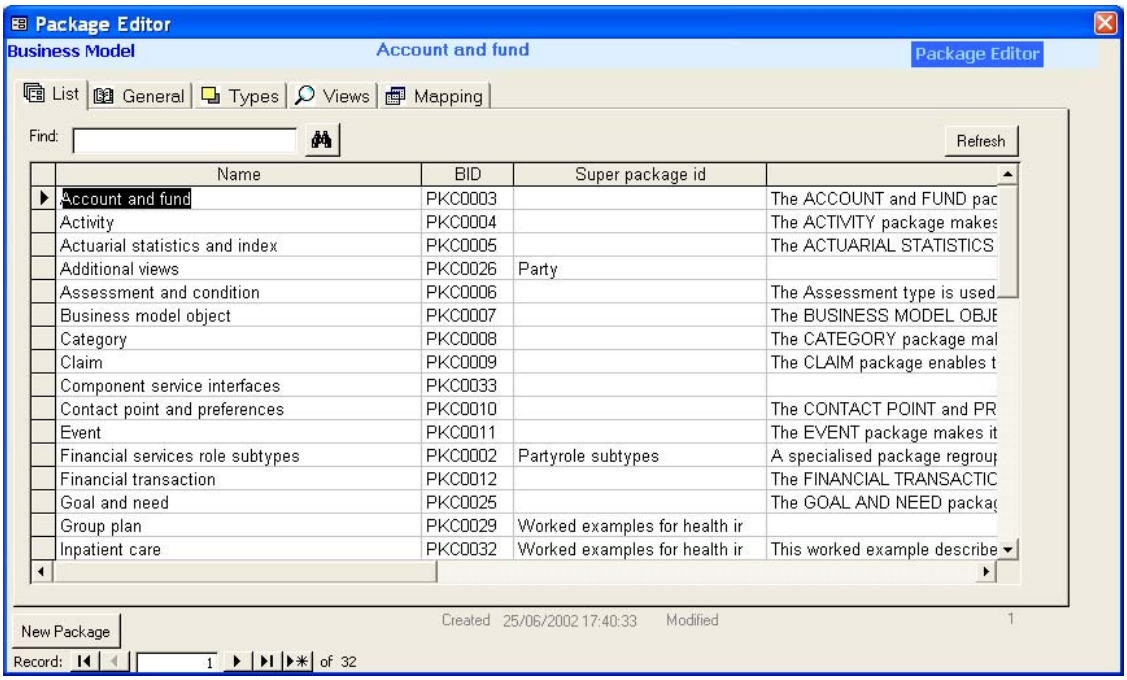
The MMM's Package Editor is used to manage a model's packages, if packages have been defined for that model.

Although Packages are not mandatory, they are useful for grouping related Types together.

MMM Packages are different from ERwin® Subject Areas, however, where an entity can belong to one or more subject areas. ERwin® Subject Areas are defined in MMM as Views (of Object Type "E/R Diagram").

The Package Editor is organised into four tabs: the **List**, **General**, **Types** and **Views** Tab.

Package Editor: List Tab



The **List Tab** in the Package Editor contains the complete list of Packages for a given model. In addition to the package's **Name**, the Tab includes other details pertinent to the package such as the **BID**, **Super package Id**, and a **Definition** of the package.

The **Find** function can be used to search for a specific value in any of the columns on the **List Tab**. For example, if the Package list is especially long, the Find function can be used to locate a specific package. In addition, the Package list can be sorted on any column of the **List Tab** by right clicking on the column header, and choosing *Sort Ascending* or *Sort Descending*. This allows the user to sort the packages by Name, BID, and so on.

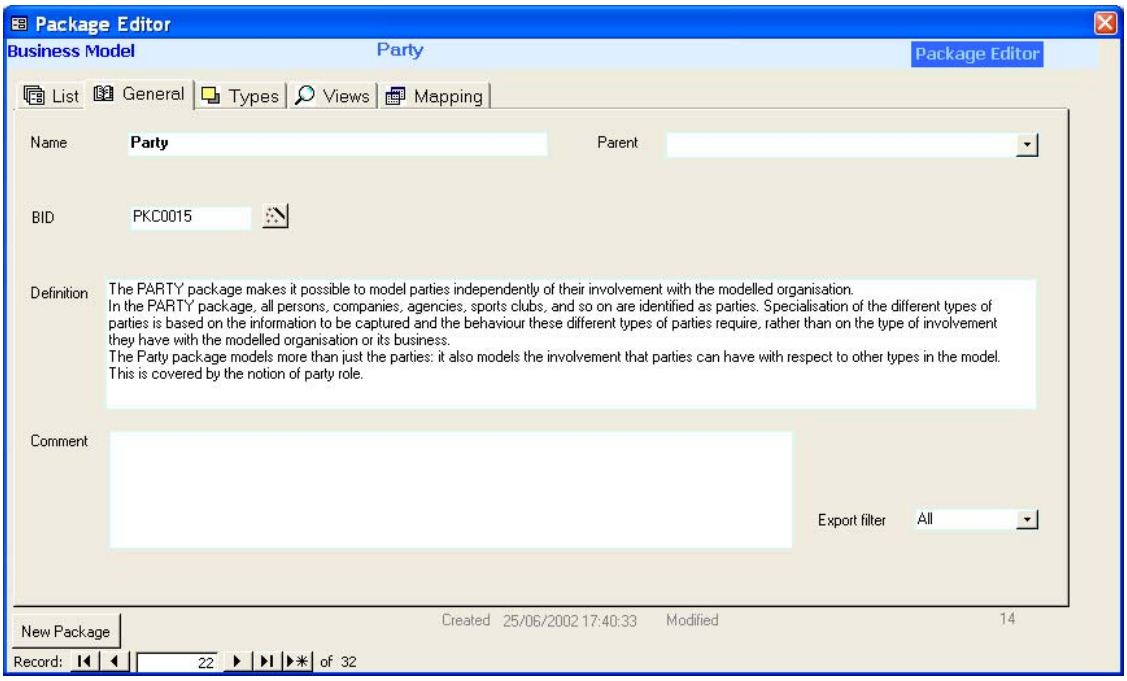
A Package can also be deleted from the **List Tab**, by right clicking on the row margin (where a black arrow appears in front of the selected row) and choosing the *Delete Record* option. As a precaution, a window containing a warning will be displayed whenever one tries to delete a package which still contains Types.

When on the **List Tab**, single clicking on the Package name will open that package in the **General Tab** where the package details can be maintained.

Package Editor: General Tab

The **General Tab** of the Package Editor is where a Package's details can be edited, or a new Package added.

For creating a new Package, click on the New Package button (see below for screenshot), which will display a new record where the package details can be entered. The minimum information required is a package name, and if not provided manually, it will be defaulted to the BID number.



Definition
The Package **Definition** is used to give a textual description of the Package. As elsewhere throughout the MMM, specifying a **<**Type name**>** between angle brackets in the definition will create an active hypertext in the Hyperlinks Navigator.

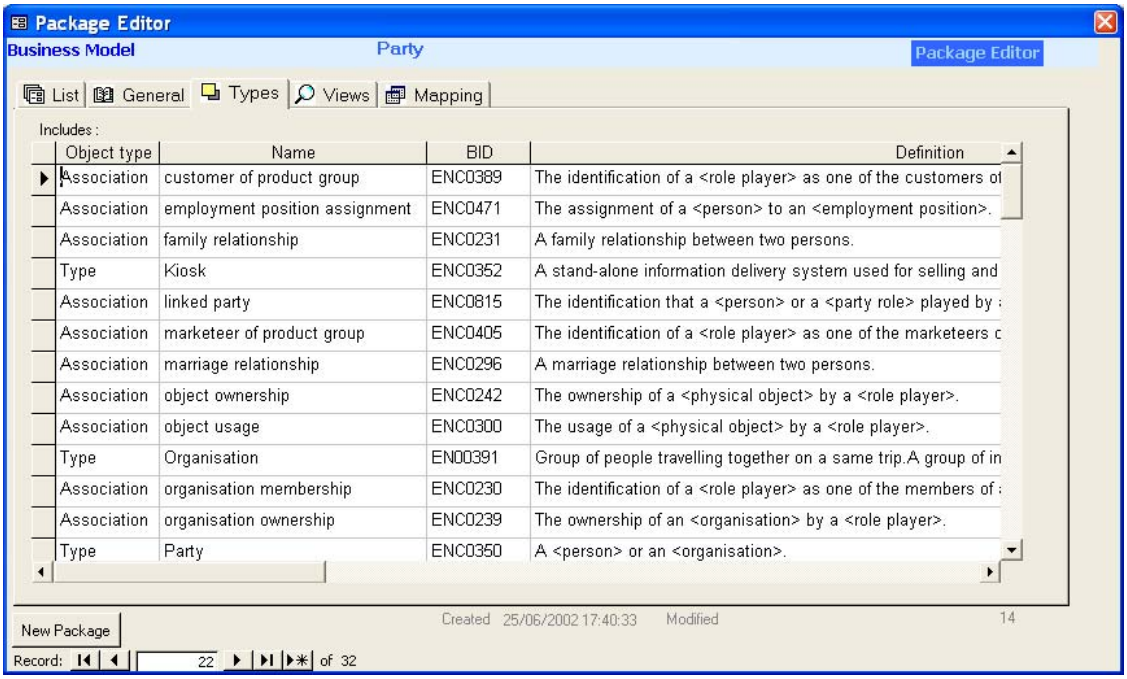
Parent
This drop-down list is used to build the Package Hierarchy.

Business Identifier (BID)
A Package's BID can be created and maintained using the BID wizard function, using either a supplied prefix or none. In addition, this field can be left blank and allocated at another time via a batch job allocating all BIDs for a given model at one time (refer to the Model Management chapter for more information).

Package Editor: Types Tab

The **Types Tab** of the **Package Editor** is a READ ONLY screen that lists all Types that belong to a given Package.

Allocating a Type to a certain Package is done via the Type Editor.



Package Editor: Views Tab

The **Views Tab** of the **Package Editor** is a READ ONLY screen that lists all Views that belong to a given Package.

Allocating a View to a certain Package is done via the View Editor.

Package Editor: Mapping Tab

A Package can be mapped to one or more Packages, from one or more models. The **Mapping Tab** captures the Source model(s) for the current Package. Once specified in the Mapping Tab, the navigation will be mapped across the appropriate models in the Hyperlinks Navigator.

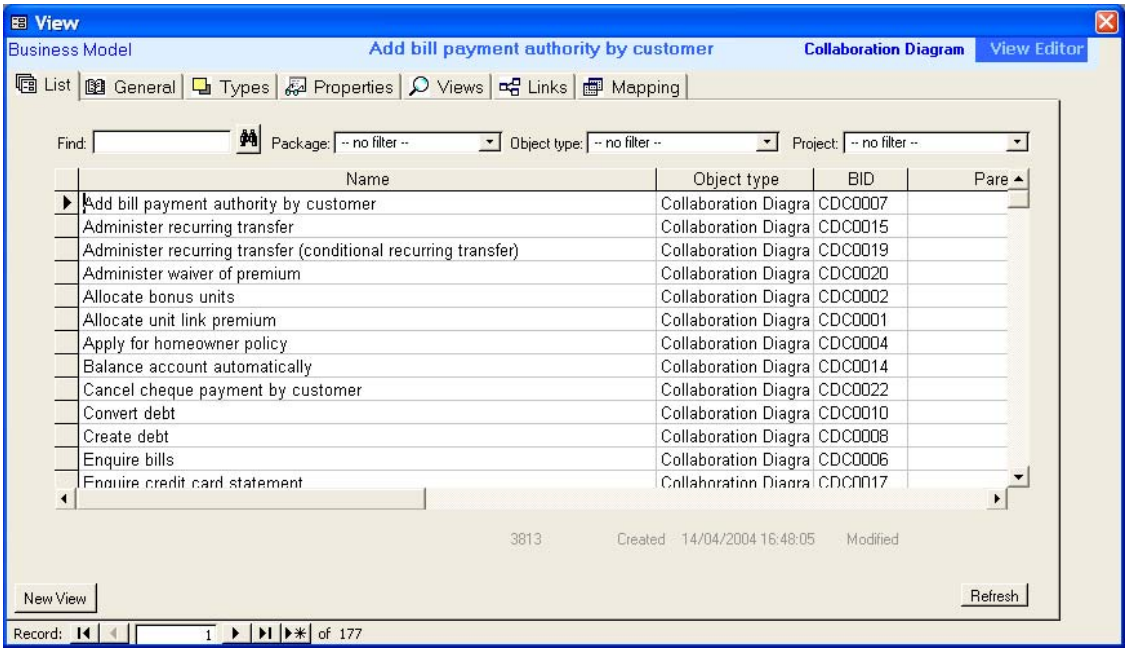
2.6 View Editor

The MMM's View Editor makes it possible to group Types, and/or Properties, together within a certain View object type. The various View object types are actually *defined* using the Object Type Editor, which will be explained further in the chapter on Model Management.
Each View object type will correspond to an entry-point in the Hyperlinks Navigator.

Note: The concept of MMM View is used to define artefacts such as Component, Diagram, Worked Example, etc, but also a particular use of an MMM View is for defining a **Project Scope**.
A *Project Scope* view makes it possible to define all Types, Type Properties and Views that belong to a certain project.
This *Project Scope* view will then be used by the Copy Model function to create a subset of the model to a new one, or to propagate the equivalent scope in a target model (See section *Hints and Tips*).

The View Editor is organised with the following tabs: **List**, **General**, **Types**, **Properties**, **Views** and **Mapping Tabs**, as well as the optionally displayed **Links Tab**.

View Editor: List Tab



The **List Tab** in the **View Editor** is very similar to the **List Tab** in the **Package Editor**. It provides a list of the View Object Types for a given model, including **Name**, **Object type**, **BID**, **Parent id,** and other pertinent details.

There are two main functions for locating Views on the **View Editor's List Tab:** a **Find** function, which can be used to search for a specific value in any of the columns, and a filter on **View type** function. In addition, the columns can be sorted by right clicking on the column header, and choosing *Sort Ascending* or *Sort Descending*. This allows the user to sort the Views by Name, BID, and so on.

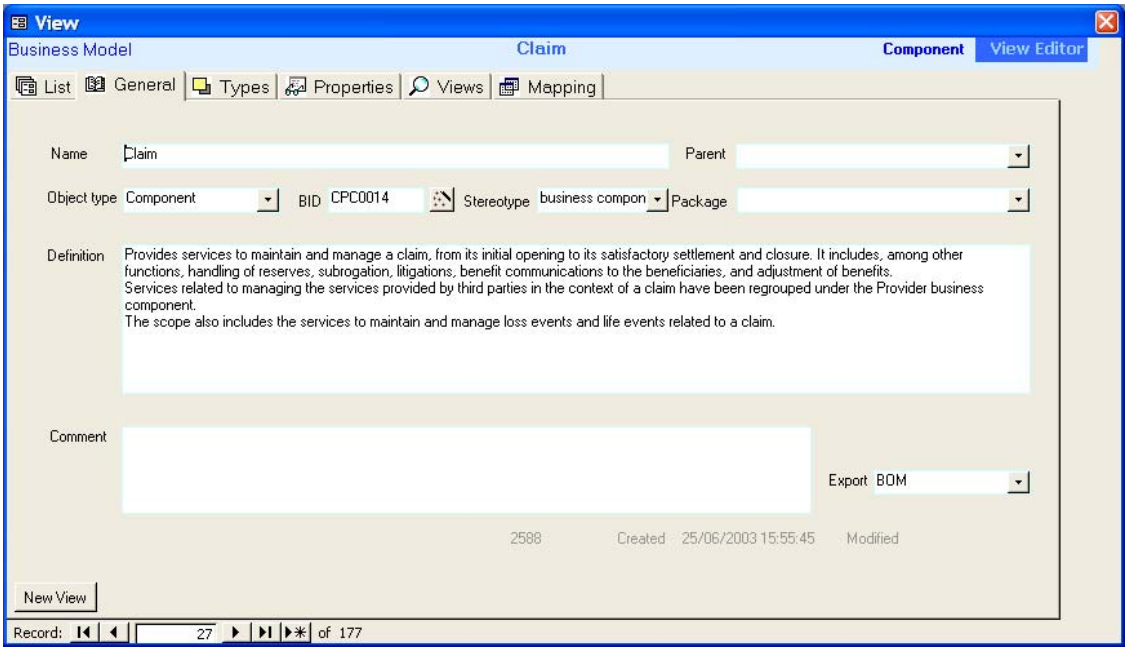
It is also from the **List** tab that you can delete a View: right-click on the row margin (where a black arrow appears in front of the selected row) and choose *Delete Record.*

Unlike deleting a Package, deleting a View does not delete the Types and Properties allocated to the View.

View Editor: General Tab

As with the Package Editor, the **General Tab** of the View Editor is where a View's details can be edited, or a new View added.

For creating a new View, click on the New View button (see below for screenshot), which will display a new record where the View details can be entered. The minimum information required is a name, and if not provided manually, it will be defaulted to the BID number.



The **General Tab** of the **View Editor** is nearly identical to the **General Tab** of the **Package Editor**. For descriptions of the **Definition**, **Parent**, and **BID** fields, refer to the explanation for the Package Editor.

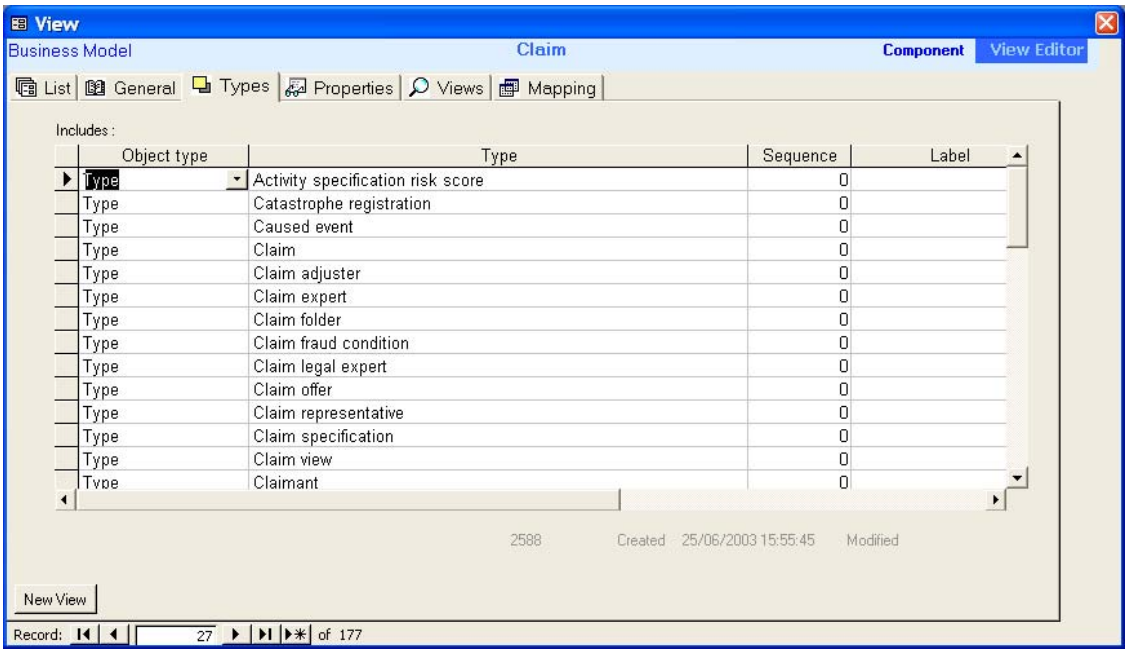
Package

A View can belong to (only) one Package. This drop-down list displays all of the packages defined in the current model. When creating a new View, use this field to select its package.

View Editor: Types / Properties Tabs

The **Types Tab** can be used to define which elements are parts of the View. Allocating a Type or an Association as part of a View implicitly means that all Properties are part of the View as well.

Specifying a property in **Properties Tab** means that this specific Property belongs to the View (in opposition to all Properties of a Type), regardless of whether this Type is specified in the Type Tab or not.



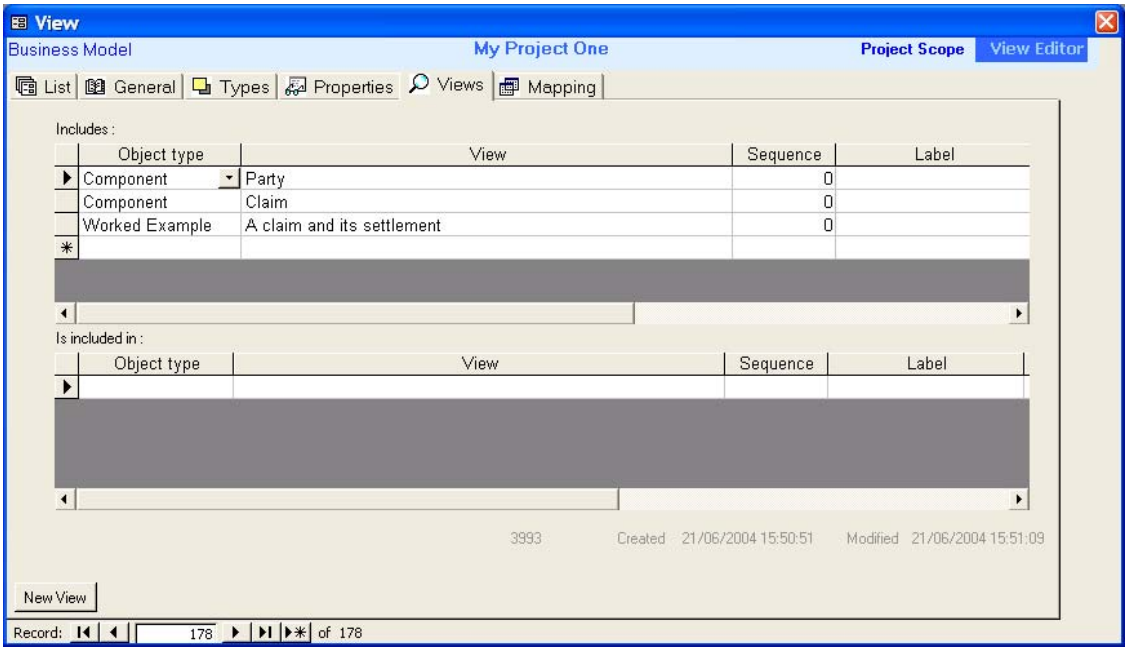
There are two fields common to all of these Tabs, the Sequence field, and the Label field.

Label
This field allows the user to specify a contextual name for an element in the View.

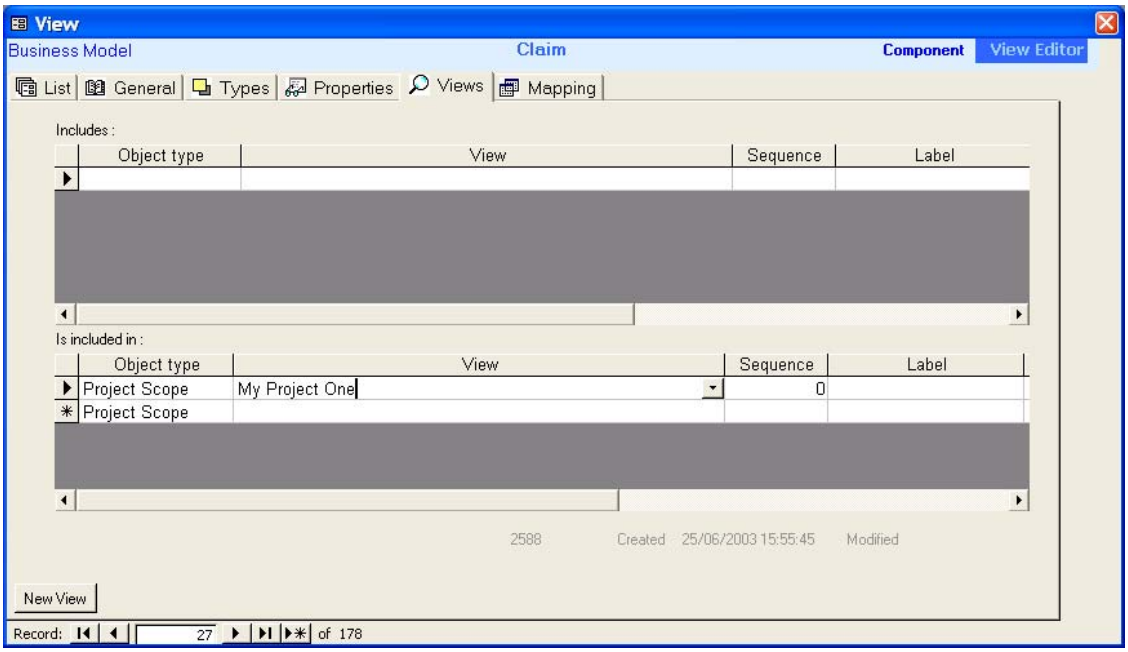
Sequence
This field is used to force the sequence in the Hyperlinks Navigator.

View Editor – Views Tab

Similarly to the previous tabs, the **Views Tab** allows to specify the views that the current View **includes**:

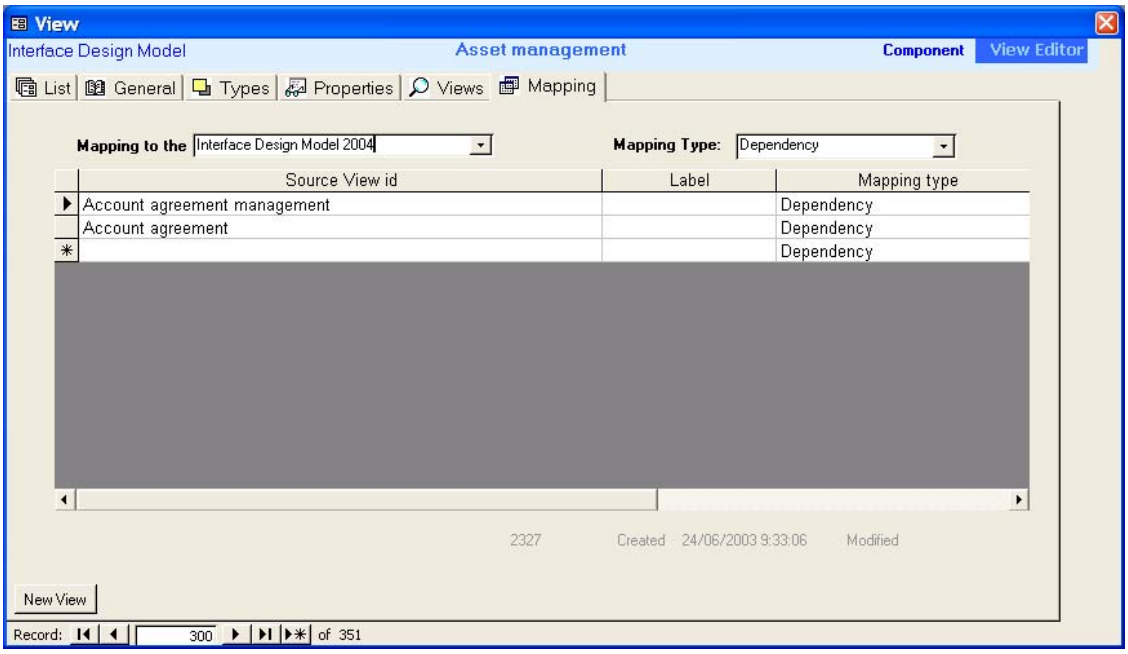


It is also possible from this tab to specify if the current View **is included in** another one:



View Editor: Mapping Tab

A View can be mapped to one or more Views, from one or more models. The **Mapping Tab** captures the Source model(s) for the current View. Once specified in the Mapping Tab, the navigation will be mapped across the appropriate models in the Hyperlinks Navigator.



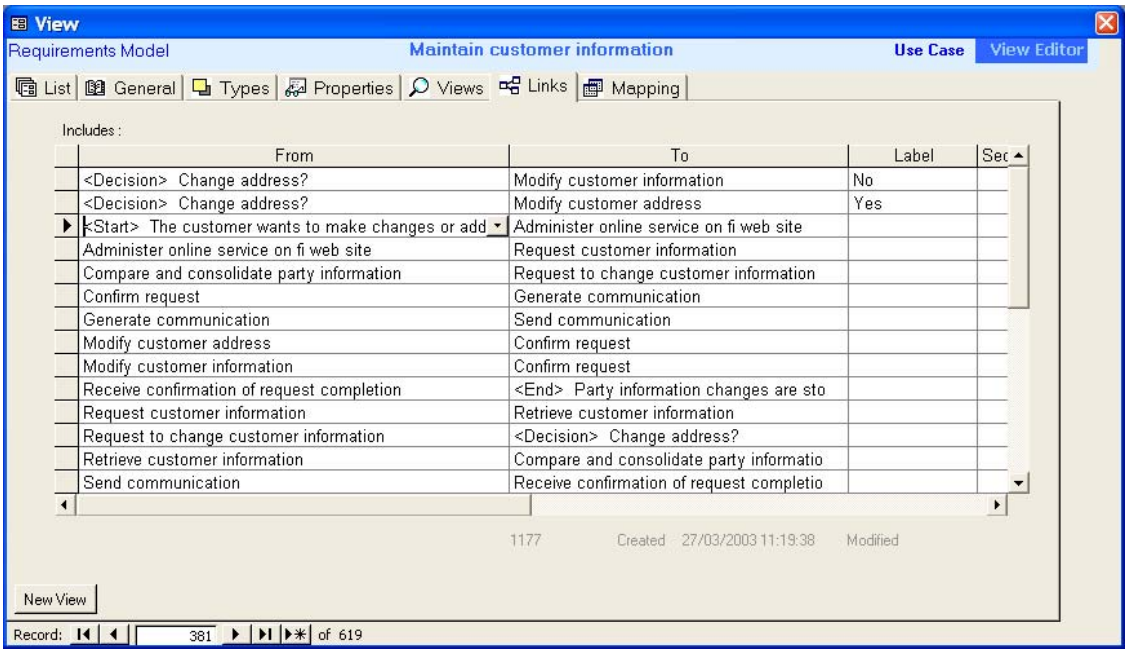
For a list of **Mapping types**, refer to the explanation provided in the **Type Editor: Mapping Tab** section. In addition to the Mapping types defined in that section, for the case of the IAA Interface Design Model (IDM), there is a "Dependency" Mapping Type used intra-model to define the Component Dependency.

Note that you can create your own mapping type by simply entering it in the field. Next time it will be part of the pull-down list.

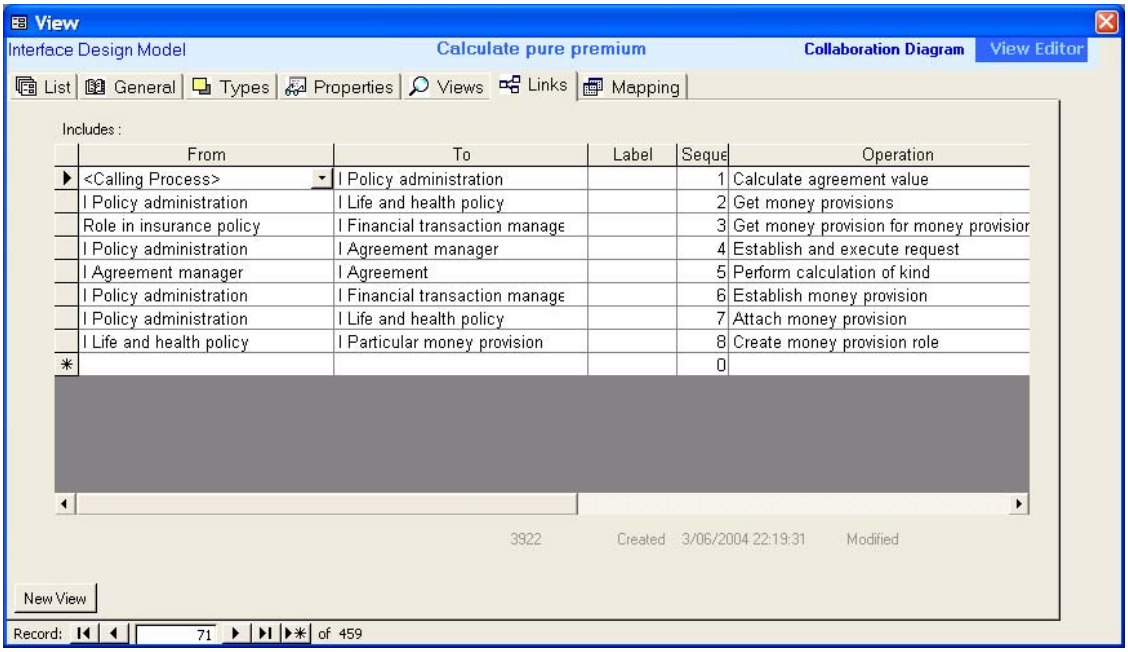
View Editor: Links Tab

When a View is of behaviour ***Diagram*** (see Object Type Editor), a View has an extra **Links** tab that represents how the view elements are linked to each other.

Here is an example of an Activity Diagram (Requirements Model) :

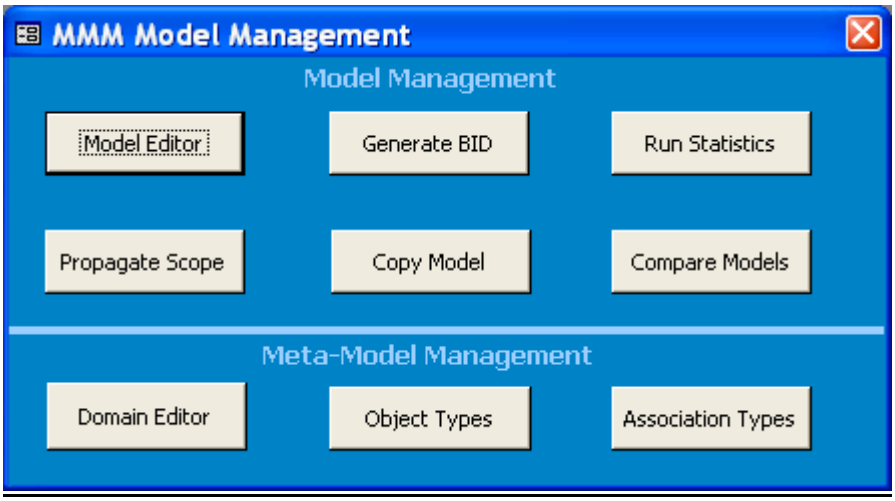


Here is an example of a Collaboration Diagram (Interface Design Model) :



CHAPTER 3: MODEL MANAGEMENT

The **Model Management** button from the main MMM Menu opens a Sub Menu:

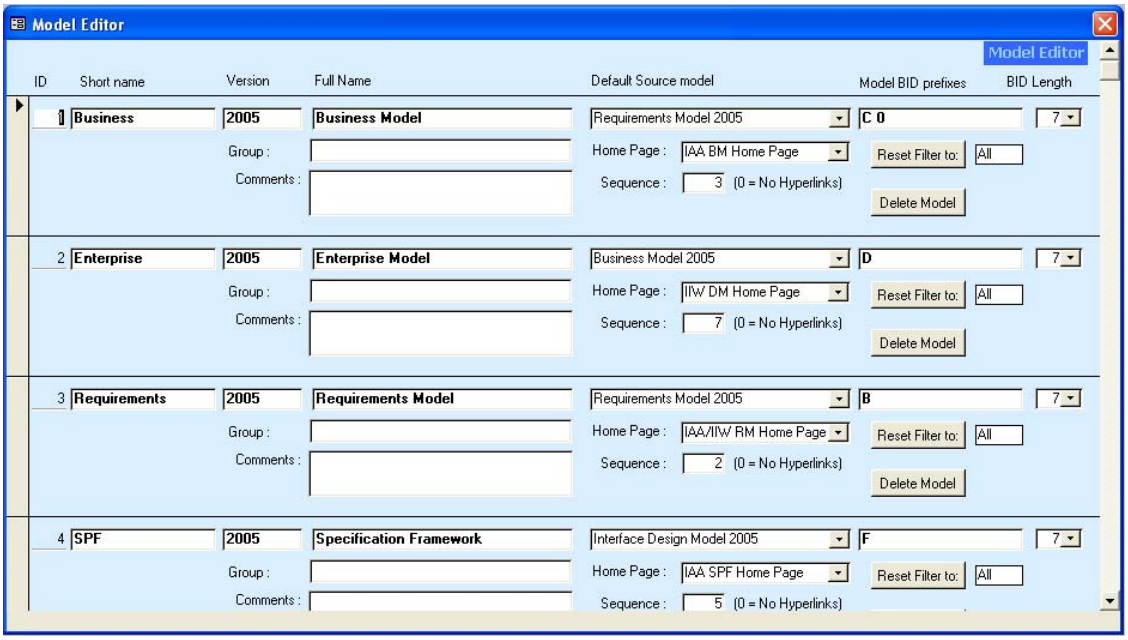


The menu is composed of two portions:

- The **Model Management,** which is related to model-specific customisation and manipulations,
- The **Meta-Model Management**, which is related to global tailoring of MMM behaviours applicable to all models.

3.1 Model Editor

The Model Editor lists all of the models stored in the MMM.
Note: A specific model can be in the MMM multiple times, in several versions (e.g. IDM2003, IDM2005, AcmeIDM…).



Short Name
This field uniquely identifies a model.

Full Name and **Version**
These fields are used on reports, and HTML pages.

Default Source model
This field is used to designate which model will be used as the default value in the Mapping tabs of the Type and Property Editors.
Note: The most often used source model is given here; however, other source models can be specified when creating a mapping).

Model BID prefixes
The prefixes are entered for each model with a space separating the characters.
The first prefix of the list will be the default prefix used by the BID generator, if no other prefix is specified.

The BID will be formed by XX**Y**nnnn, where:

- XX is the Object Type prefix (defined in the Object Type Editor)
- **Y is the Model prefix**
- nnnn is a meaningless sequential number

This list of prefixes can be updated for a specific project, according to the *Standards and Naming Conventions* appendix.

Group
The group makes it possible to create Hyperlinks navigation sub-menus, that regroups models together, such as all Datamarts in one sub-menu.

Sequence
The sequence makes it possible to force the sequence in which the models will appear in the Hyperlinks main navigation pane.
Setting the Sequence to 0 (zero) will hide the model in the Hyperlinks.

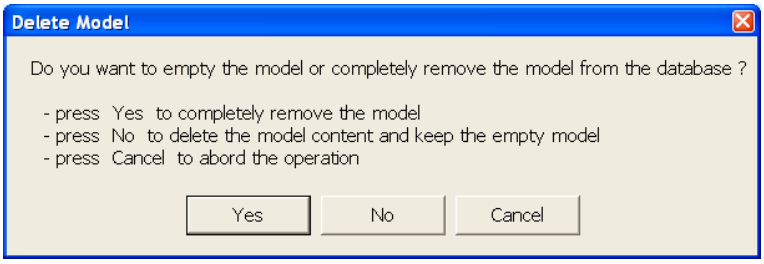
Home Page
This is a pull-down list of all Forms defined in the MS-Access database. The Home Page button of the main Menu will open the Form specified here.
This makes it possible to customise a simplified entry-point to that specific model.

The 

function makes it possible to re-initialise the Export filter in the Package, View, Type and Type property tables with a certain value. (Refer to the Hints and Tips appendix for more information about the Export Filter)

The 

function makes it possible to delete a model.

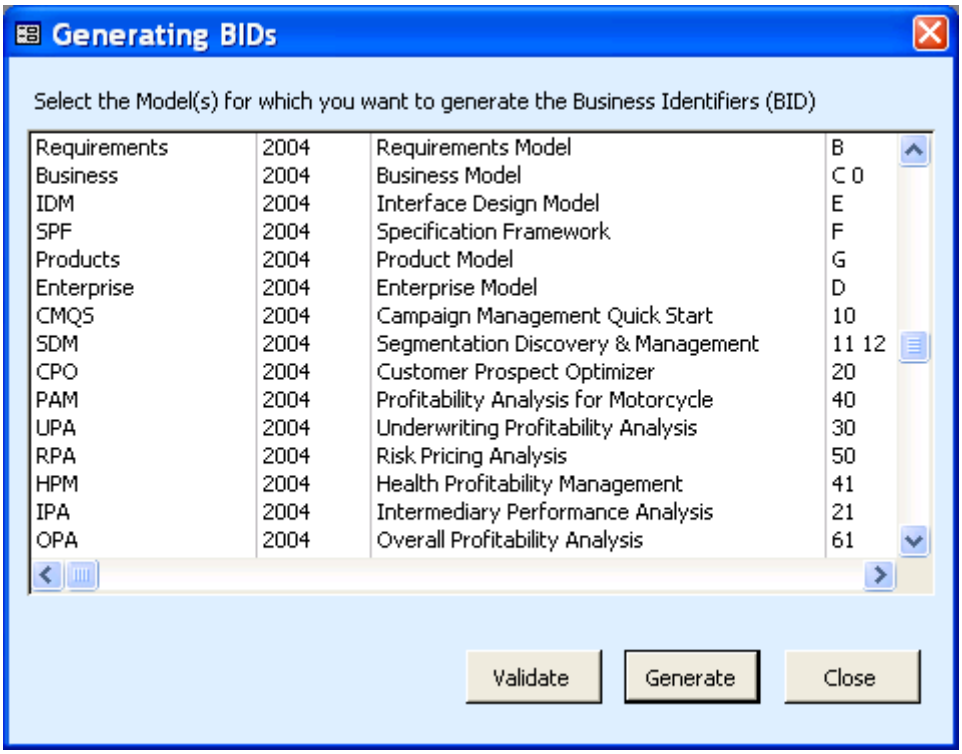


Warning: By deleting a model, its entire contents (packages, views, types, properties, mapping..) will be deleted as well.

Due to the complex inter-dependent relationships between a model's elements, it is **strongly advised** to delete a model using the Delete button of this window, rather than deleting a model directly from the Model table by simply deleting its row.

3.2 Generate BID

The Business Identifier Generator scans all of the selected models, in order to allocate a unique BID for each Package, View, Type and Property that does not yet have a BID assigned, or has invalid one.



The **Validate** option will only check the model(s), but does not actually update them.
Part of the validation function is to look for possible BID duplicates. If a duplicate is found, the validation results will report the error.

Note: If duplicates are found, the redundancy will have to be corrected in the Type Editor prior to continue.
A report that lists duplicates is available from the Report menu (see Chapter 5).

The **Generate** option allocates new BIDs and actually updates the model(s).
If a valid prefix has not been supplied within the Type Editor, the default prefix (the first one in the list) is used.
The BID will be formed by XXYnnnn, where:

- XX is the Object Type prefix (defined in the Object Type Editor)
- Y is the Model prefix (defined in the Model Editor)
- nnnn is a meaningless sequential number

BID will not be generated if the Object Type prefix (provided in the Object Type Editor) is followed by an asterisk (e.g. XX*). This allows preserving pre-allocated BIDs by an external tool.

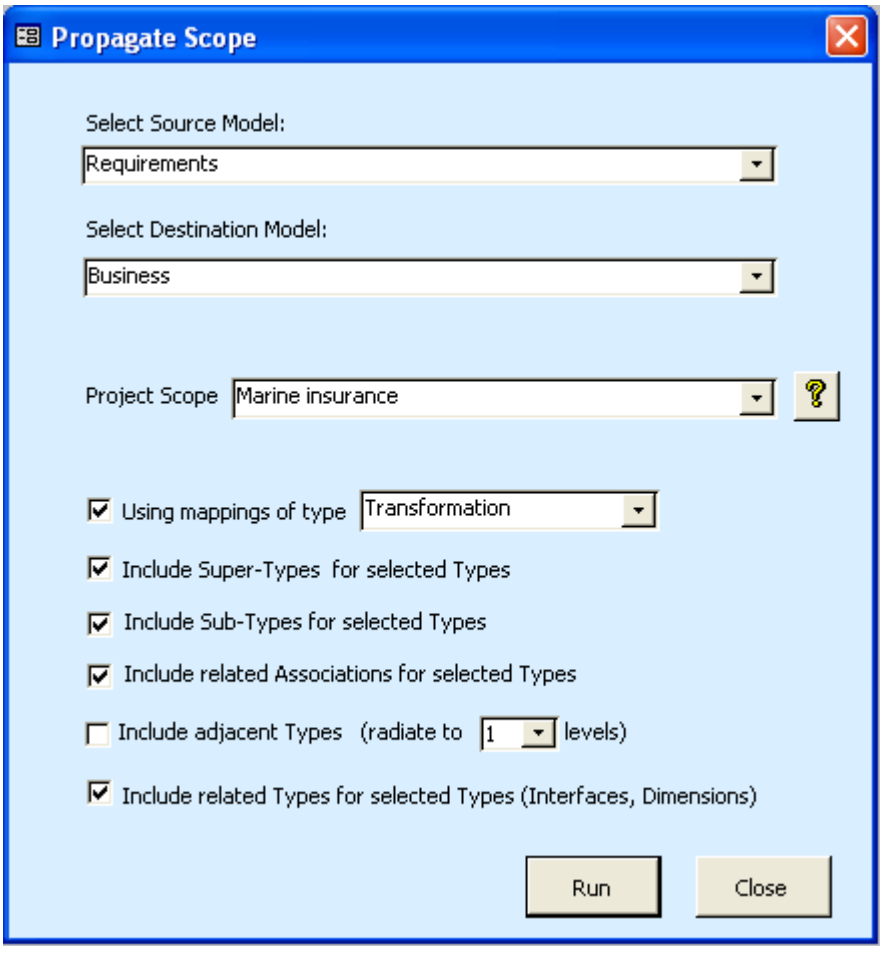
If the BID allocation reaches the maximum number (e.g. ATD9999) the generator will try to find wholes in numbering and allocate these numbers (reusing previously deleted BID). The Comment field of all involved instances in that reuse process will be updated with a message recording this behaviour.

3.3 Propagate Scope

The Propagate Scope function makes it possible to create a *Project Scope* view in a Destination model according to the **mapping** provided between the Source and the Destination models.

This function can be used for example to:

- Create a Project Scope view in the Business Model based on a Project Scope view created at the Requirements Model level
- Create a Project Scope view in the Enterprise Model based on a Project Scope view created at the Business Model level
- Create a Project Scope view in the Business Model based on a Project Scope view created at the Enterprise Model level (reverse engineering - impact analysis)



Using mappings of type …
Depending of the situation, the user may want to provide a different mapping type between the Source model and the Destination model. This mapping type is used to filter the mappings that are traced to propagate the scope to the Destination model.

Include Super-types for selected Types
The user can decide to include or not the Super-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Organisation*, *Party*, *Role player* and *Business model object* will be traced in the subset as well (as being super-types).

Include Sub-Types for selected Types
The user can decide to include or not the Sub-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Branch, Department, Employment position, Regional unit,* and *Team* will be traced in the subset as well (as being sub-types).

Include Associations related to Types in scope
The user can decide to include or not the Associations that have the two parent Types included in the Project Scope.

Include Adjacent Types
The user can decide to include all Types that are next to the in-scope Types. The broadness of the inclusion can be set by the number of levels of radiation. (Can be for example used in the Product Model to select one Product and all its specifications and sub products).

Include related Types for selected Types
The user can decide to include or not the Interfaces (typically, target model is IDM) or the Dimensions of a Fact table (typically, target model is EM) for the Types that are included in the Project Scope (based on "*realizes*", "*best implements*", or "*has for dimension*" relationships).
For example, in the Business Model, by having included *Organisation Unit* in the scope, *I Organisation unit* in the Interface Design Model will be in the scope as well (as being the interface that *Organisation Unit* realizes).

The  function provides a full Resulting Scope Log that explains the "why" for the presence of each instance in the scope.

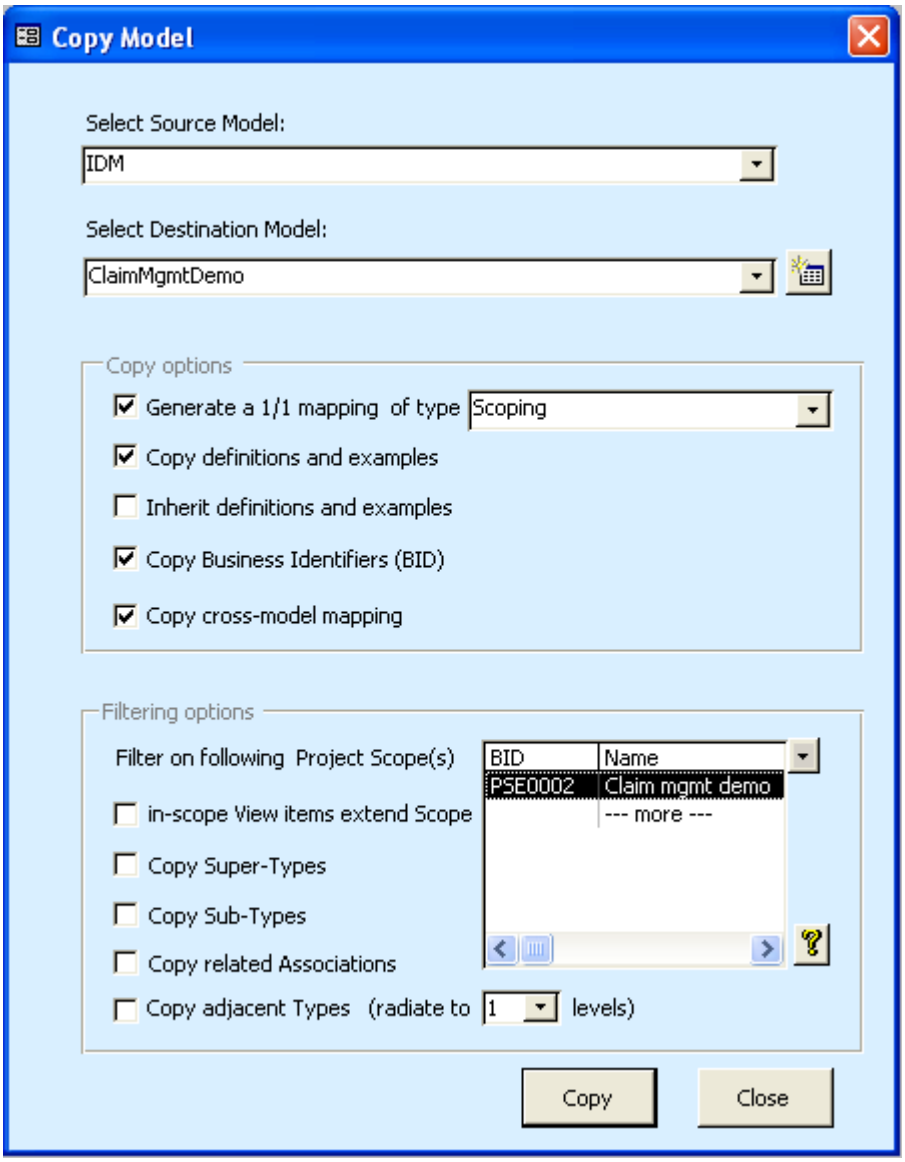
Please refer to the ***Hints and Tips*** appendix for more explanation on how to use the *Scope Propagator* in a project.

3.4 Copy Model

The Copy Model function makes it possible to *clone* a model, or to create a subset of it. This function can be used for example to:

- Create a new Edition/Version of a model
- Create a Design model starting from a Business Model
- Create a Data mart starting from the Enterprise model
- Create a subset of a model for scoping a Project
- More generally speaking, create a new model starting from a similar one.

The filter on a **Project Scope** view can be used to create a subset of a model.
(See section *Hints and Tips* for how to define a Project Scope view)



Please refer to the ***Hints and Tips*** appendix for more explanation on how to use the *Copy Mode*l in a project.

Copy options

Generate a 1/1 mapping of type …
Depending of the situation, the user may want to maintain a mapping between the Source model and the newly created Destination model. This 1/1 mapping is generated at both Type and Property levels.

Copy definitions and examples or **Inherit Copy definitions and examples**
It is also possible to actually replicate the definitions to the Destination model (this will be the definition of the Source model, and its inherited definitions), or to only set the *Append Definition* and *Append Examples* flags in the mapping records.

Copy Business Identifiers (BID)
In the case of creating a new Version of a model, the user will most likely want to copy the Business Identifiers as well, whereas in the case of creating a data mart, the BIDs would be left blank, and then generated afterwards using the BID Generator.

Copy cross-model mapping
The user can decide to include or not the Mappings that exist to the other models (or intra-model).

Filtering options

When **filtering on Project Scope(s)**, the following options can be used to extend the scope as defined in the Project Scope View(s):

in-scope View items extend Scope
The user can decide whether the views that are defined in the Project Scope view can themselves be used to define the Types and Properties that belong to the scope.
If the option is selected, the view items extend the scope, while if the option is not selected, the views in the target model will only contain Types and Properties that belong to the scope.

For example: in the source model, we have Types T1, T2, T3, Views V1 (that includes T1 T3), V2 (that includes T2 T3), and the project scope PS1 (that includes T1 V1)

- With the option **not selected**: the Destination model will have T1 and V1 (that includes only T1, as T3 is out of scope).
- With the option **selected**: the Destination model will have T1, T3 (thanks to V1 items) and V1 (that includes T1 T3).

Copy Super-types
The user can decide to copy or not the Super-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Organisation*, *Party*, *Role player* and *Business model object* will be copied in the subset as well (as being super-types).

Copy Sub-Types
The user can decide to copy or not the Sub-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Branch, Department, Employment position, Regional unit,* and *Team* will be copied in the subset as well (as being sub-types).

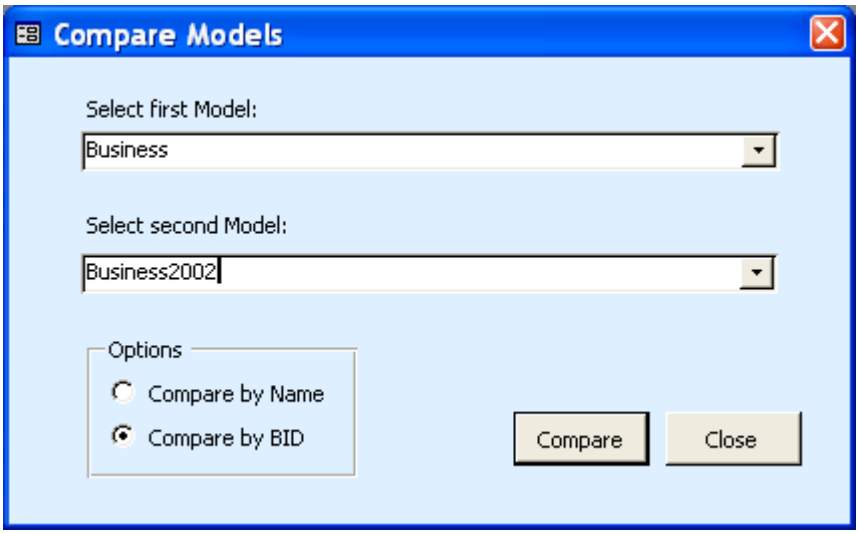
Copy related Associations
The user can decide to copy or not the Associations that have the two parent Types included in the Project Scope.

Copy Adjacent Types
The user can decide to include all Types that are next to the in-scope Types. The broadness of the inclusion can be set by the number of levels of radiation.

3.5 Compare Models

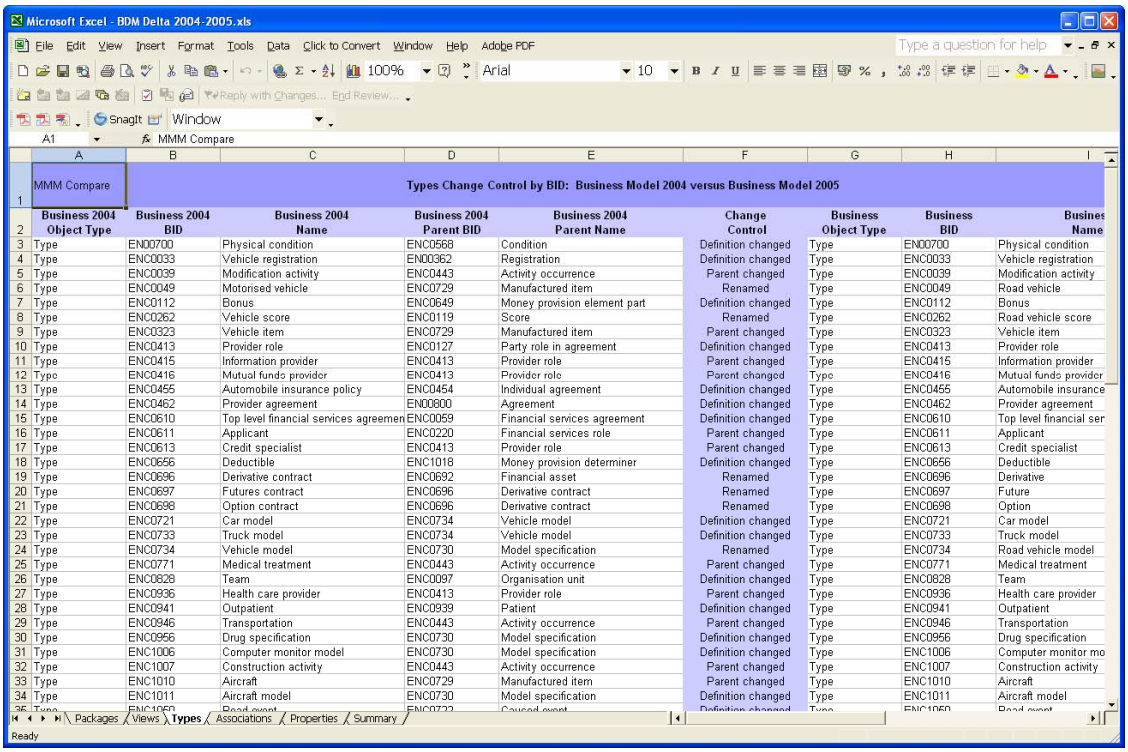
The Compare function provides a quick way to perform the following analysis:

- Compare versions (for example IDM2004 versus IDM2005)
- Perform a gap analysis (for example Acme_Business2002 versus IAA_Business2005)
- Detect transformations (for example Business model versus Enterprise model)



The Compare Models function produces an MS-Excel spreadsheet with the following tabs:

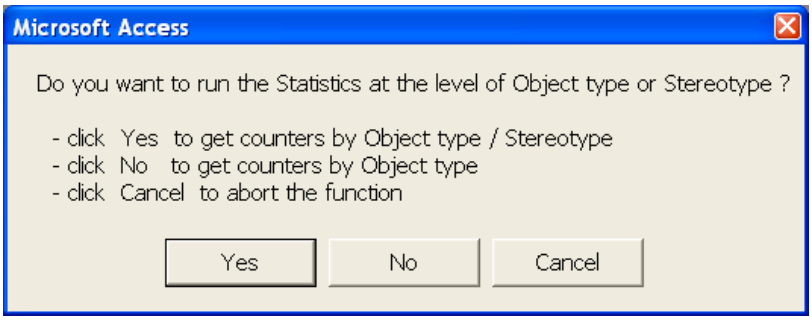
- Packages
- Views
- Types
- Associations
- Properties
- Summary



	Business 2004 Object Type	Business 2004 BID	Business 2004 Name	Business 2004 Parent BID	Business 2004 Parent Name	Change Control	Business Object Type	Business BID	Busines Name
3	Type	EN00700	Physical condition	ENC0568	Condition	Definition changed	Type	EN00700	Physical condition
4	Type	ENC0033	Vehicle registration	EN00362	Registration	Definition changed	Type	ENC0033	Vehicle registration
5	Type	ENC0039	Modification activity	ENC0443	Activity occurrence	Parent changed	Type	ENC0039	Modification activity
6	Type	ENC0049	Motorised vehicle	ENC0729	Manufactured item	Renamed	Type	ENC0049	Road vehicle
7	Type	ENC0112	Bonus	ENC0649	Money provision element part	Definition changed	Type	ENC0112	Bonus
8	Type	ENC0262	Vehicle score	ENC0119	Score	Renamed	Type	ENC0262	Road vehicle score
9	Type	ENC0323	Vehicle item	ENC0729	Manufactured item	Parent changed	Type	ENC0323	Vehicle item
10	Type	ENC0413	Provider role	ENC0127	Party role in agreement	Definition changed	Type	ENC0413	Provider role
11	Type	ENC0415	Information provider	ENC0413	Provider role	Parent changed	Type	ENC0415	Information provider
12	Type	ENC0416	Mutual funds provider	ENC0413	Provider role	Parent changed	Type	ENC0416	Mutual funds provider
13	Type	ENC0455	Automobile insurance policy	ENC0454	Individual agreement	Definition changed	Type	ENC0455	Automobile insurance
14	Type	ENC0462	Provider agreement	EN00800	Agreement	Definition changed	Type	ENC0462	Provider agreement
15	Type	ENC0610	Top level financial services agreemen	ENC0059	Financial services agreement	Definition changed	Type	ENC0610	Top level financial ser
16	Type	ENC0611	Applicant	ENC0220	Financial services role	Parent changed	Type	ENC0611	Applicant
17	Type	ENC0613	Credit specialist	ENC0413	Provider role	Parent changed	Type	ENC0613	Credit specialist
18	Type	ENC0656	Deductible	ENC1018	Money provision determiner	Definition changed	Type	ENC0656	Deductible
19	Type	ENC0696	Derivative contract	ENC0692	Financial asset	Renamed	Type	ENC0696	Derivative
20	Type	ENC0697	Futures contract	ENC0696	Derivative contract	Renamed	Type	ENC0697	Future
21	Type	ENC0698	Option contract	ENC0696	Derivative contract	Renamed	Type	ENC0698	Option
22	Type	ENC0721	Car model	ENC0734	Vehicle model	Definition changed	Type	ENC0721	Car model
23	Type	ENC0733	Truck model	ENC0734	Vehicle model	Definition changed	Type	ENC0733	Truck model
24	Type	ENC0734	Vehicle model	ENC0730	Model specification	Renamed	Type	ENC0734	Road vehicle model
25	Type	ENC0771	Medical treatment	ENC0443	Activity occurrence	Parent changed	Type	ENC0771	Medical treatment
26	Type	ENC0828	Team	ENC0097	Organisation unit	Definition changed	Type	ENC0828	Team
27	Type	ENC0936	Health care provider	ENC0413	Provider role	Parent changed	Type	ENC0936	Health care provider
28	Type	ENC0941	Outpatient	ENC0939	Patient	Definition changed	Type	ENC0941	Outpatient
29	Type	ENC0946	Transportation	ENC0443	Activity occurrence	Parent changed	Type	ENC0946	Transportation
30	Type	ENC0956	Drug specification	ENC0730	Model specification	Definition changed	Type	ENC0956	Drug specification
31	Type	ENC1006	Computer monitor model	ENC0730	Model specification	Definition changed	Type	ENC1006	Computer monitor mo
32	Type	ENC1007	Construction activity	ENC0443	Activity occurrence	Parent changed	Type	ENC1007	Construction activity
33	Type	ENC1010	Aircraft	ENC0729	Manufactured item	Parent changed	Type	ENC1010	Aircraft
34	Type	ENC1011	Aircraft model	ENC0730	Model specification	Definition changed	Type	ENC1011	Aircraft model

3.6 Run Statistics

The run Statistics function counts the number of instances per Model, detailed by Object type and, optionally detailed per Stereotype as well.



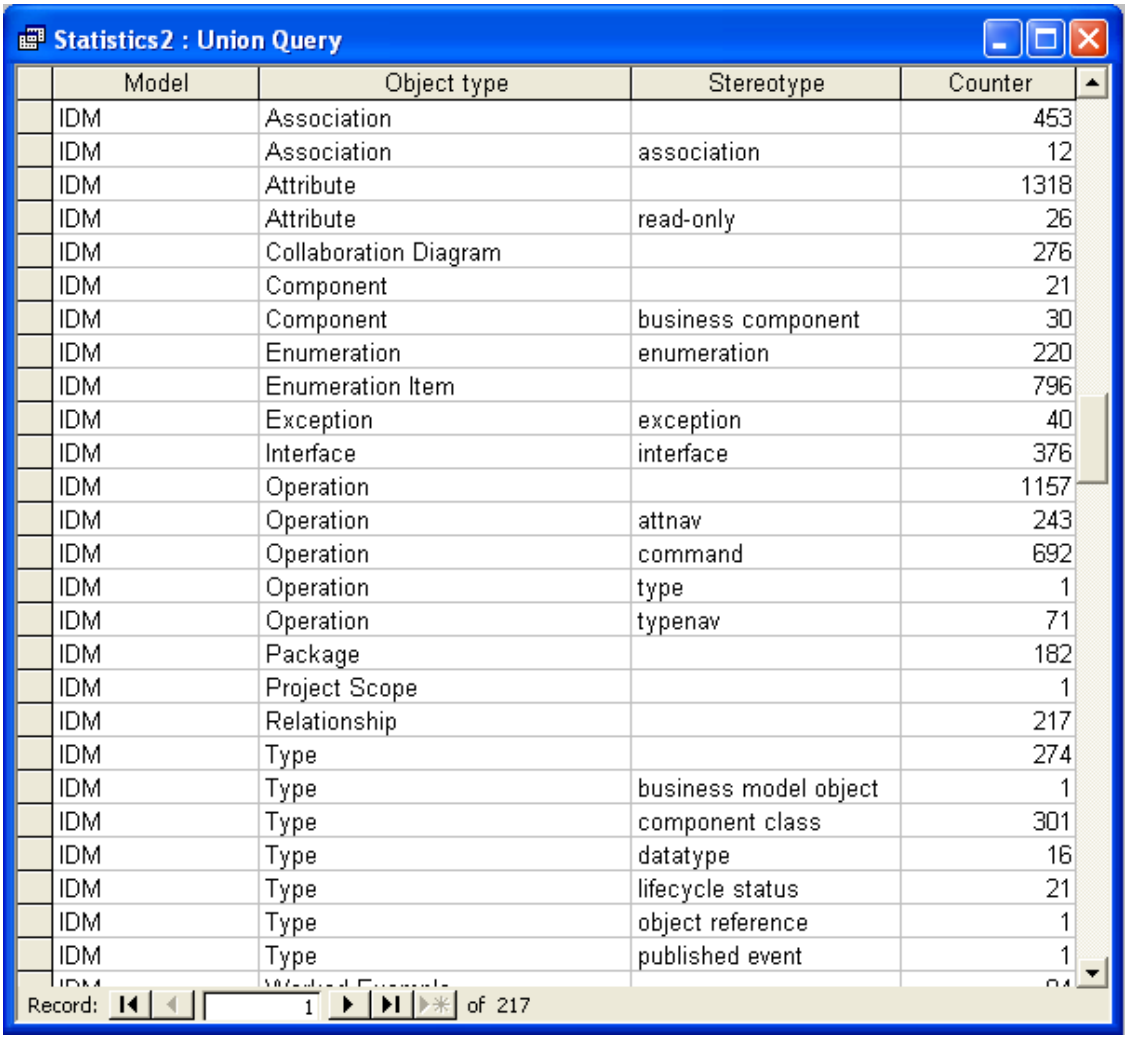
Statistics per Model / Object type:

Statistics1 : Union Query

Model	Object type	Counter
Business	Association	526
Business	Attribute	961
Business	Collaboration Diagram	22
Business	Component	31
Business	Component Service	255
Business	E/R Diagram	35
Business	Interface	20
Business	Package	26
Business	Project Scope	4
Business	State	252
Business	State Diagram	29
Business	State Machine	29
Business	Type	479
Business	Worked Example	73
CEA	Association	31
CEA	Attribute	561
CEA	E/R Diagram	9
CEA	Package	2
CEA	Relationship	34
CEA	Type	29

Record: 1 of 154

Statistics per Model / Object type / Stereotype:



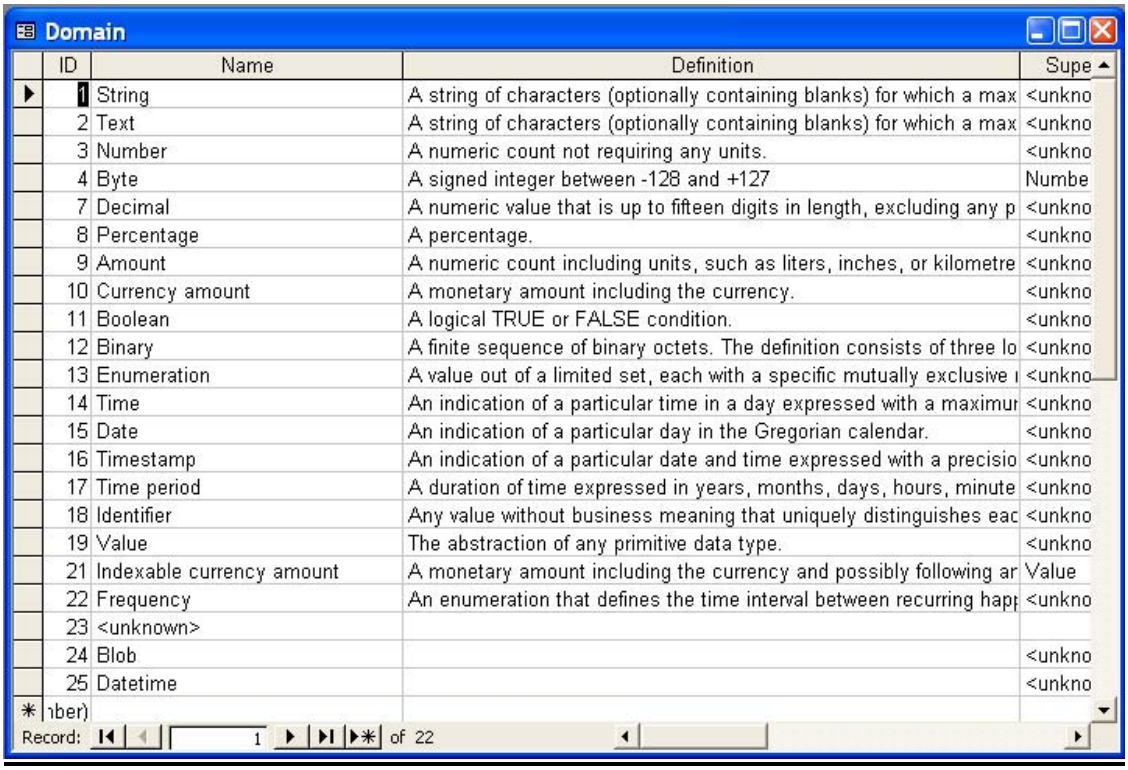
Statistics2 : Union Query

Model	Object type	Stereotype	Counter
IDM	Association		453
IDM	Association	association	12
IDM	Attribute		1318
IDM	Attribute	read-only	26
IDM	Collaboration Diagram		276
IDM	Component		21
IDM	Component	business component	30
IDM	Enumeration	enumeration	220
IDM	Enumeration Item		796
IDM	Exception	exception	40
IDM	Interface	interface	376
IDM	Operation		1157
IDM	Operation	attnav	243
IDM	Operation	command	692
IDM	Operation	type	1
IDM	Operation	typenav	71
IDM	Package		182
IDM	Project Scope		1
IDM	Relationship		217
IDM	Type		274
IDM	Type	business model object	1
IDM	Type	component class	301
IDM	Type	datatype	16
IDM	Type	lifecycle status	21
IDM	Type	object reference	1
IDM	Type	published event	1

Record: 1 of 217

3.7 Domain Editor

The Domain Editor is not tied to a particular model. All Primitive Domain Types are common to all models in MMM and can be referenced in the Property Editor.
The Interface Design Model makes an exception to this statement as it uses its own Domain Types as Classes defined in the IDM itself.



3.8 Object Type Editor

The object Type Editor makes it possible to define the meta-model constructs that will be supported by MMM, and how they will behave.

Object type

Object type	Table	Behaviour	Object BID prefix	Default Property type
Package	Package	Package	PK	
Business Process	Type	Type	BP	
Metric	Type	Type	MT	
Informational Bus Process	Type	Type	IP	
External Activity	Type	Type	EA	
Exception	Type	Type	EN	
Enumeration	Type	Type	EN	Enumeration Item
Product	Type	Type	PR	Constant Spec
Calculation Spec	Type	Type	CA	
Interface	Type	Type	EN	Operation
Business Direction	Type	Type	BD	
Business Activity	Type	Type	BA	
Atomic Subject Area	Type	Type	CS	Atomic Data Element
Association	Type	Association	EN RL	Attribute
Analytical Subject Area	Type	Type	AS	Measure Data Element
Actor	Type	Type	AC	
Constant Role Spec	Type	Type	CR	Constant Spec
State Machine	Type	Type	SM	State
Request Spec	Type	Type	RE	Property Spec
Request Behaviour Spec	Type	Type	RB	
Relationship	Type	Association	RL	
Rule Spec	Type	Type	RU	
Role Spec	Type	Type	RO	Property Spec
System	Type	Type	SY	
System Service	Type	Type	SS	
Type	Type	Type	EN	Attribute
Product modelling associat	Type	Association	PA	
Attribute	Type property	Attribute	AT	
Atomic Data Element	Type property	Attribute	DE	
Constant Spec	Type property	Attribute	CO	
Component Service	Type property	Operation	OP	
Measure Data Element	Type property	Attribute	ME	
State	Type property	Attribute	ST	
Property Spec	Type property	Attribute	PP	
Enumeration Item	Type property	Attribute	AT	
Parameter	Type property	Attribute	PM	
Operation	Type property	Operation	OP	
Use Case	View	Diagram	UC	
Product Specification Diagr	View	Diagram	PD	
Project Scope	View	View	PS	

Record: 1 of 49

The BID will be formed by **XX**Ynnnn, where:

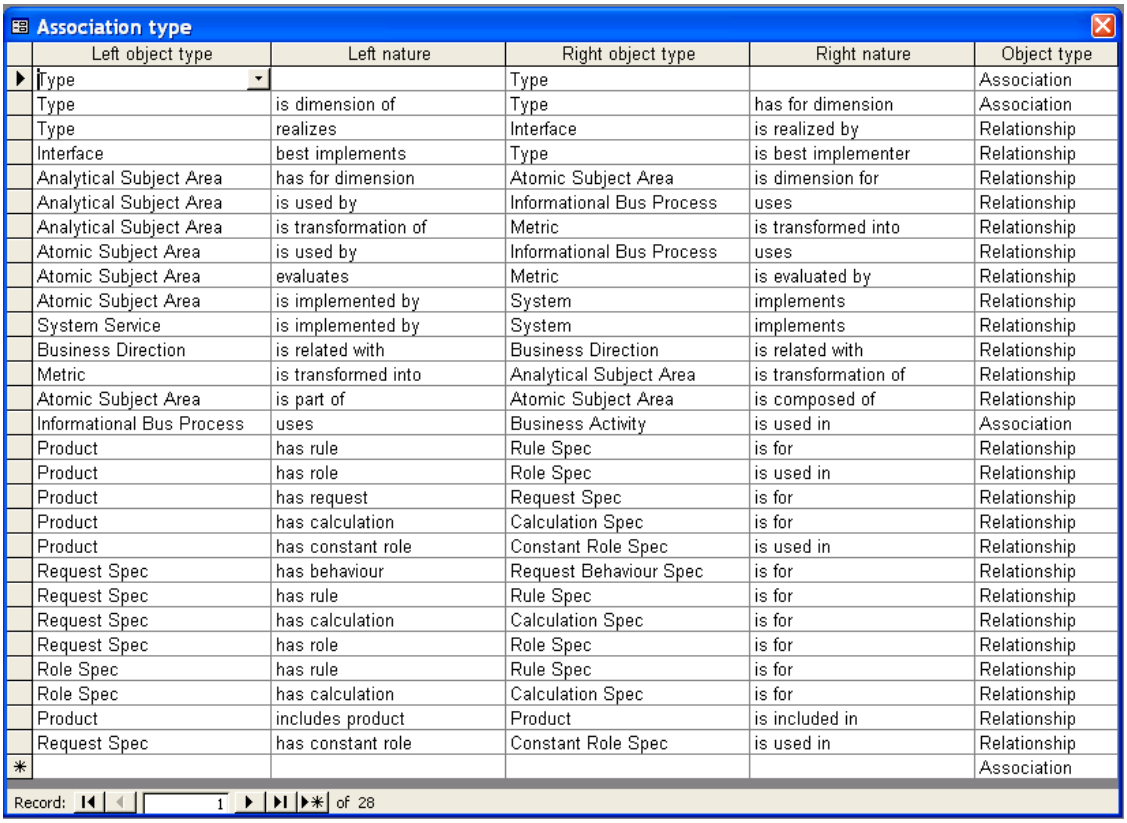
- **XX is the Object Type prefix**
- Y is the model prefix (defined in the Model Editor)
- nnnn is a meaningless sequential number

This list of prefixes can be updated for a specific project, according to the *Standards and Naming Conventions* appendix.

Note: If importing an external (non Industry Models-Based) model that brings its own BID standards, it can be loaded into the MMM with them, but the BID allocation wizards can't be used. In such a case, an asterisk must follow the prefix (e.g. XX*) in order to prevent any re-generation by MMM.

3.9 Association Type Editor

The Association Type Editor makes it possible to define the meta-model constructs that will be supported by MMM in term of pre-defined associations and relationships and their natures (role names).

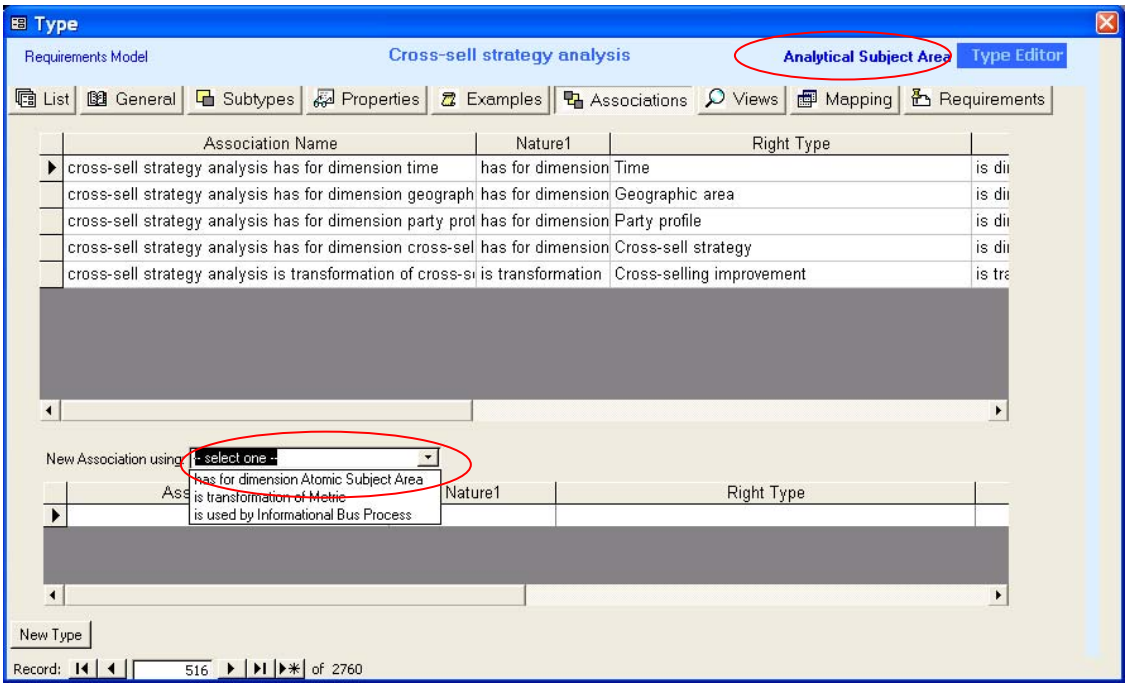


Association type

Left object type	Left nature	Right object type	Right nature	Object type
Type		Type		Association
Type	is dimension of	Type	has for dimension	Association
Type	realizes	Interface	is realized by	Relationship
Interface	best implements	Type	is best implementer	Relationship
Analytical Subject Area	has for dimension	Atomic Subject Area	is dimension for	Relationship
Analytical Subject Area	is used by	Informational Bus Process	uses	Relationship
Analytical Subject Area	is transformation of	Metric	is transformed into	Relationship
Atomic Subject Area	is used by	Informational Bus Process	uses	Relationship
Atomic Subject Area	evaluates	Metric	is evaluated by	Relationship
Atomic Subject Area	is implemented by	System	implements	Relationship
System Service	is implemented by	System	implements	Relationship
Business Direction	is related with	Business Direction	is related with	Relationship
Metric	is transformed into	Analytical Subject Area	is transformation of	Relationship
Atomic Subject Area	is part of	Atomic Subject Area	is composed of	Relationship
Informational Bus Process	uses	Business Activity	is used in	Association
Product	has rule	Rule Spec	is for	Relationship
Product	has role	Role Spec	is used in	Relationship
Product	has request	Request Spec	is for	Relationship
Product	has calculation	Calculation Spec	is for	Relationship
Product	has constant role	Constant Role Spec	is used in	Relationship
Request Spec	has behaviour	Request Behaviour Spec	is for	Relationship
Request Spec	has rule	Rule Spec	is for	Relationship
Request Spec	has calculation	Calculation Spec	is for	Relationship
Request Spec	has role	Role Spec	is for	Relationship
Role Spec	has rule	Rule Spec	is for	Relationship
Role Spec	has calculation	Calculation Spec	is for	Relationship
Product	includes product	Product	is included in	Relationship
Request Spec	has constant role	Constant Role Spec	is used in	Relationship
				Association

Record: 1 of 28

This information is then used in the **Type Editor - Associations Tab** to suggest the appropriate possible associations with their roles, and filter the target list:



CHAPTER 4: GENERATE HYPERLINKS

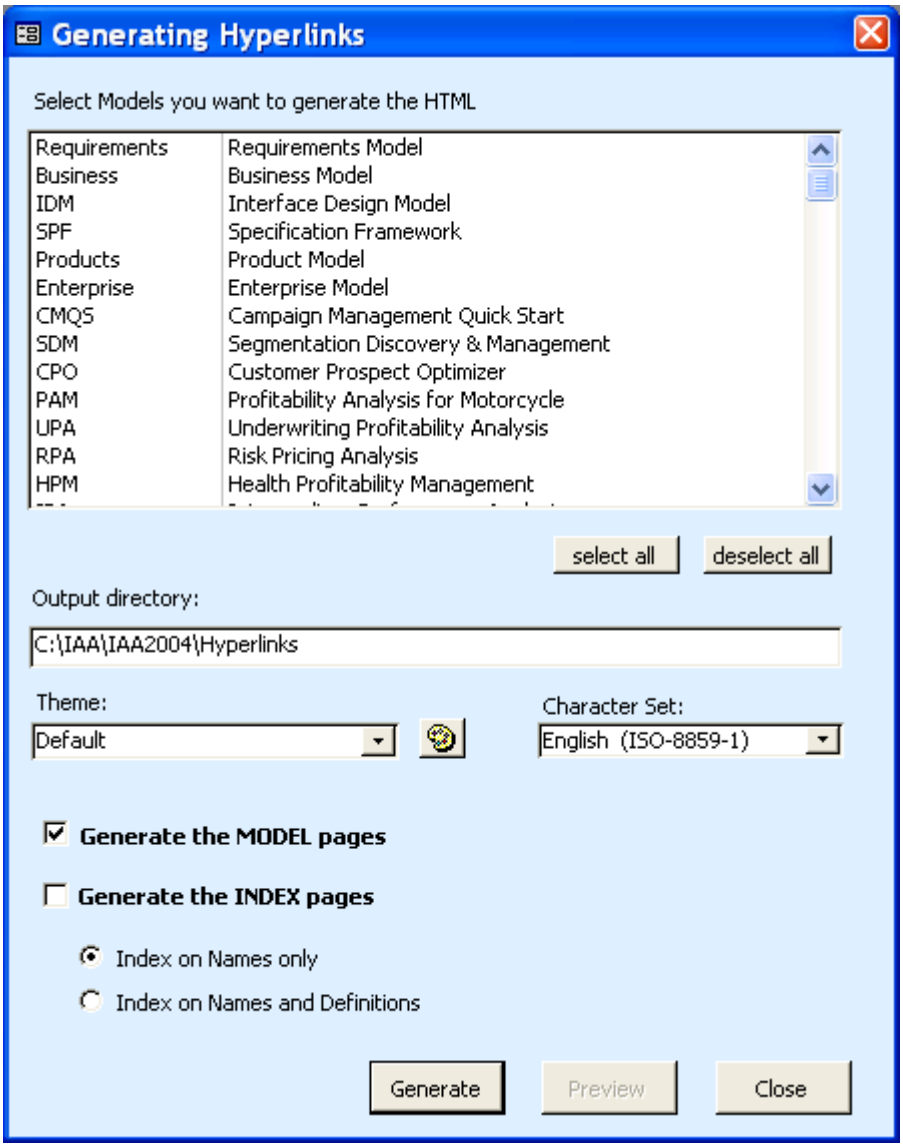
4.1 Generate Model Pages

This function generates the Hyperlinks for the selected model(s) in the specified directory:

- one page per Type, and per Property
- one page for the Alphabetic, Numeric and Hierarchy View,
- one page for the Package List
- one page per View Type
- optionally, one Main page (customised entry-point)

If, pre-existing in the model directory before starting the generation, there is a file with the BID as the file name, and one of the appropriate extensions (**.gif** or **.jpg** or **.png** or **.wmf**), then a Package or View page will imbed a link to the diagram(s).

For example: c:\Hyperlinks\Business\PartyJPG (a RSA exported diagram)
c:\Hyperlinks\Enterprise\PKD0001.GIF (an ERwin® screen capture)

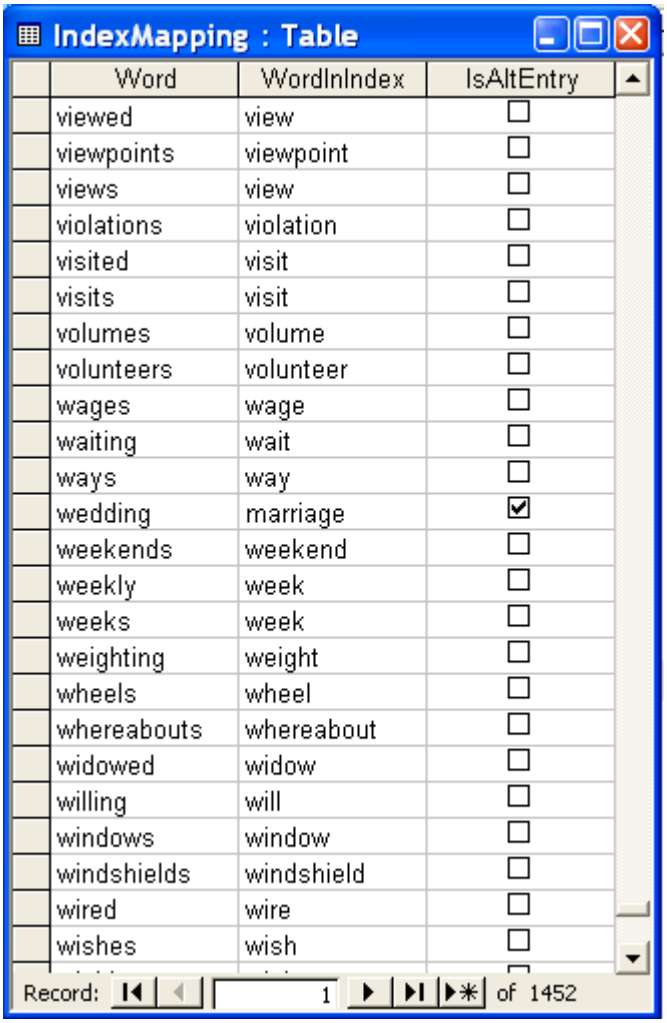


4.2 Generate Index Pages

The Generate Index Pages function scans all of the names (and optionally all of the definitions) in the selected models, and then builds an index entry for each keyword. Basically, any word is considered a keyword unless it is listed in the ***IndexNotToAppear*** table. This table has been pre-filled with common words that are not likely needed to be indexed; however the user can enrich it further.

In addition, The ***IndexMapping*** table maintains a list of common synonyms and similar words (including tense and case variations) , in an attempt to have only one unique entry for each word.

To open the Tables Window, click on  from the Main toolbar.



If the Alternate Entry (IsAltEntry) flag is selected, the word to be substituted will nevertheless appear as an entry in the index list.

For example, both *marriage* and *wedding* will appear in the Hyperlinks Index list respectively at *M* and *W* letters, but both will open *marriage* search result page with links to both words used in the models.

4.3 Defining Themes

Defining new Themes allows the user to customise sections of the Hyperlinks Navigator, such as the welcome page, background, company logo, fonts, colours, and so on.
Most of the customization can be done by editing the **style.css** file and storing it back in MMM using the Theme Editor.

To open the Theme Editor, click on the  button, and then provide a theme name.
The MMM comes with a Default theme. When using a custom theme, if a file is not supplied, the Default theme's corresponding file will be used.

Note: if no *Navigation bar* file (Navig_bar.htm) is provided, MMM will generate it with the list of all models defined in the database.

4.4 Character Set

Depending upon language-specific characters used within the MMM database content, the following Character Sets can be specified when generating HTML pages for the Hyperlinks:

Arabic (ISO-8859-6)
Catalan (ISO-8859-1)
Chinese Simplified (GB2312)
Chinese Traditional (BIG5)
Danish (ISO-8859-1)
Dutch (ISO-8859-1)
English (ISO-8859-1) **← the default value**
Esperanto (ISO-8859-3)
Finnish (ISO-8859-1)
French (ISO-8859-1)
Georgian (UTF-8)
German (ISO-8859-1)
Hebrew (ISO-8859-8-I)
Hungarian (ISO-8859-2)
Irish Gaelic (ISO-8859-1)
Italian (ISO-8859-1)
Japanese (SHIFT_JIS)
Korean (EUC-KR)
Norwegian (ISO-8859-1)
Occitan (ISO-8859-1)
Portuguese (ISO-8859-1)
Romanian (ISO-8859-2)
Russian (ISO-8859-5)
Slovenian (ISO-8859-2)
Spanish (ISO-8859-1)
Swedish (ISO-8859-1)
Yiddish (UTF-8)
UNICODE (UTF-8) **← a good alternative to support any language-specific character**

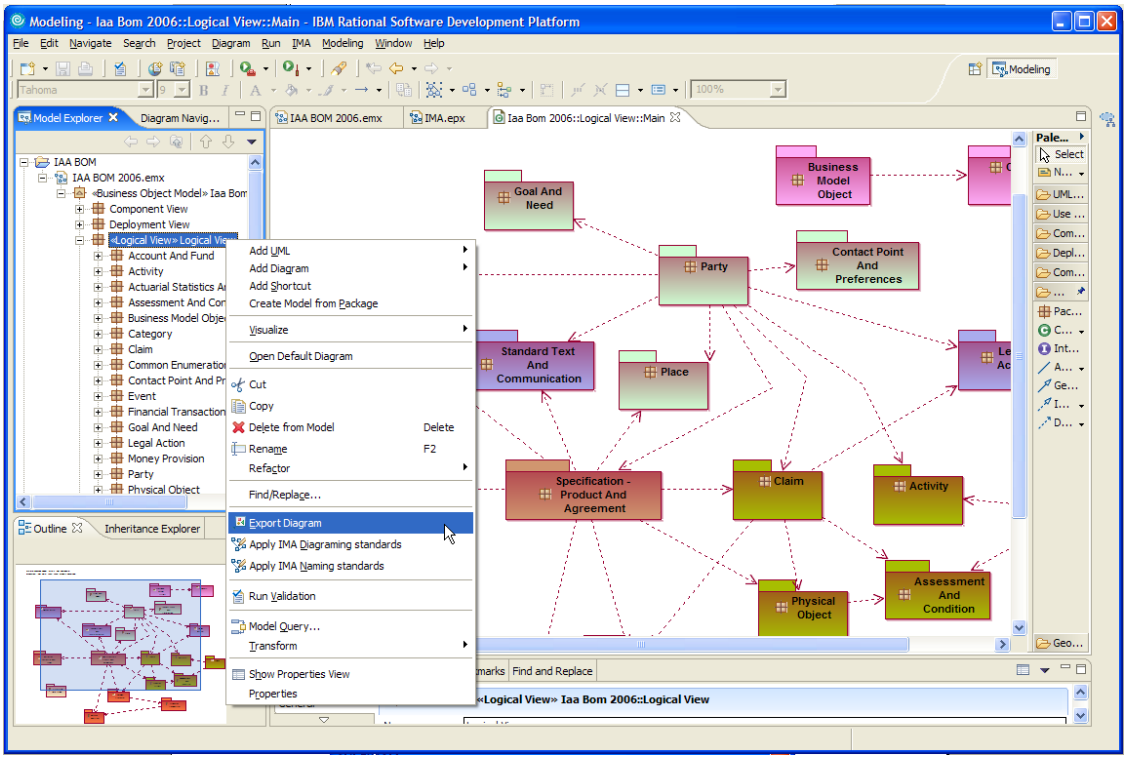
What is Unicode?
Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.

Fundamentally, computers just deal with numbers. They store letters and other characters by assigning a number for each one. Before Unicode was invented, there were hundreds of different encoding systems for assigning these numbers. No single encoding could contain enough characters: for example, the European Union alone requires several different encoding to cover all of its languages. Even for a single language like English no single encoding was adequate for all of the letters, punctuation, and technical symbols in common use.

The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others. Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646. It is supported by many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends.

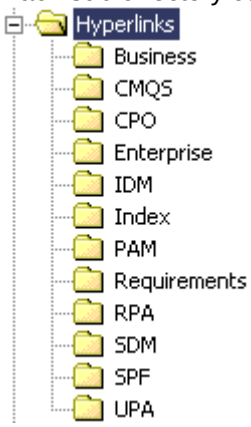
4.5 Generate diagrams

In order to extract the RSA class diagrams and place them as .jpg (or other format specified in *Preferences*) files in the Hyperlinks Navigator directories, run the **Export Diagram** function from the IMA menu or by right-click on a model, or package, or diagram.



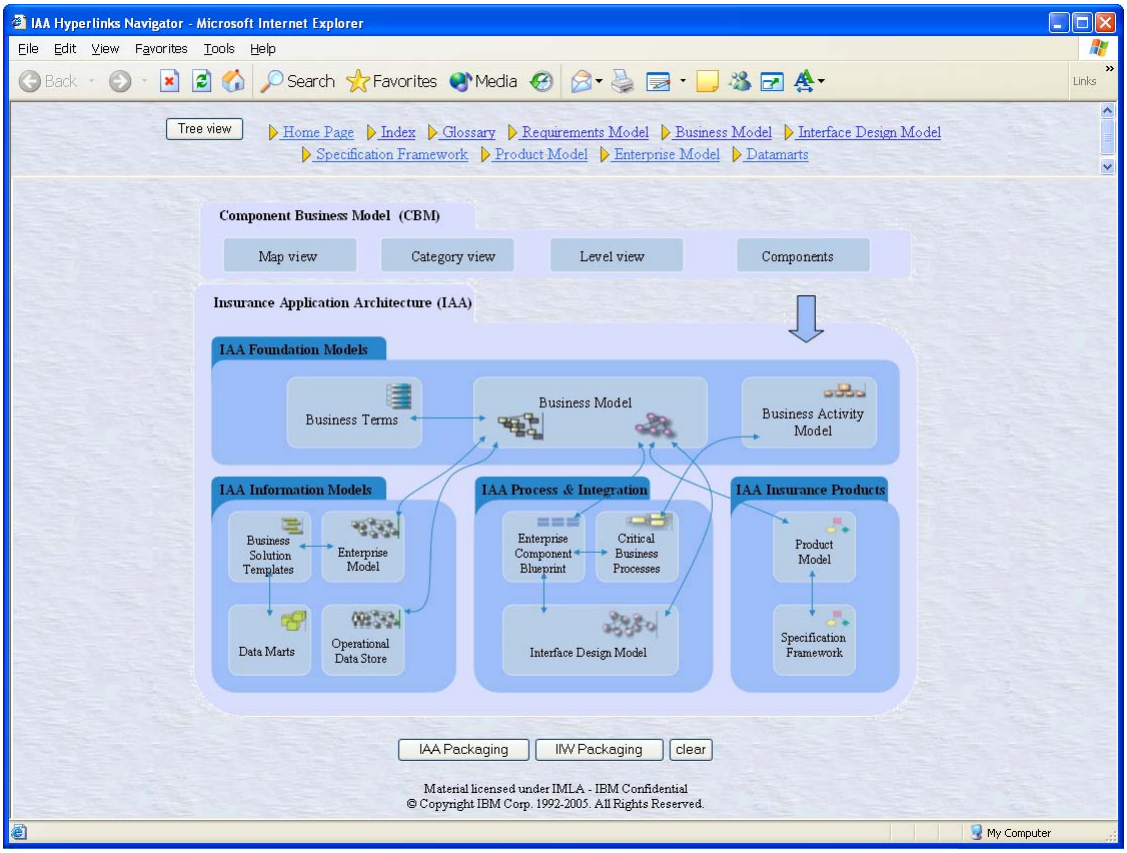
4.6 The Hyperlinks Navigator

The Hyperlinks Navigator is composed of a set of HTML files that are generated by the MMM. The files are stored in a directory structure (one sub-directory per model) with an Index directory.
Each sub-directory contains one file per Type per Property, and per diagram.



The root directory will contain the common files (logo, background, icons,..), depending upon the Theme that has been selected at generation time, as well as the entry point:

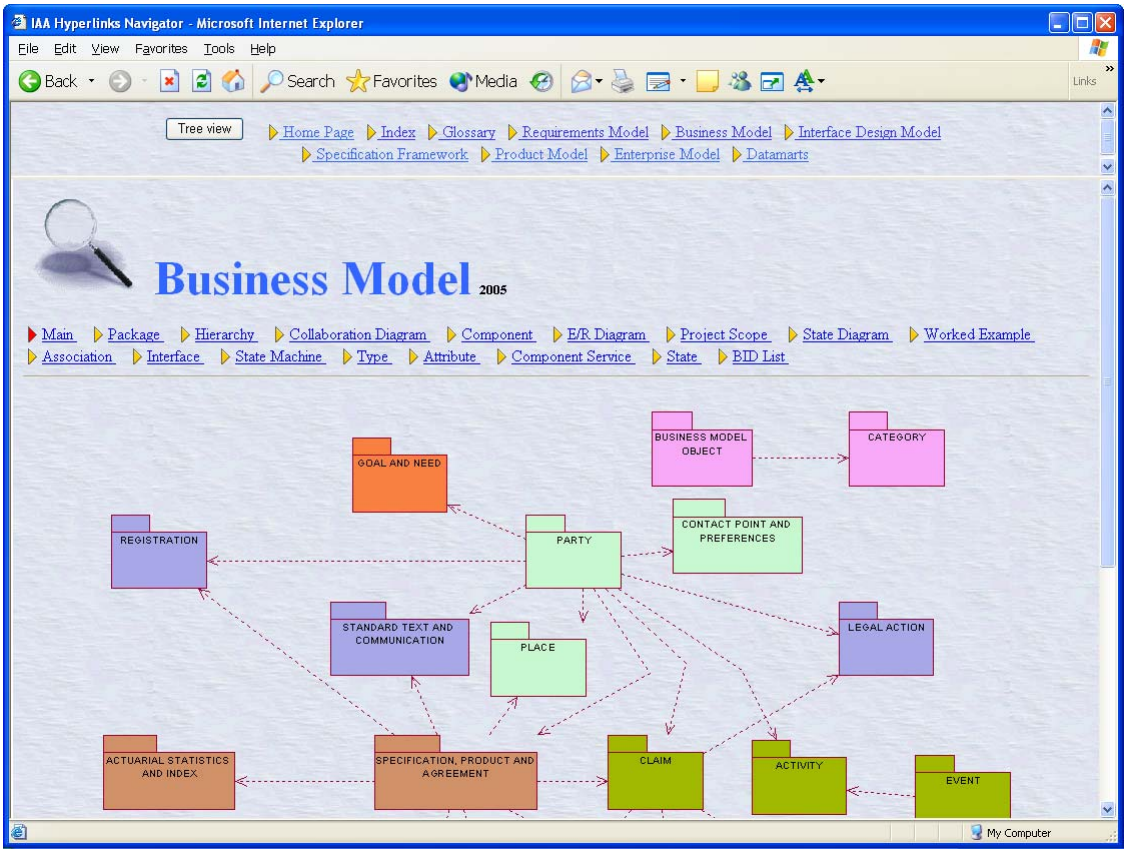
Home.htm



Main pages

For each model, there is a set of entry points:

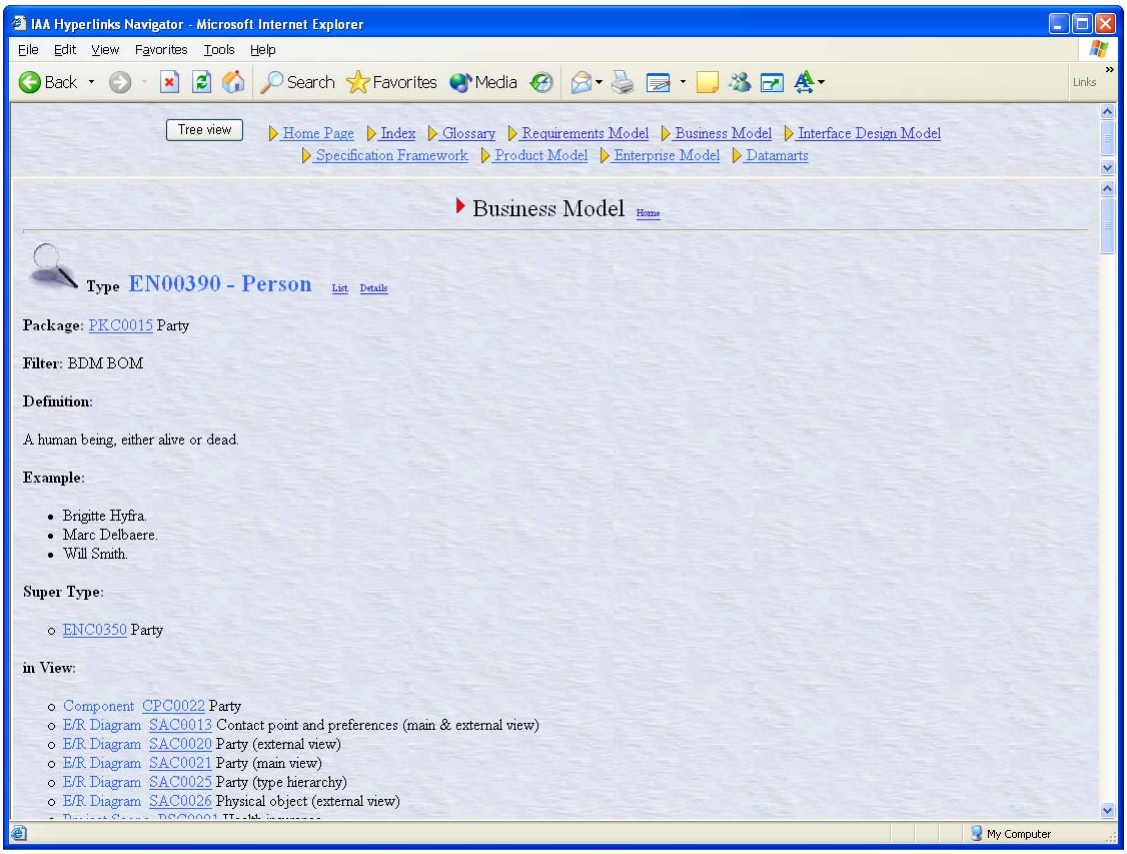
- Main View (optional, manually created as customised entry point)
- Hierarchy View (Tree view of all Types and their sub-types)
- Package View (Tree view of all Packages and their Types)
- One View per View object type defined
- One List per Type object type defined
- One List per Type Property object type defined
- One BID List (all Types ordered by BID)



Type pages

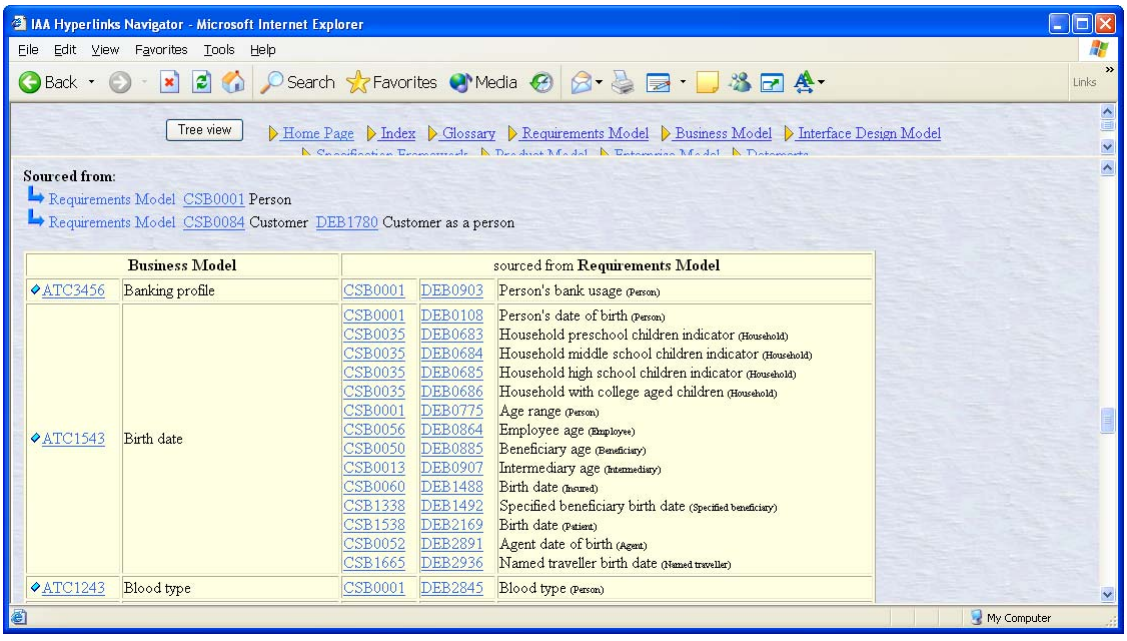
For each **Type**, there is an HTML file which displays the Type's details, including the Type's package, definition, examples, super-type, sub-types, list of views the type is part of, as well as its associations and mappings. It's then followed by a table list of all properties and inherited properties and their corresponding mapping.

When a definition or an example is **inherited** from a Source Model, the text appears ***in italic***. The mapping which is responsible for the inheritance is ***in italic*** as well.

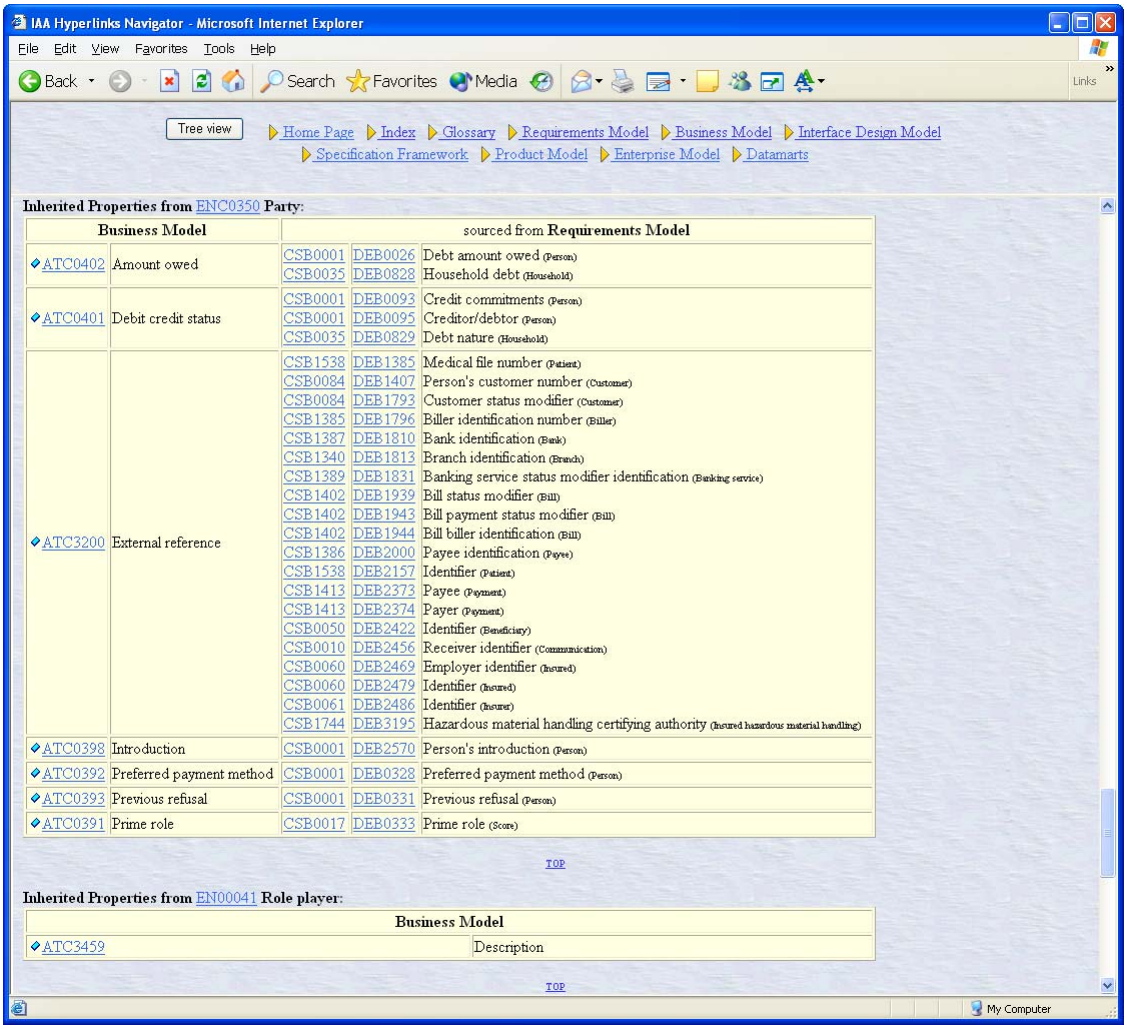


In the **MMM**, mappings are provided from a Model to its Source Model. In the **Hyperlinks Navigator**, note, however, that the mapping from the Source Model to the Model is also derived and displayed.

Mapping information is always preceded by a  symbol.



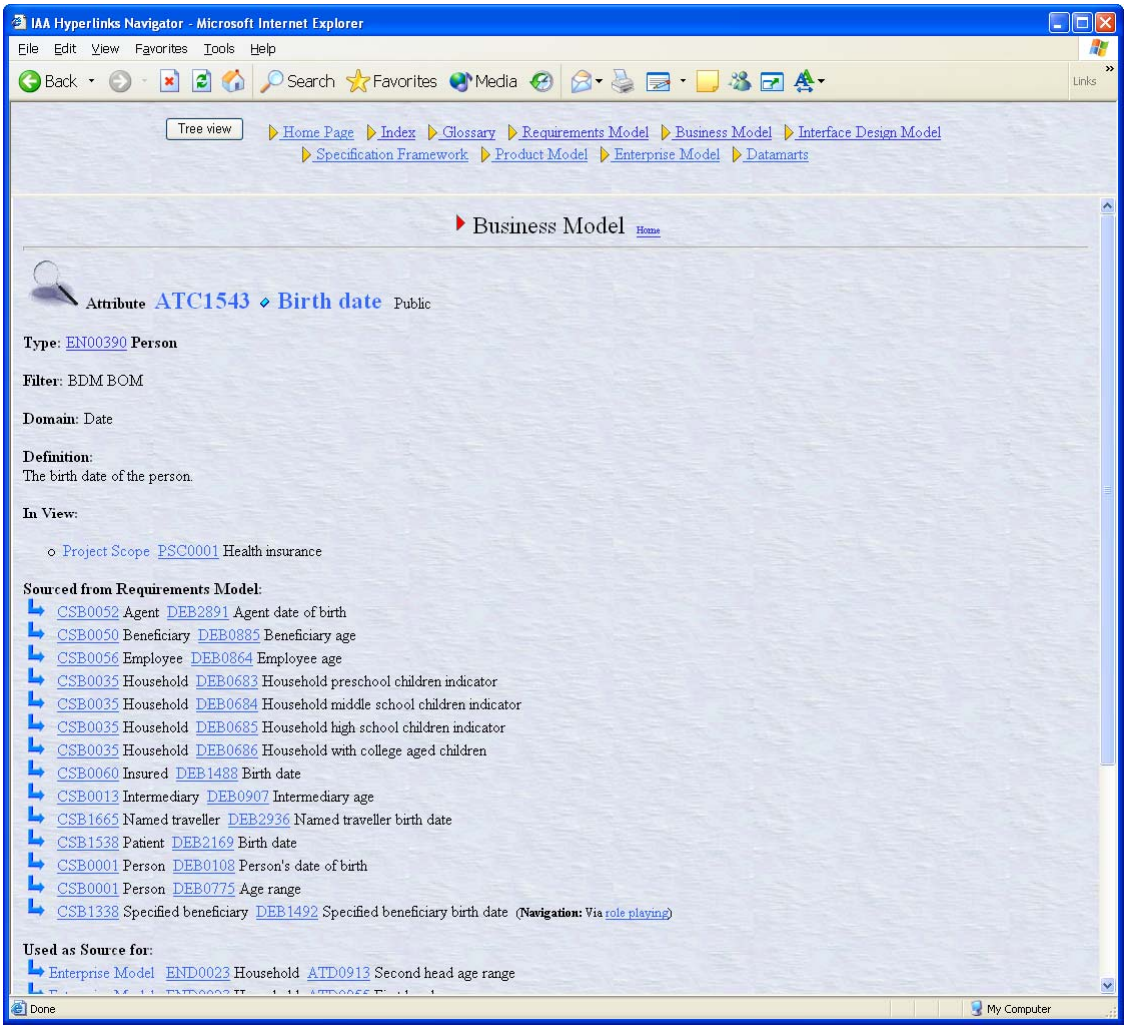
A table listing the Type's **Properties** follows the Type details. **All inherited Properties** (according to the super-Type structure) are included as well.



Type Property pages

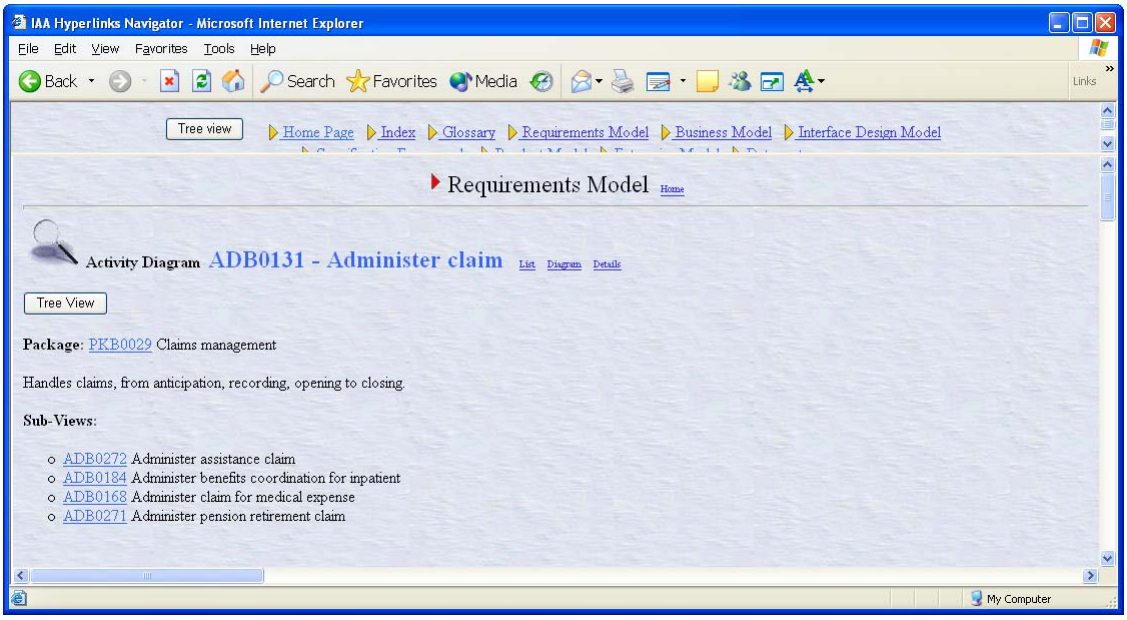
For each **Type Property**, there is an HTML file that displays its details: type, definition, examples, domain, return type, parameters, and mapping.

When a definition or an example is **inherited** from a Source Model, the text appears ***in italic***. The mapping which is responsible for the inheritance is ***in italic*** as well.

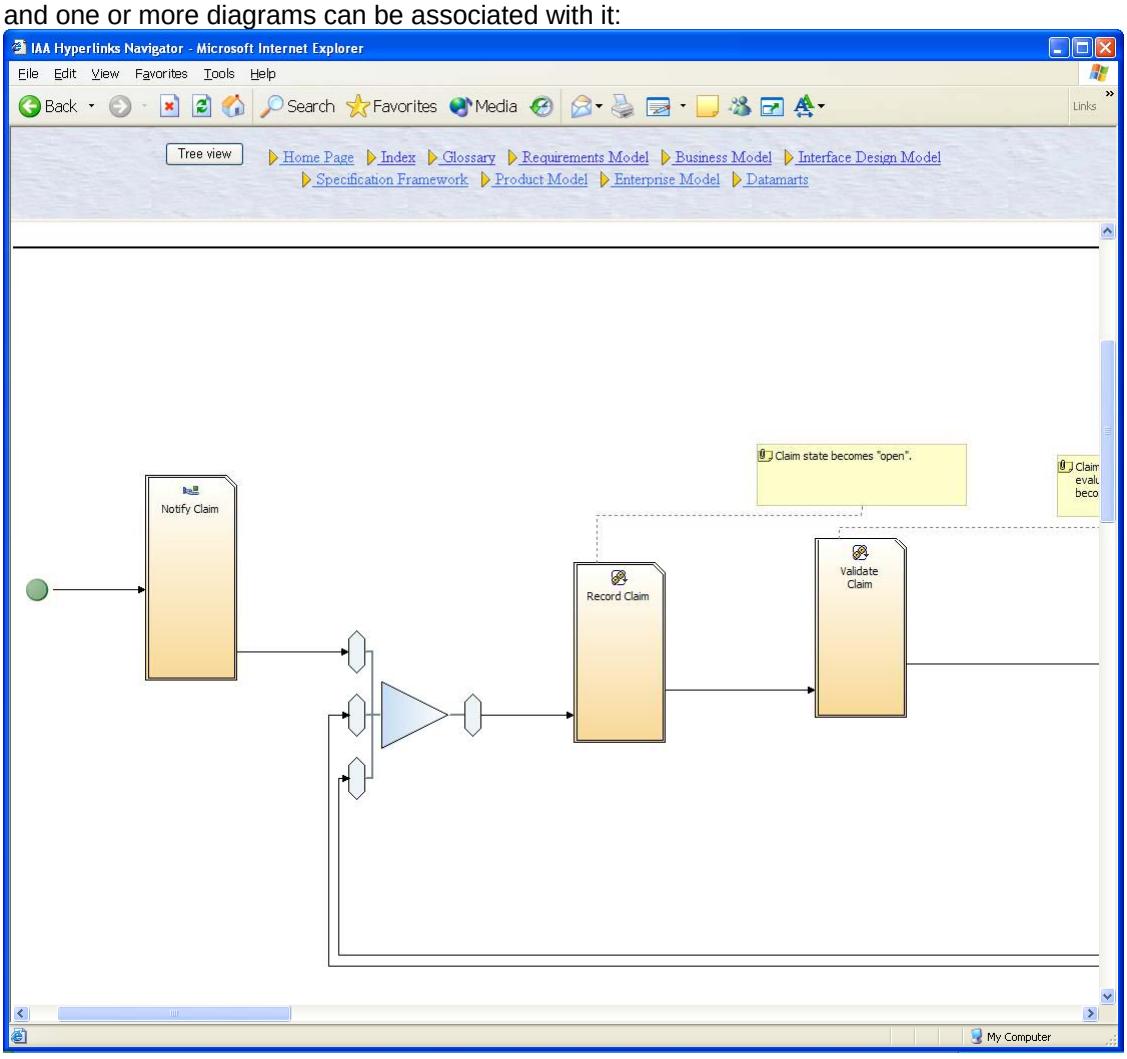


View pages

For each **View**, there is an HTML file that displays its details: view type, definition, view members, and mapping,

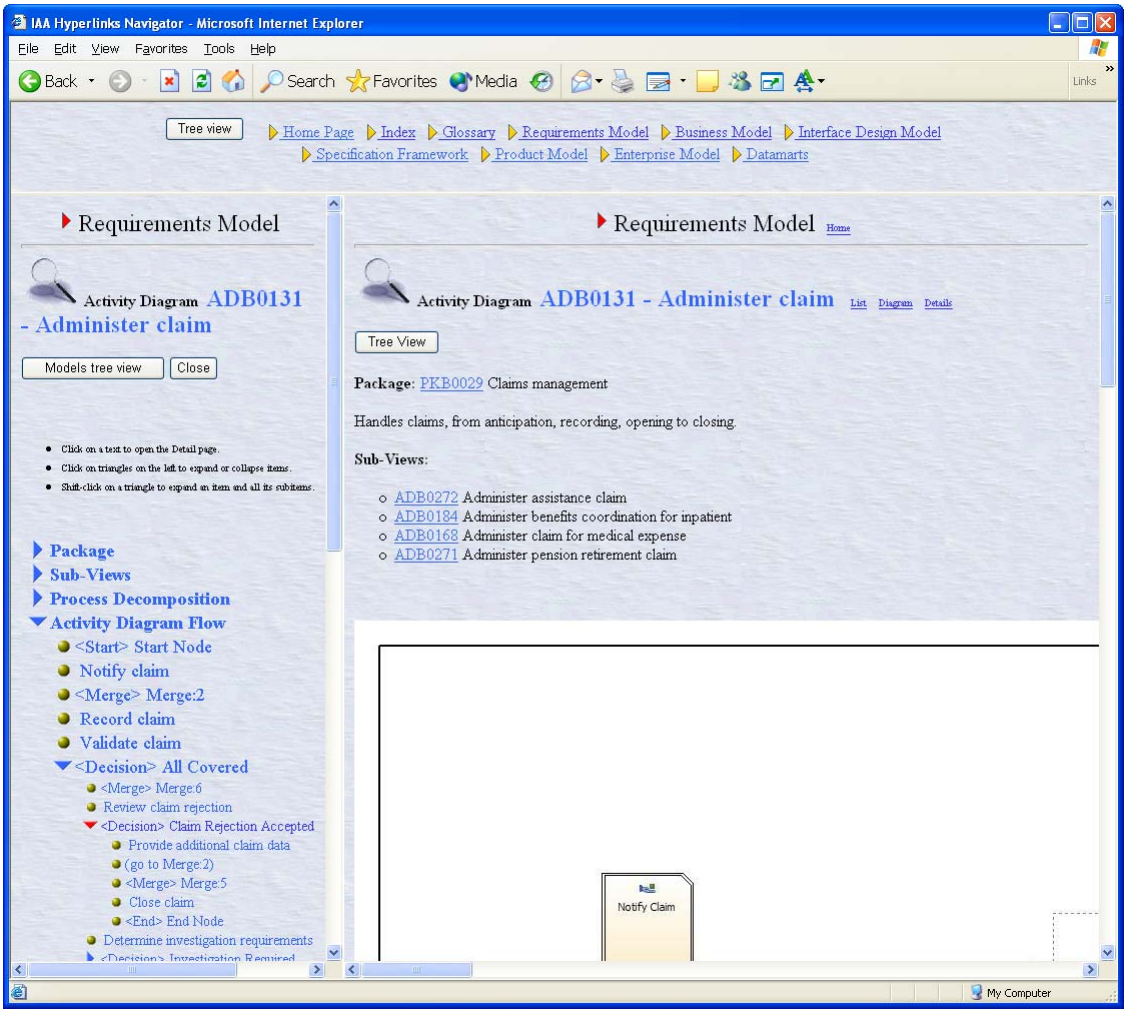


and one or more diagrams can be associated with it:

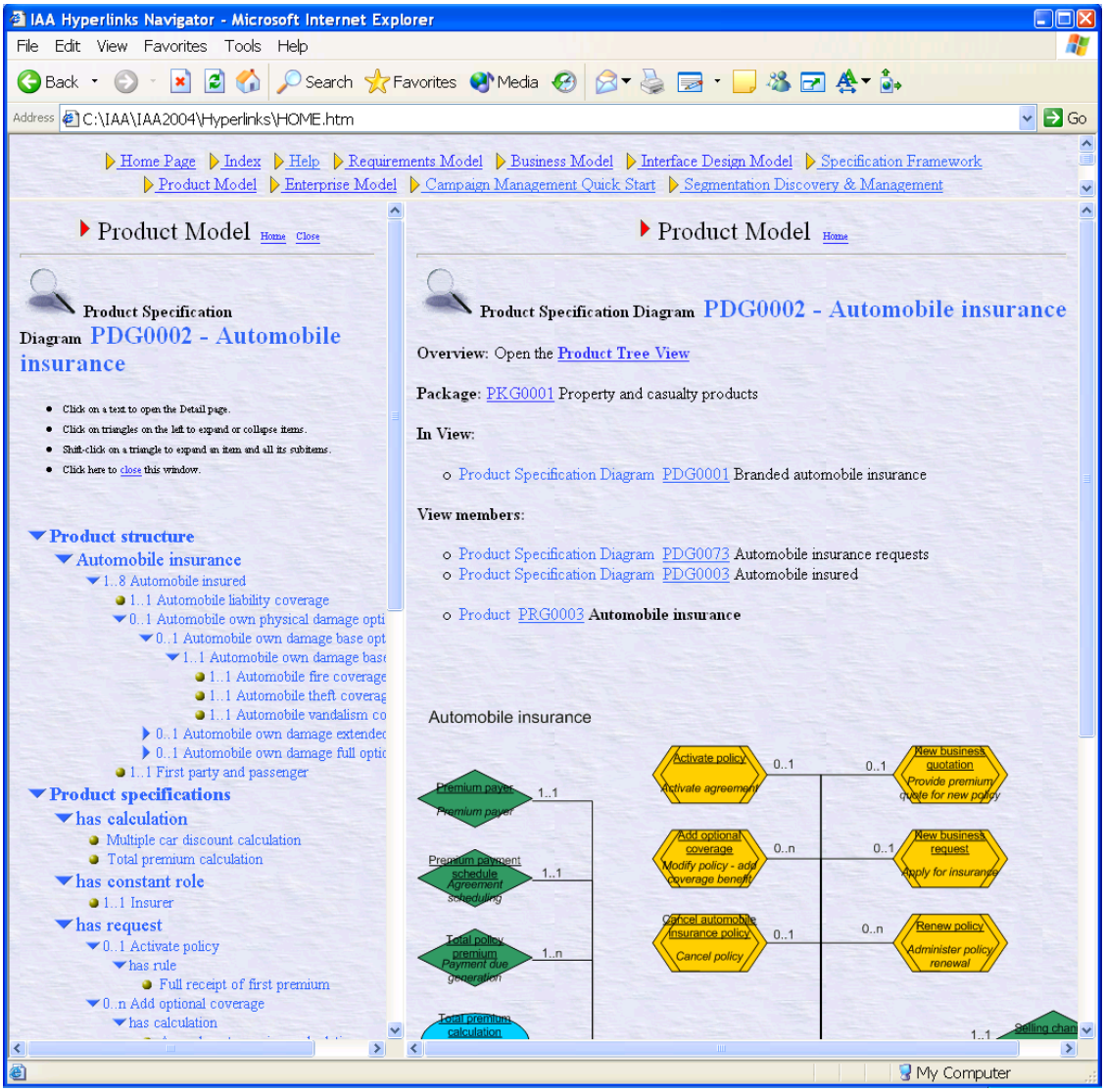


All views can be displayed as a **Tree View** in a left pane.
This left pane tree view can then be used to easily and quickly navigate on the right pane.
Dedicated tree views are also available to represent the Process Decomposition, the Activity Diagram Flow, the Product structures and the Business Solution Templates.

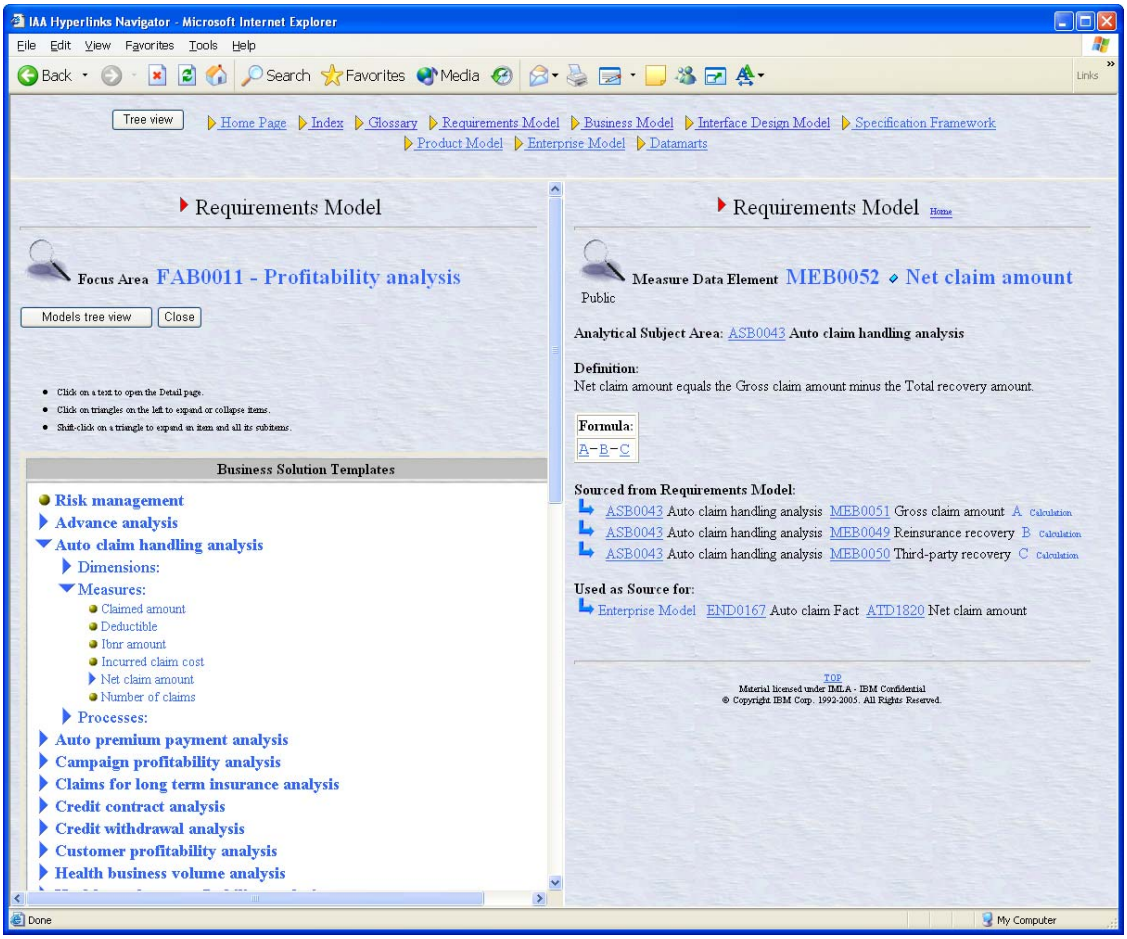
Example of **Activity Diagram Flow** tree view:



Example of **Product structure** and **Product Specifications** tree views:

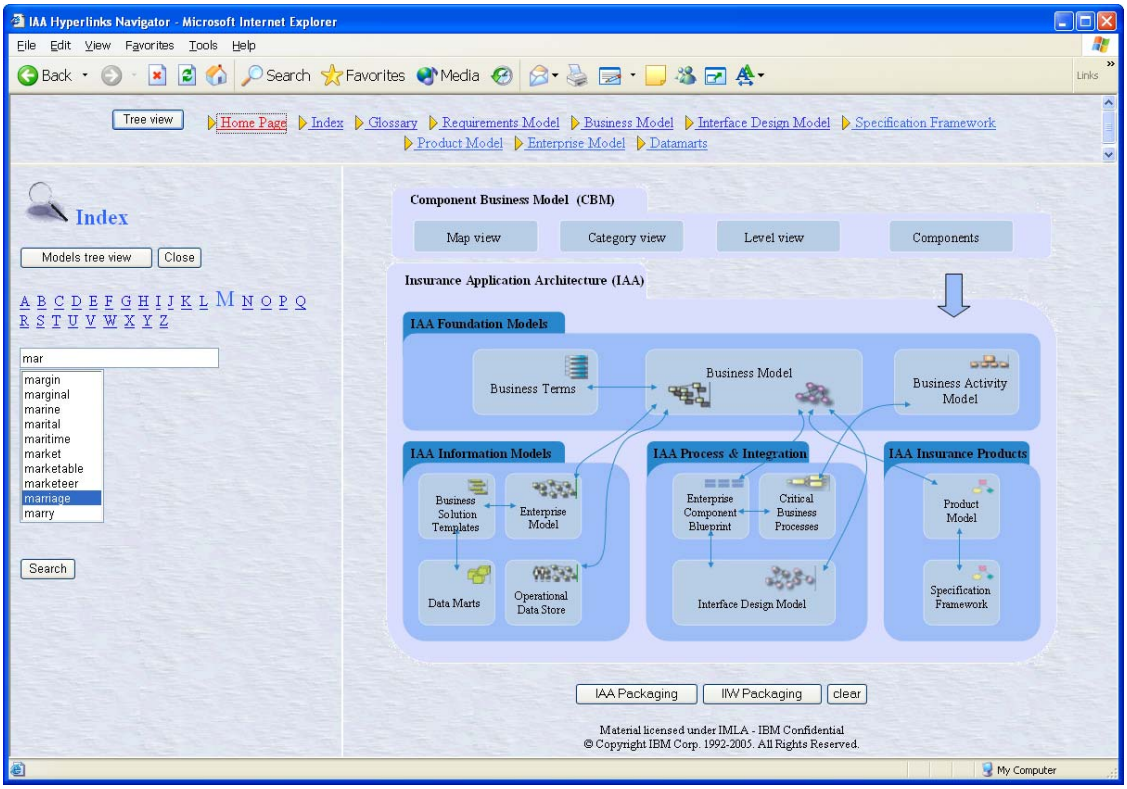


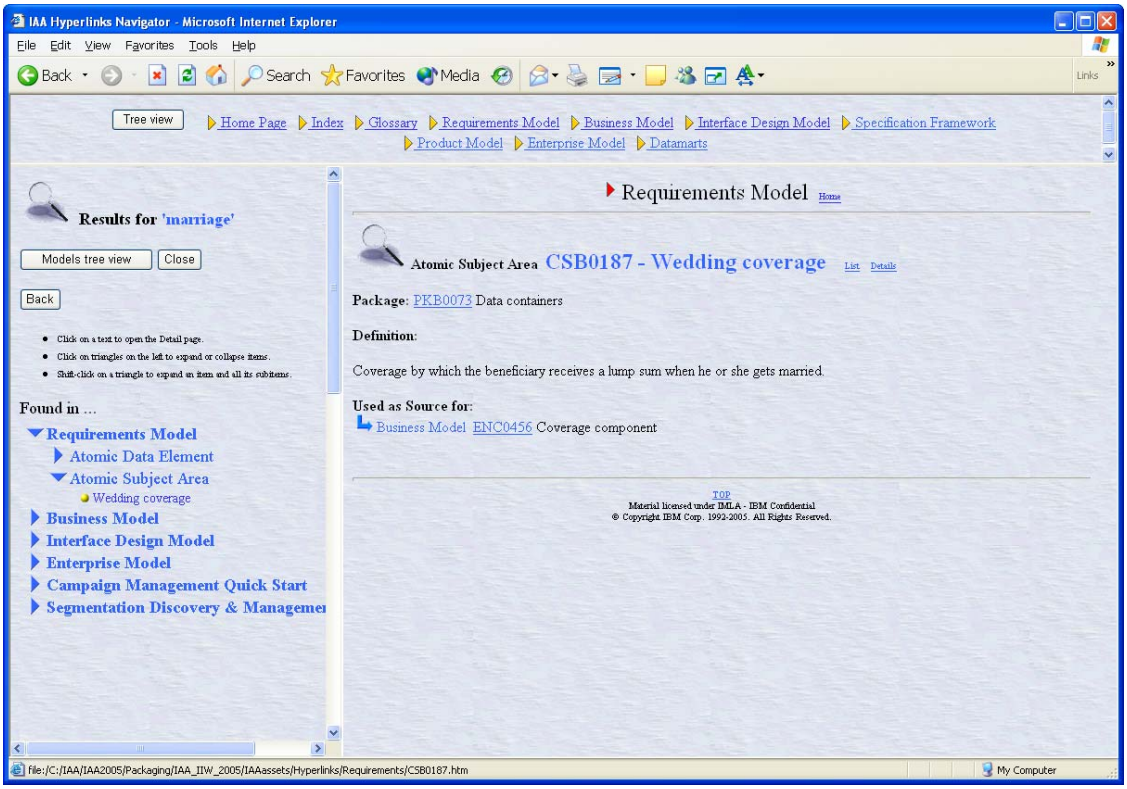
Example of **Business Solution Templates** tree view:



Index pages

The Hyperlinks **Index** covers all of the models and is based upon words found in the Type and Property Names. Optionally, the definitions can also be used as a source of the index.
Synonyms and similar words (including tense and case variations) are grouped in the same Index entry thanks to the *IndexMapping* table, and meaningless words (such as **a**, **one**, **the**, **for**, etc…) are excluded from the Index based upon the *IndexNotToAppear* table.

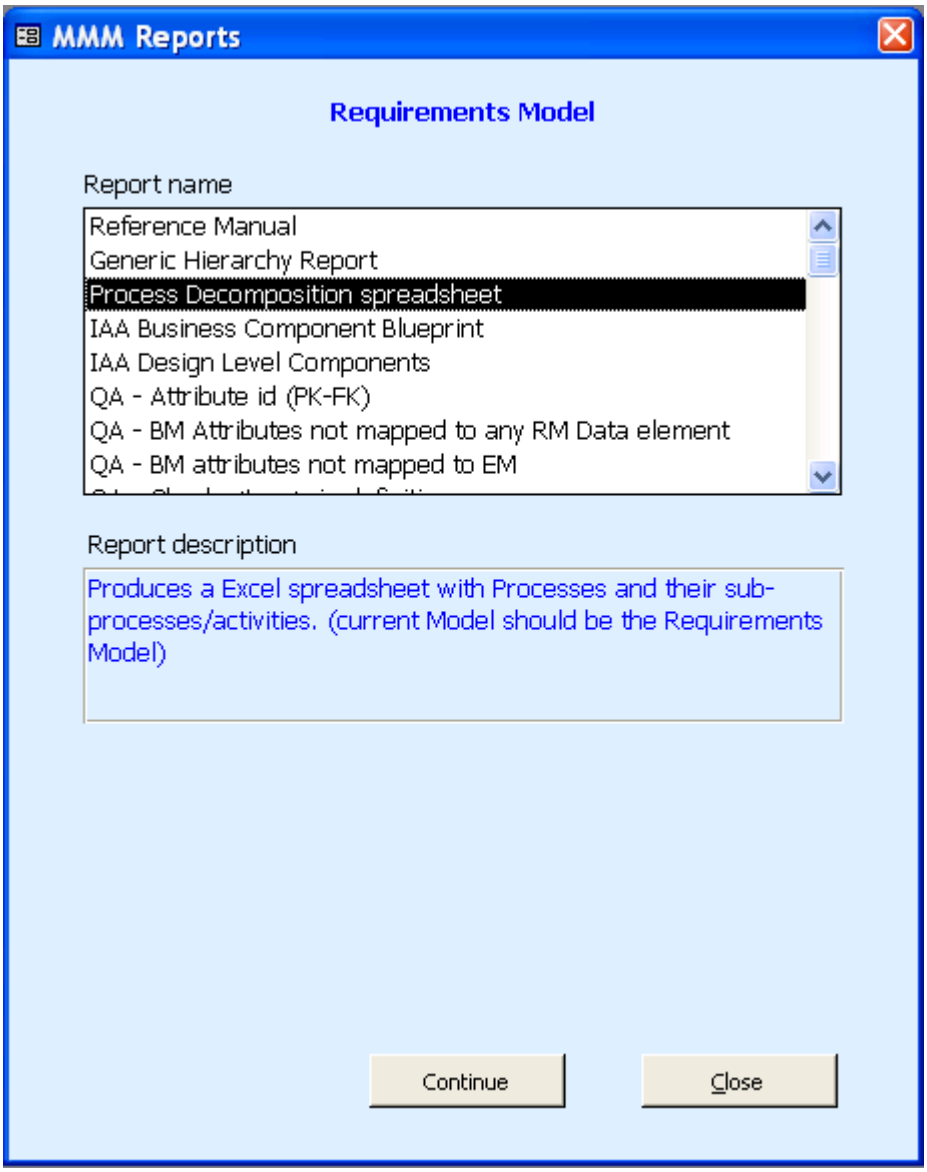




CHAPTER 5: GENERATE REPORTS

5.1 The reports menu

Several reports are available in MMM, these are accessed via the reports menu. The underlying design of the reports menu allows users to easily integrate custom reports to the menu. For details on how to integrate custom reports into the reports menu refer to the section 5.5 *Defining additional reports*



The reports menu (shown above) lists all reports available in MMM, this list is obtained from a table in which report details have been entered (after the report has been physically created in MS Access).

The reports menu shows the selected model. A brief report description is also shown on the menu when the user clicks on the report name.

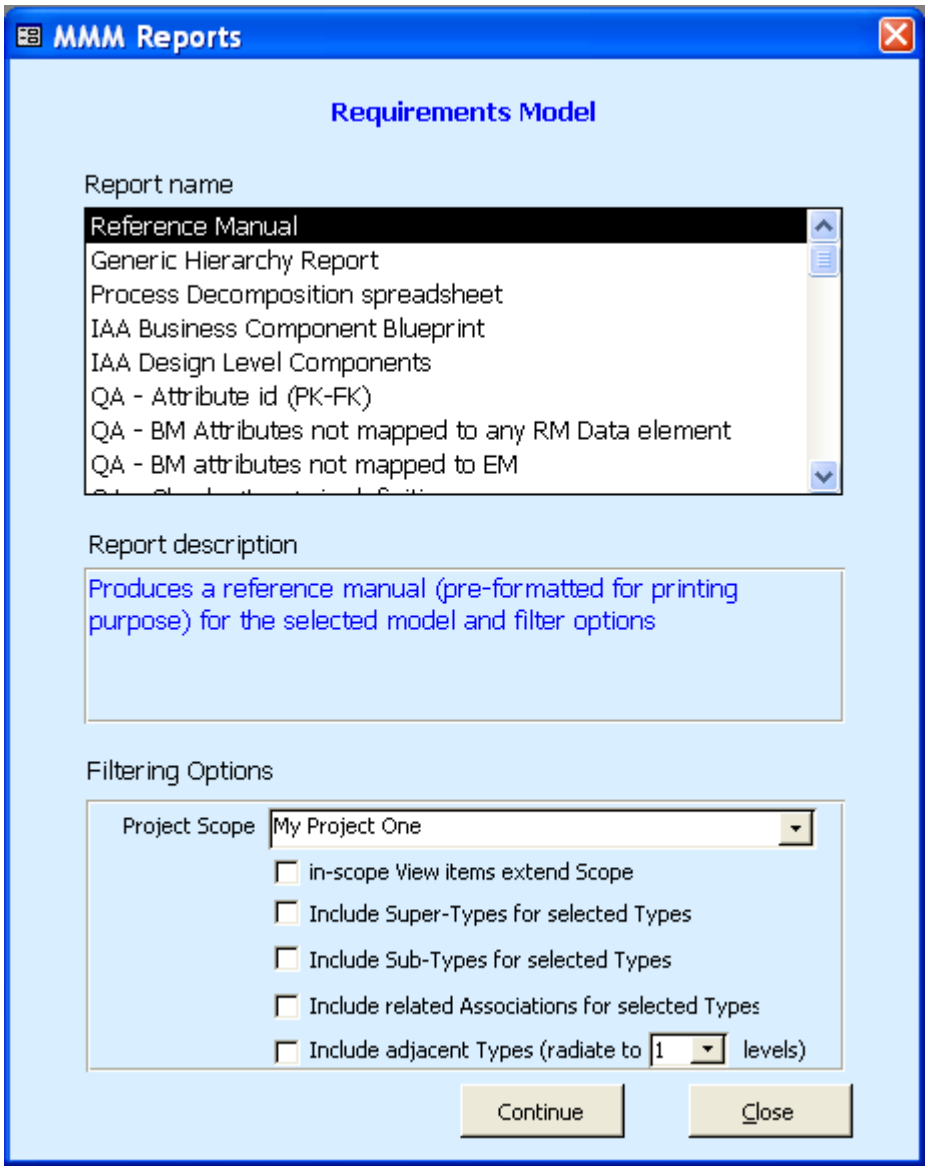
A report is generated when the user selects the desired report and then clicks on the *continue* button. If the report requires the user to select generation options then a form specific to that

report is displayed. If no user input is required, report generation commences when the user clicks the *Continue* button.

If a report requires no user input and it implements the *filter on export field* feature, the reports menu will allow the user to specify the filter. The filter field is shown on the reports menu only if specified in the definition of the report in the reports table.

5.2 Generate the Reference Manual

The reference manual report shows details of objects within a given model in MMM.



Following is **sample** information shown on the report:

- Lists objects based on filter specified
- Selected model name and version
- Report generation timestamp
- Filter specification
- Object definition (inherited and/or local). Inherited definitions will be shown in *Italic*
- Object examples (inherited and/or local) Inherited examples will be shown in *Italic*

- The package the object belongs to
- Object dependants
- Object parent if any
- Object properties and their associated definition and examples
- If an object property is of type *attribute* the report will detail if it is a Primary or Foreign key
- Association object details

Filtering options

When **filtering on Project Scope(s)**, the following options can be used to extend the scope as defined in the Project Scope View(s):

In-scope View items extend Scope
The user can decide whether the views that are defined in the Project Scope view can themselves be used to define the Types and Properties that belong to the scope.
If the option is selected, the view items extend the scope.
If the option is not selected, the views in the target model will only contain Types and Properties that belong to the scope.

Include Super-types
The user can decide to include or not the Super-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Organisation*, *Party*, *Role player* and *Business model object* will be copied in the subset as well (as being super-types).

Include Sub-Types
The user can decide to include or not the Sub-types for the Types that are included in the Project Scope.
For example, in the Business Model, by having included *Organisation Unit* in the scope, *Branch, Department, Employment position, Regional unit,* and *Team* will be included in the manual as well (as being sub-types).

Include related Associations
The user can decide to include or not the Associations that have the two parent Types included in the Project Scope.

Include Adjacent Types
The user can decide to include all Types that are next to the in-scope Types. The broadness of the inclusion can be set by the number of levels of radiation. (Can be for example used in the Product Model to select one Product and all its specifications and sub products).

The  function provides a full Resulting Scope Log that explains the “why” for the presence of each instance in the scope.

Once report generation is complete, the report is displayed in preview mode and can be printed or exported to MS-Word if required. Some sample report previews follow:

Sample report preview showing report heading:

Model : Business Model

Model Version : 2003, Report Generated on Monday 19.May.2003 at 15:16:59

Filter = All, report sorted by Object Type/Object Name, properties sorted by Sequence/Name.

Object Name	Object type	BID
account - activity rlship	Association	ENC0012

Definition:

A relationship between an occurrence of <account> and an occurrence of <activity>.

Dependents:

Sample report preview showing association details:

account claim tracking	Association	ENC0877

Package: Account and fund *Parent:* account - claim rlship

Definition:

The identification of an <account> as the one that tracks the monetary transfers of a <claim>.

Association details:

EN00130 Account tracks (0..1) EN00122 Claim is tracked by (0..n)

Left aggregate: Yes

Sample report preview showing inherited definition (*in Italic text)* and local definition:

Report financial position and balances	Operation	OPC4005

Definition:

Amount and currency of the balance of the account.

Produces reports on financial position and balances, for example, documentation on customer or broker or re-insurance (useful for renewing treaties). They may be produced for internal use or to meet internal audit requirements.

Examples:

An example of an account balance

Sample report preview showing requirements detail at the Type level:

Object Name	Object type	BID
Account	Type	EN00130

Package: Account and fund

Definition:

A title under which records of financial items are grouped.

Examples:

A savings account
A cheque account

Requirements:

Priority	*Owner*	*Organisation*	*Location*
Medium	John Iertile	IBM Australia	Collins St Melbourne
Distribution level	*Volumetrics*	*Number of Users*	
Corporate	100 new records per month	200	

External documentation

c:\ProjectX V1.doc

Data quality specifications

Data must be cleansed prior to loading into WH.
See DQ specs for individual properties.

History specifications

Data retention for a period of 7 years from the transaction date is required by legislation.

Security specifications

Data can be accessed by all users accross the Corporation

Transformation specifications

See transformation specs for individual properties.

Sample report preview showing requirements details at the property level:

Properties

Closing date Attribute AT01516

Domain: Date

Definition:

Date and time on which the balance of the account ceases to be effective.

The date on which the account is closed. This means that the balance of the account needs to be set to zero, and no more entries can be made for the account.

Examples:

Any valid date in the form of dd/mm/yycc

Requirements:

Priority	*Owner*	*Organisation*	*Location*
High	John Ientile	IBM Australia	IBM Towers, Southgate

Distribution level

Local

Data quality specifications

Data quality is believed to be average. This field in the legacy application is not always validated as being a valid date.

History specifications

History must be maintain for a period of 7 years from the transaction date.

Security specifications

Not applicable

Transformation specifications

This must be converted into a valid date prior to loading into the warehouse. Transform all invalid dates to 01/01/1900.

Sample report preview identifying properties as Primary/Foreign keys:

Object Name	*Object type*	*BID*
Asset holding	Type	END0047

Package: Atomic - Account and fund *Parent:* Account

Definition:

An account that holds a specific number of units of a <financial asset>. It can be viewed as an account that does not hold monetary amounts but assets.

Examples:

50 Units in The World Stock Fund Plus Global.
Ten shares of IBM.

Properties

Account id (FK) Attribute ATD2446

Domain: Identifier

Definition:

Any value without business meaning that uniquely distinguishes occurrences of this entity independently of its history.

Asset holding id (PK) Attribute ATD2443

Domain: Identifier

Definition:

Any value without business meaning that uniquely distinguishes each occurrence of this entity.

Closing date Attribute ATD0445

Domain: Date

Definition:

The date on which the account is closed. This means that the balance of the account needs to be set to zero, and no more entries can be made for the account.

Examples:

Any valid date in the form of dd/mm/ccyy

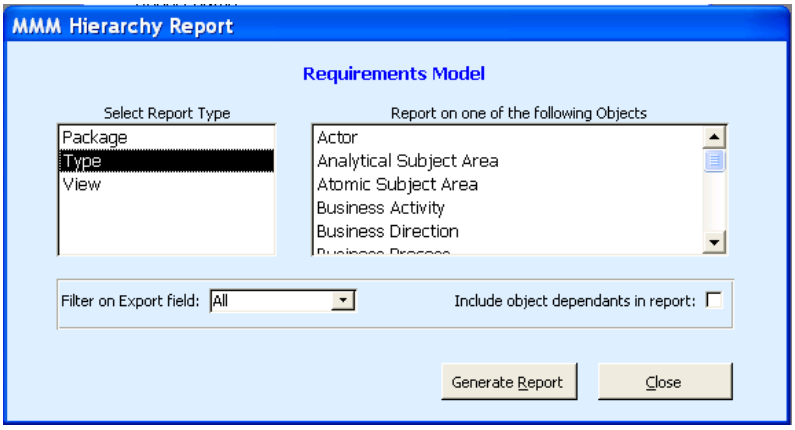
External reference Attribute ATD7239

Domain: String

5.3 Generic hierarchy report

The generic hierarchy report will show the hierarchical structure of objects of a given type within a given model in MMM.

The following user interface is provided whereby the user can specify report generation options. As objects in MMM can group other objects, for example an object of type *Package* groups other objects also of type *Package*, and an object of type *Type* can group objects of type *Attribute/Operation* etc (depending on the model in context) the user interface allows the user to select which object type to include in the report. The list of object types available for selection will change dynamically based on the model in context.



The report is generated as a Microsoft Excel Workbook. The report is produced in ascending alphabetical order within each node of the hierarchy.

Generation options/features:

Include object dependants in report

Users have the option of including object dependants in the final report. When this option is selected, object names in the hierarchy will be shown in **bold** (even if the object has no dependants) and the objects' dependants (prefixed with the object type and BID) will be shown in *italic* to enhance readability. When dependants are excluded from the report no font changes are made. See partial sample report below:

14		PKB0067	Actors						
15		PKB0046	Channel management						
16			PKB0028	Channel set up					
17				PKB0072	Compensation plan establishment				
18				PKB0060	Interm agmt spec design				
19				PKB0063	Intermediary establishment				
20			PKB0013	Intermediary management					
21				PKB0065	Intermediary agreement administration				
22				PKB0070	Intermediary commission management				
23				PKB0064	Intermediary performance monitoring				
24				PKB0062	Intermediary support				
25		PKB0029	Claims management						
26			PKB0038	Benefit offering					
27			PKB0088	Claim anticipation and loss event maintenance					
28			PKB0023	Claim investigation					
29			PKB0004	Claim recording					
30			PKB0017	Claim recovery					
31			PKB0034	Claim reporting and statistics					
32			PKB0066	Claim settlement					
33			PKB0010	Claim validation					
34			PKB0042	Service and claim status					
35		PKB0009	Communication management						

***Include objects (*or *Report on one of the following Objects* List Box)**
Worth noting, is that MMM defines many different object types, however, not all of these types are necessarily used by a particular model. The user interface however will only show object types that have actually been used in the selected model.

Filter on export field
This functionality is a standard feature of MMM and has been adopted for the hierarchy report, this feature essentially provides the ability to report on objects in a particular scope.

Auto generated hierarchy root level
In some cases in MMM there is no real hierarchical structure of objects, *component* objects for example are not hierarchical, in such cases a dummy top level for these objects is automatically created in the final report, this simply aids visibility of the report and highlights the fact that these objects are not hierarchical in nature. See partial sample report below:

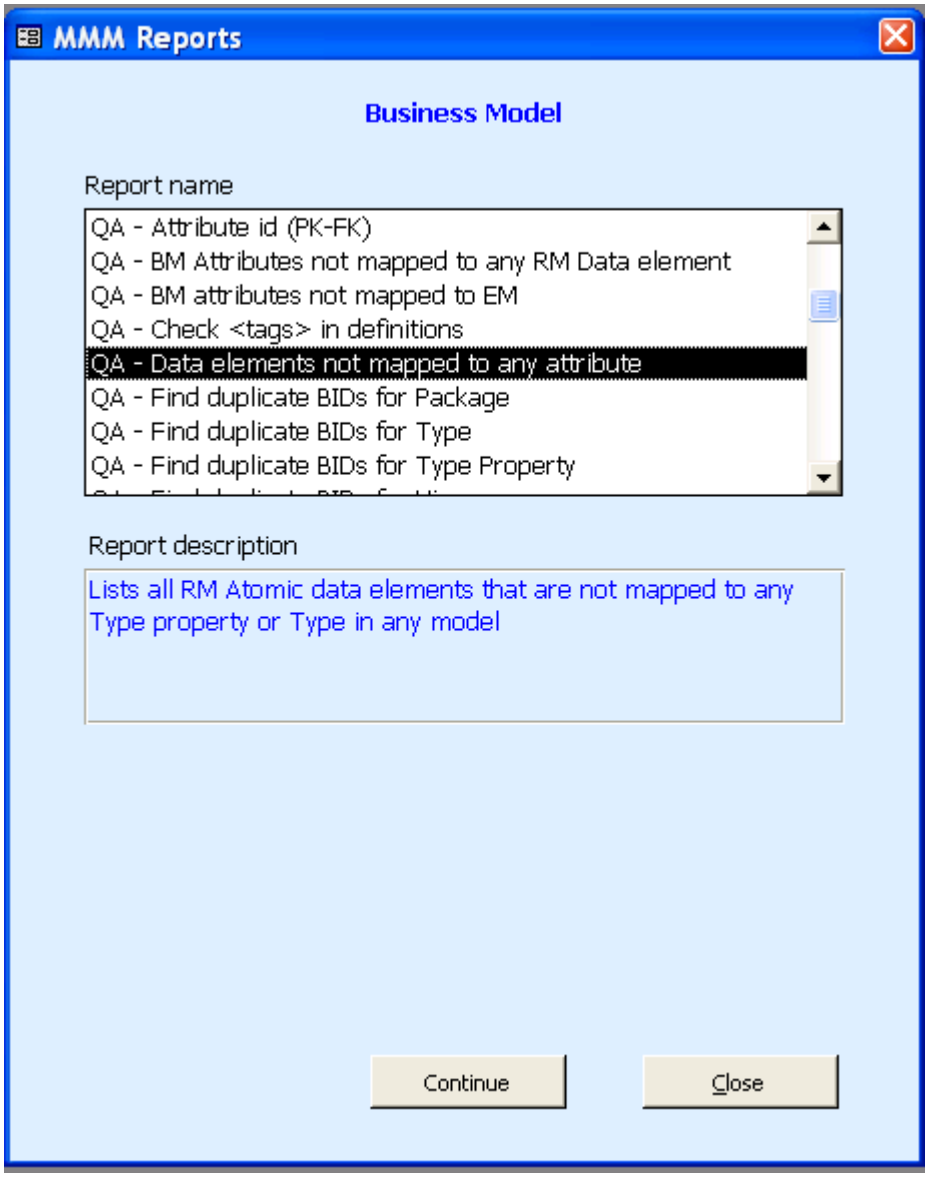
User specified options
When the report is generated it will contain as part of the heading information (as shown above), details of the generation options specified by the user, for example it will detail:

- The model name and version
- Report type
- Include object
- Include object contents in report
- Filter

Heading information is frozen at the top of the worksheet. This allows the heading information to be seen as the user scrolls through the content of the report.

5.4 Quality Assurance reports

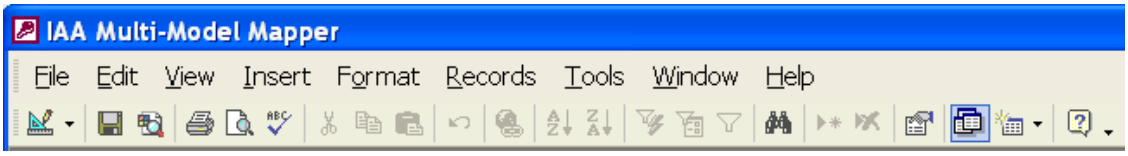
MMM delivers several predefined QA queries, these are accessible from the reports menu. Additional queries can be easily created using the MS-ACCESS wizards and integrated into the reports menu.



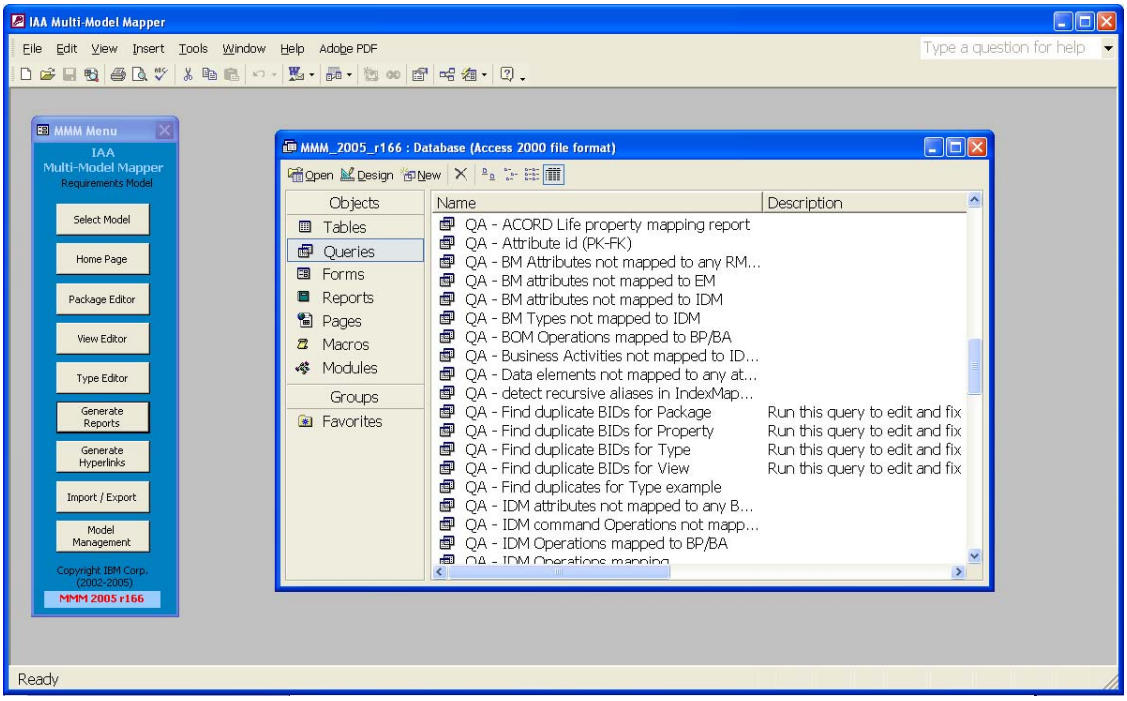
Some additional reports can be created, such as quality assurance (QA) queries, or reports, then added in the MMM Reports Form.

5.5 Defining additional reports

To open the Query window, click on the  button from the main Toolbar



Some queries are already predefined, and new ones can easily be created using the MS-ACCESS wizards.



Once the query defined, it can be catalogued in the MMM table **Reports**. It will then appear in the list of available reports.

CHAPTER 6: IMPORT / EXPORT SRI

The Simplified Readable Input format (.SRI) file consists of multiple lines in a delimited ASCII format. Each line contains two tokens, in the order given:

- Tag name
- Value

It contains blocks of tags that provide the information necessary for creating or updating an object (Package, View, Type, Association, Property, Mapping).

The SRI can be based on **Names** (identifying an object based on its name) or on **BID** (identifying an object by its unique Business Identifier).

An SRI file based on Names is less rigorous, and has some restrictions. For example, it does not allow for renaming an object, or managing non-unique names.

To exchange the complete content of a model, six SRI files are needed:

- An SRI that contains all Packages and Views (usually called Package.SRI)
- An SRI that contains all Types –and Association Classes- (usually called Type.SRI)
- An SRI that contains all Associations (usually called Association.SRI)
- An SRI that contains all Properties (usually called Property.SRI)
- An SRI that contains all Mapping to other models (usually called Mapping.SRI)
- An SRI that contains all additional information -Requirements Tab- (usually called Extensions.SRI)

6.1 Import SRI

This function on the MMM main Menu imports a model, (or part of a model) into the current MMM model using the SRI format. It looks for an existing object in the MMM Repository based on the **Name** or the **BID** (depending upon which SRI format is used), and if not found, the new object will be inserted.

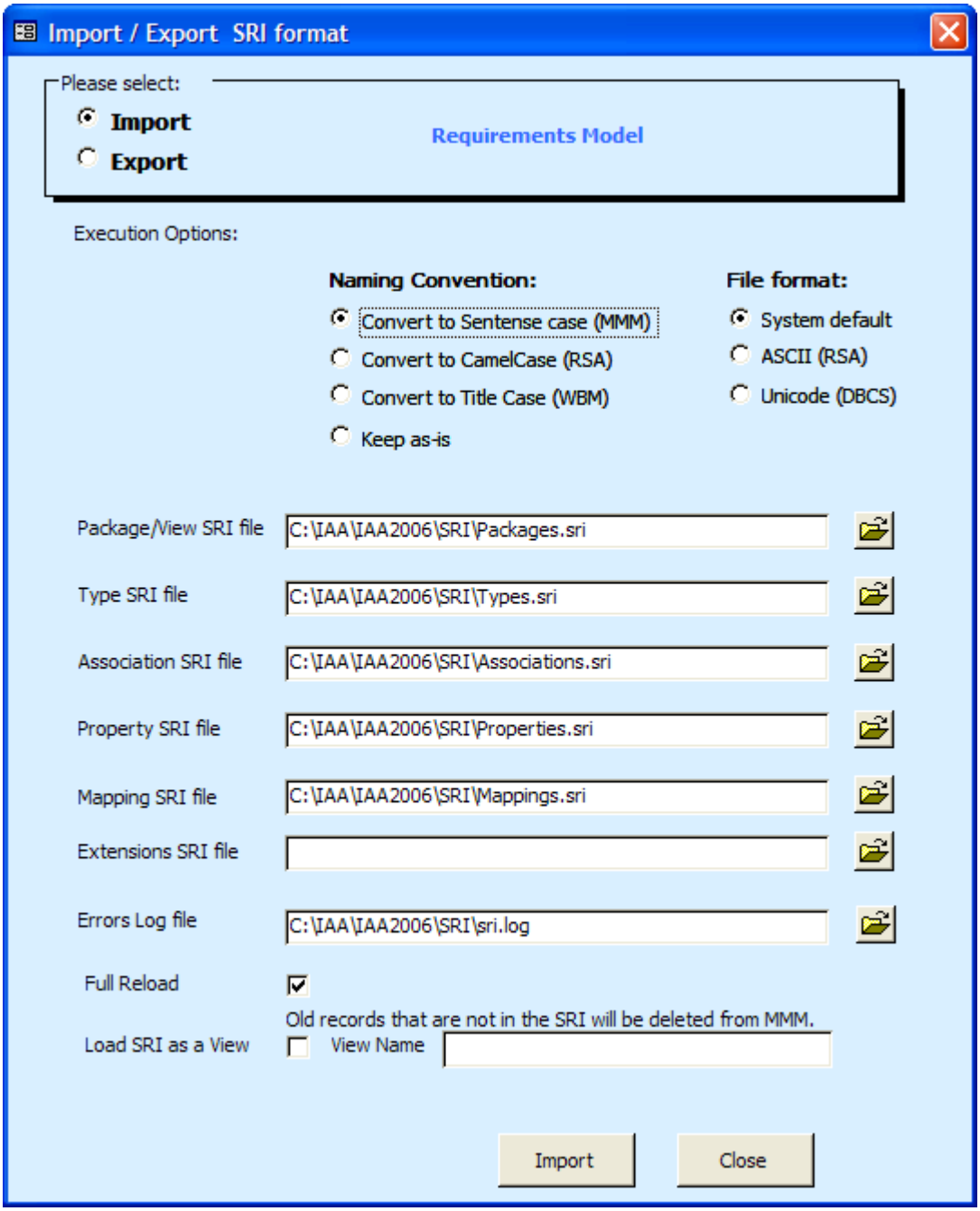
Naming Convention:

Object names can optionally be converted into Business Names or into the Object Oriented Syntax (see *Standards and Naming Conventions* Appendix)

File format:

There are three options for file format: System default, ASCII and Unicode. Unicode should be used when the database contains language-specific characters. For instance, Japanese characters require a double-byte character support and therefore the SRI files need to be created with the Unicode option.

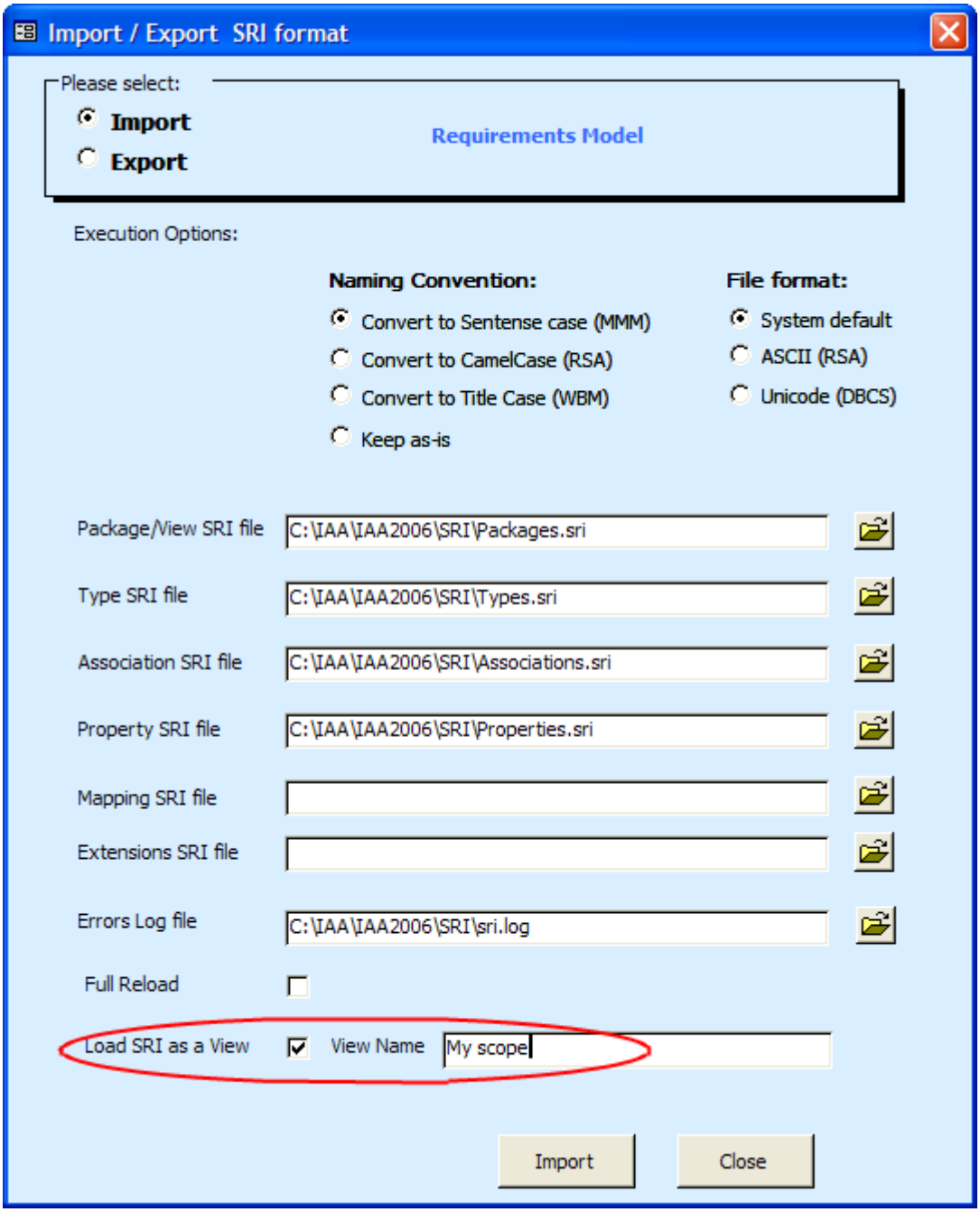
The remaining fields are used to designate input files. The optional input files enable the user to customise the import or export. For example, by only specifying the Property SRI file, it would be possible to re-load just the Attributes and Operations.



When the **Full Reload** option is selected, MMM will propose, at the end of the Import process, to delete all Packages, Views, Types, Type Properties, and Mappings that were not inserted or updated by the provided SRI.

6.2 Import SRI as View

This option if the Import SRI function makes it possible to create a Project Scope View that reflects the content of the SRI provided as input, instead as importing the input SRI as actual instances.
The actual instances must pre-exist in the model.



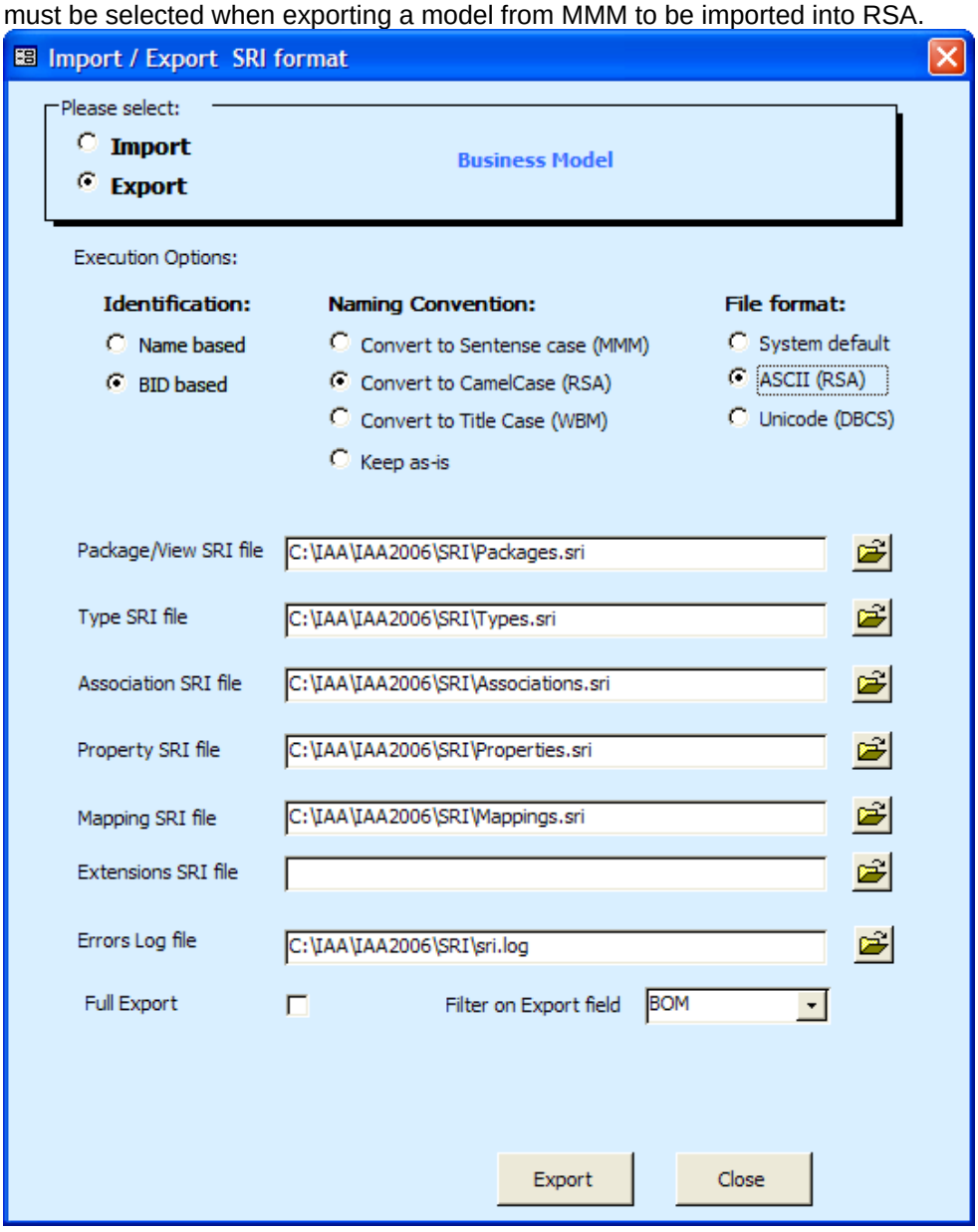
6.3 Export SRI

When you select the Export option, two extra fields are displayed:

Identification:
This field allows the user to choose the SRI format, which can be either Name-based or BID-based. The latter is recommended whenever possible.

Filter on Export field:
This drop-down field is used to designate filtering for the content to be exported. For example, if the filter is set to “FILT1”, all Types that either have their filter field set to “All” or to FILT1”, as well as to any value that begins with “FILT1”, such as “FILTA”, “FILT1B”, will be exported.
In case of the Business model, the Filter field has also been used to filter entities/attributes, which are specific to the **BOM** (the export for RSA) or to the **BDM** (the export for ERwin®).
When exporting the full content of a model for backup or migration purpose, the filter can be deactivated by selecting the **Full Export** option.

Important: The current version of RSA bridges only support ASCII files. The ASCII option must be selected when exporting a model from MMM to be imported into RSA.



6.4 SRI Format

Below are some examples of Name-based and BID-based SRI file content.

Example of Type.sri (name-based)

```
"AddClassName" "AccessFacility"
"SetClassBID" "ENC0500"
"SetClassType" "Type"
"SetClassDoc" "A facility that provides a particular type of access to an <account agreement>."
"SetClassExpl" "Web banking.<NewLine>Phone banking."
"SetClassStereotype" ""
"SetClassCategory" "SPECIFICATION, PRODUCT AND AGREEMENT"
"SetClassInheritFrom" "AccountFacilityComponent"

"AddClassName" "Account"
"SetClassType" "Type"
"SetClassDoc" "A title under which records of financial items are grouped."
"SetClassExpl" ""
"SetClassStereotype" ""
"SetClassCategory" "ACCOUNT AND FUND"
```

Example of Attribute.sri (name-based)

```
"AddClassName" "Account"
"SetClassBID" "EN00130"
"AddAttName" "closingDate"
"SetAttBID" "AT01516"
"SetAttType" "Attribute"
“SetAttSequence” “1”
"AddAttDoc" "The date on which the account is closed. This means that the balance of the account needs to
be set to zero, and no more entries can be made for the account."
"SetAttDomainType" "Date"
"SetAttStereotype" ""

"AddClassName" "Account"
"AddAttName" "description"
“SetAttSequence” “2”
"SetAttDoc" "A free-text statement used to explain what is represented by this account. The description may
```

Example of Type.sri (BID-based)

```
"AddClassBID" "ENC0500"
"SetClassName" "AccessFacility"
"SetClassDoc" "A facility that provides a particular type of access to an <account agreement>."
"SetClassExpl" "Web banking.<NewLine>Phone banking."
"SetClassStereotype" ""
"SetClassCategory" "PKC0018"
"SetClassInheritFrom" "ENC0631"

"AddClassBID" "EN00130"
"AddClassName" "Account"
"SetClassDoc" "A title under which records of financial items are grouped."
"SetClassExpl" ""
"SetClassStereotype" ""
"SetClassCategory" "PKC0003"
"SetClassInheritFrom" "ENC0468"
```

Example of Attribute.sri (BID-based)

```
"AddClassBID" "EN00130"
"AddAttBID" "AT01516"
"SetAttName" "closingDate"
"SetAttType" "Attribute"
“SetAttSequence” “1”
"AddAttDoc" "The date on which the account is closed. This means that the balance of the account needs to
be set to zero, and no more entries can be made for the account."
"SetAttDomainType" "Date"
"SetAttStereotype" ""

"AddClassBID" "EN00130"
"AddAttBID" "ATC0489"
"SetAttName" "description"
"SetAttType" "Attribute"
“SetAttSequence” “2”
"SetAttDoc" "A free-text statement used to explain what is represented by this account. The description may
contain an indication of the financial items grouped under the account, the purpose of the account."
"SetAttExpl" ""
"SetAttDomainType" "Text"
```

6.5 SRI Syntax definition

Package

- **AddPackName** Creates a *Package (or updates it, searching on its name)*
- **SetPackBID** Sets the *Business Identifier* for the Package
  Or
- **AddPackBID** Creates a *Package (or updates it, searching on its BID)*
- **SetPackName** Sets the *Name* of the Package

- **SetPackDoc** Sets the *Documentation* property for the Package (*)
- **SetPackComment** Sets the *Comment* property of the Package (*)
- **SetPackParent** *Defines the Package as sub-package of the Parent package* (**)
- **SetPackFilter** Sets the *Export Filter* of the Package
- **SetPackCreateDate** Sets the *Creation date* of the Package (yyyy-mm-dd hh:mm)
- **SetPackUpdateDate** Sets the last *Modification date* of the Package

View

- **AddViewName** Creates a *View (or updates it, searching on its name)*
- **SetViewBID** Sets the *Business Identifier* for the View
  Or
- **AddViewBID** Creates a *View (or updates it, searching on its BID)*
- **SetViewName** Sets the *Name* of the View

- **SetViewType** Sets the View *type* (“Component”, ”Subject Area”, ”Project Scope”,…)
- **SetViewDoc** Sets the *Documentation* property for the View (*)
- **SetViewComment** Sets the *Comment* property of the View (*)
- **SetViewParent** *Defines the* View *as sub-View of the Parent* View (**)
- **SetViewCategory** Adds the View into the *Category* (Package) (**)
- **SetViewStereotype** Sets the *Stereotype* of the View
- **SetViewFilter** Sets the *Export Filter* of the View
- **SetViewCreateDate** Sets the *Creation date* of the View (yyyy-mm-dd hh:mm)
- **SetViewUpdateDate** Sets the last *Modification date* of the View
-

(*) If a <NewLine> tag is encountered in the text, the following text starts at a new line.
(**) The value can be either a *Name* or a *BID*
(***) Only used for Views of *Diagram* behavior

View - Type

- **AddViewName** Creates a *View (or updates it, searching on its name)*
- **SetViewBID** Sets the *Business Identifier* for the View
  Or
- **AddViewBID** Creates a *View (or updates it, searching on its BID)*
- **SetViewName** Sets the *Name* of the View

- **AddViewClass** Defines the *Class* as part of the View (**)
  Or
- **AddViewAssoc** Defines the *Association* as part of the View (**)
- Or
- **AddViewObjectType** Defines the *Diagram Object* as part of the View (***)

- **SetViewLabel** Sets the *Label* for the above object in the View
- **SetViewSequence** Sets the *Sequence* for the above object in the View
- **SetViewObjectType** Sets the *Object type* for the above object in the View
- **SetViewStartCond** Sets the *Start Condition* for the above object in the View (***)
- **SetViewExitCond** Sets the *Exit Condition* for the above object in the View (***)
- **SetViewRole** Sets the *Role* (swimlane) for the above object in the View (***)

View – Type property

- **AddViewName** Creates a *View (or updates it, searching on its name)*
- **SetViewBID** Sets the *Business Identifier* for the View
  Or
- **AddViewBID** Creates a *View (or updates it, searching on its BID)*
- **SetViewName** Sets the *Name* of the View

- **SetViewClass** Sets the current *Class* for the following Attributes/Operations (**)
  Or
- **SetViewAssoc** Sets the current *Association* for the following Attribute/Operations (**)

- **AddViewAtt** Defines the *Attribute* of the current *Class* as part of the View (**)
  Or
- **AddViewOper** Defines the *Operation* of the current *Class* as part of the View (**)

- **SetViewLabel** Sets the *Label* for the above object in the View
- **SetViewSequence** Sets the *Sequence* for the above object in the View
- **SetViewObjectType** Sets the *Object type* for the above object in the View (***)

View - View

- **AddViewName** Creates a *View (or updates it, searching on its name)*
- **SetViewBID** Sets the *Business Identifier* for the View
  Or
- **AddViewBID** Creates a *View (or updates it, searching on its BID)*
- **SetViewName** Sets the *Name* of the View

- **AddViewView** Defines the *View* as part of the View (**)

- **SetViewLabel** Sets the *Label* for the above object in the View
- **SetViewSequence** Sets the *Sequence* for the above object in the View
- **SetViewObjectType** Sets the *Object type* for the above object in the View (***)

View Link (***)

- **AddViewName** Creates a *View (or updates it, searching on its name)*
- **SetViewBID** Sets the *Business Identifier* for the View
  Or
- **AddViewBID** Creates a *View (or updates it, searching on its BID)*
- **SetViewName** Sets the *Name* of the View

- **AddViewLink** Creates a link between two Objects in a *View*

- **SetViewFromObjectType** Sets the first part of the *From* identification
- **SetViewFromClass** Sets the second part of the *From* identification (**)
- **SetViewFromLabel** Sets the third part of the *From* identification
- **SetViewFromRole** Sets the fourth part of the *From* identification

- **SetViewToObjectType** Sets the first part of the *To* identification
- **SetViewToClass** Sets the second part of the *To* identification (**)
- **SetViewToLabel** Sets the third part of the *To* identification
- **SetViewToRole** Sets the fourth part of the *To* identification
-

- **SetViewOperClass** *Defines* the *Class* of the Operation invoked by the View link (**)
- **SetViewOper** *Defines* the *Operation* (Message) invoked by the View link (**)
- **SetViewLinkType** Sets the *Link Type* of the View link
- **SetViewLabel** Sets the *Label* property for the View link
- **SetViewSequence** Sets the *Sequence* property of the View link

Type (Class and Association Class)

- **AddClassName** Creates a *Class (or updates it, searching on its name)*
- **SetClassBID** Sets the *Business Identifier* for the Class
  Or
- **AddClassBID** Creates a *Class (or updates it, searching on its BID)*
- **SetClassName** Sets the *Name* of the Class

- **SetClassType** Sets the View *type* (“Type”, “Interface”, ”Atomic Subject Area”, …)
- **SetClassDoc** Fills the *Documentation* property of the Class (*)
- **SetClassComment** Sets the *Comment* property of the Class (*)
- **SetClassExpl** Fills the *Examples* property of the Class (*)
- **SetClassStereotype** Sets the *Stereotype* of the Class
- **SetClassAccess** Sets the *Access specifier* of the Class (Public, Private, Protected)
- **SetClassAbstract** Sets the *Abstract* flag for the Class to True/False
- **SetClassCategory** Adds the Class into the *Category* (Package) (**)
- **SetClassInheritFrom** Adds a *Dependency* to the parent class (**)
- **SetClassFilter** Sets the *Export Filter* of the Class
- **SetClassCreateDate** Sets the *Creation date* of the Class (yyyy-mm-dd hh:mm)
- **SetClassUpdateDate** Sets the last *Modification date* of the Class

Association

- **AddAssocName** Creates an *Association* From the current Class to the Class specified with the SetAssocRole2Type *(or updates it, searching on its name)*
- **SetAssocBID** Sets the *Business Identifier* for the Association
  Or
- **AddAssocBID** Creates an *Association* From the current Class to the Class specified with the SetAssocRole2Type *(or updates it, searching on its BID)*
- **SetAssocName** Sets the *Name* of the Association

- **SetAssocType** Sets the View *type* (“Association”, ”Relationship”, …)
- **SetAssocCategory** Adds the *Association* into the *Category* (Package) (**)
- **SetAssocInheritFrom** Adds a *Dependency* to the parent class (**)
- **SetAssocStereotype** Sets the *Stereotype* of the Association
- **SetAssocAccess** Sets the *Access specifier* of the Association
- **SetAssocAbstract** Sets the *Abstract* flag for the Association to True/False
- **SetAssocDoc** Sets the *Documentation* property for the Association (*)
- **SetAssocComment** Sets the *Comment* property of the Association (*)
- **SetAssocExpl** Sets the *Examples* property for the Association (*)
- **SetAssocFilter** Sets the *Export Filter* of the Association
- **SetAssocRole1Type** Used to define the *Start Class* of the Association (**)
- **SetAssocRole1Name** Sets the name for *Role A*
- **SetAssocRole1Card** Sets the *Cardinality* for the Role A Detail
- **SetAssocRole1Navi** Sets the *Navigability* flag for the Role A to True/False
- **SetAssocRole1Aggr** Sets the *Aggregate* flag for the Role A to True/False
- **SetAssocRole1Cont** Sets the *Containment* value for the Role A
- **Set AssocRole1Stat** Sets the *Static* flag for the Role A to True/False
- **SetAssocRole2Type** Used to define the *End Class* of the Association (**)
- **SetAssocRole2Name** Sets the name for *Role B*
- **SetAssocRole2Card** Sets the *Cardinality* for the Role B Detail
- **SetAssocRole2Navi** Sets the *Navigability* flag for the Role B to True/False
- **SetAssocRole2Aggr** Sets the *Aggregate* flag for the Role B to True/False
- **SetAssocRole2Cont** Sets the *Containment* value for the Role B
- **Set AssocRole2Stat** Sets the *Static* flag for the Role B to True/False
- **SetAssocCreateDate** Sets the *Creation date* of the Association (yyyy-mm-dd hh:mm)
- **SetAssocUpdateDate** Sets the last *Modification date* of the Association
- **EndAssocName** marks the end of the association block Or
- **EndAssocBID** marks the end of the association block

Attribute

- **AddClassName** Specifies the current *Class* to which the attribute(s) belongs
- **AddAttName** Creates an *Attribute* to the current class (*or updates it, searching on its name*)
- **SetAttBID** Sets the *Business Identifier* for the Attribute
  Or
- **AddClassBID** Specifies the current *Class* to which the attribute(s) belong(s)
- **AddAttBID** Creates an *Attribute* to the current class *(or updates it, searching on its BID)*
- **SetAttName** Sets the *Name* of the Attribute

- **SetAttType** Sets the Attribute *type* (“Attribute”, ”Atomic Data Element”, …)
- **SetAttSequence** Sets the *Sequence* property of the Attribute
- **SetAttDoc** Sets the *Documentation* property of the Attribute (*)
- **SetAttExpl** Sets the *Examples* property of the Attribute (*)
- **SetAttAnExpl** Sets one single *Example* property of the Attribute (*)
- **SetAttComment** Sets the *Comment* property of the Attribute (*)
- **SetAttFilter** Sets the *Export Filter* of the Attribute
- **SetAttStereotype** Sets the *Stereotype* of the Attribute
- **SetAttAccess** Sets the *Access specifier* of the Attribute (Public, Private, Protected)
- **SetAttStatic** Sets the *Static* flag for the Attribute to True/False
- **SetAttOptional** Sets the *Optional* flag for the Attribute to True/False
- **SetAttDomainType** Sets the *Attribute primitive (MMM) Data Type* property (**)
- **SetAttReturnType** Sets the *Attribute Class Data Type* property (**)
- **SetAttIsPK** Sets the *Attribute Is Primary Key* flag to True/False
- **SetAttIsFK** Sets the *Attribute Is Foreign Key* flag to True/False
- **SetAttCreateDate** Sets the *Creation date* of the Attribute (yyyy-mm-dd hh:mm)
- **SetAttUpdateDate** Sets the last *Modification date* of the Attribute

Operation

- **AddClassName** Specifies the current *Class* to which the Operation(s) belongs
- **AddOperName** Creates a *Operation* to the current class (*or updates it, searching on its name*)
- **SetOperBID** Sets the *Business Identifier* for the Operation
  Or
- **AddClassBID** Specifies the current *Class* to which the Operation(s) belongs
- **AddOperBID** Creates a *Operation* to the current class *(or updates it, searching on its BID)*
- **SetOperName** Sets the *Name* of the Operation

- **SetOperSequence** Sets the *Sequence* property of the Operation
- **SetOperDoc** Sets the *Documentation* property of the Operation (*)
- **SetOperComment** Sets the *Comment* property of the Operation (*)
- **SetOperExpl** Sets the *Examples* property of the Operation (*)
- **SetOperAnExpl** Sets one single *Example* property of the Operation (*)
- **SetOperFilter** Sets the *Export Filter* of the Operation
- **SetOperReturnType** Sets the *Return Type* of the Operation (**)
- **SetOperParameters** Creates the *Parameters* for the current Operation (as a text string: “[in] aParam1 Type1, [out] aParam2 Type2, [in out] aParam3 Type3, …”)
- **SetOperStereotype** Sets the *Stereotype* of the Operation
- **SetOperAccess** Sets the *Access specifier* of the Operation (Public, Private, Protected)
- **SetOperCreateDate** Sets the *Creation date* of the Operation (yyyy-mm-dd hh:mm)
- **SetOperUpdateDate** Sets the last *Modification date* of the Operation

Package / View / Type / Association Mapping (xxx = **Pack** / **View** / **Class / Assoc**)

- **SetModelMapping** Specifies a *Source Model* to which the mapping refers (****)

- **Add**xxx**Name** Specifies an *object,* by its Name, to which the mapping refers

Or

- **Add**xxx**BID** Specifies an *object*, by its BID, to which the mapping refers

- **Add**xxx**Mapping** Creates a *mapping for* the current object (**)
- **Set**xxx**MappingInheritDef** Sets the *Append Definition* flag to True/False
- **Set**xxx**MappingInheritExpl** Sets the *Append Examples* flag to True/False
- **Set**xxx**MappingNavigPath** Specifies the *Navigation Path*
- **Set**xxx**MappingTransRules** Specifies the *Transformation Rules*
- **Set**xxx**MappingNature** Specifies the *Nature* of the mapping

Note: Mapping SRI is not supported by Import/Export SRI for RSA, except for the View mapping which is used to transfer IDM Component dependencies.

Attribute / Operation Mapping (xxx = **Class/Assoc**, yyy = **Att / Oper**)

- **SetModelMapping** Specifies a *Source Model* to which the mapping refers (****)

- **Add**xxx**Name** Specifies a *Class*, by its Name, to which the mapping refers
- **Add**yyy**Name** Specifies an *Attribute/*Operation to which the mapping refers

Or

- **Add**xxx**BID** Specifies a *Class*, by its BID, to which the mapping refers
- **Add**yyy**BID** Specifies an *Attribute/Operation* to which the mapping refers

- **Set**xxx**Mapping** Specifies a *Class mapping for* the current Attribute (**)
- **Add**yyy**Mapping** Creates an A*ttribute/Operation mapping for* the current Attribute (**)
- **Set**xxx**MappingInheritDef** Sets the *Append Definition* flag to True/False
- **Set**xxx**MappingInheritExpl** Sets the *Append Examples* flag to True/False
- **Set**xxx**MappingInheritDom** Sets the *Inherit Domain* flag to True/False
- **Set**xxx**MappingNavigPath** Specifies the *Navigation Path*
- **Set**xxx**MappingTransRules** Specifies the *Transformation Rules*
- **Set**xxx**MappingNature** Specifies the *Nature* of the mapping

Type to Attribute / Operation Mapping (xxx = **Class**, yyy = **Att / Oper**)

- **SetModelMapping** Specifies a *Source Model* to which the mapping refers (****)

- **Add**xxx**Name** Specifies a *Class*, by its Name, to which the mapping refers

Or

- **Add**xxx**BID** Specifies a *Class*, by its BID, to which the mapping refers

- **Set**xxx**Mapping** Specifies a *Class mapping* for the current Attribute (**)
- **Add**yyy**Mapping** Creates an A*ttribute/Operation mapping* for the current Attribute (**)
- **Set**xxx**MappingNavigPath** Specifies the *Navigation Path*
- **Set**xxx**MappingTransRules** Specifies the *Transformation Rules*
- **Set**xxx**MappingNature** Specifies the *Nature* of the mapping

(****) the Model Short Name for inter-model mapping, or “<itself>” for intra-model mapping

Attribute/Operation to Type Mapping (xxx = **Class/Assoc**, yyy = **Att/Oper**)

- **SetModelMapping** Specifies a *Source Model* to which the mapping refers (****)

- **Add**xxx**Name** Specifies a *Class*, by its Name, to which the mapping refers
- **Add**yyy**Name** Specifies an *Attribute/*Operation to which the mapping refers

Or

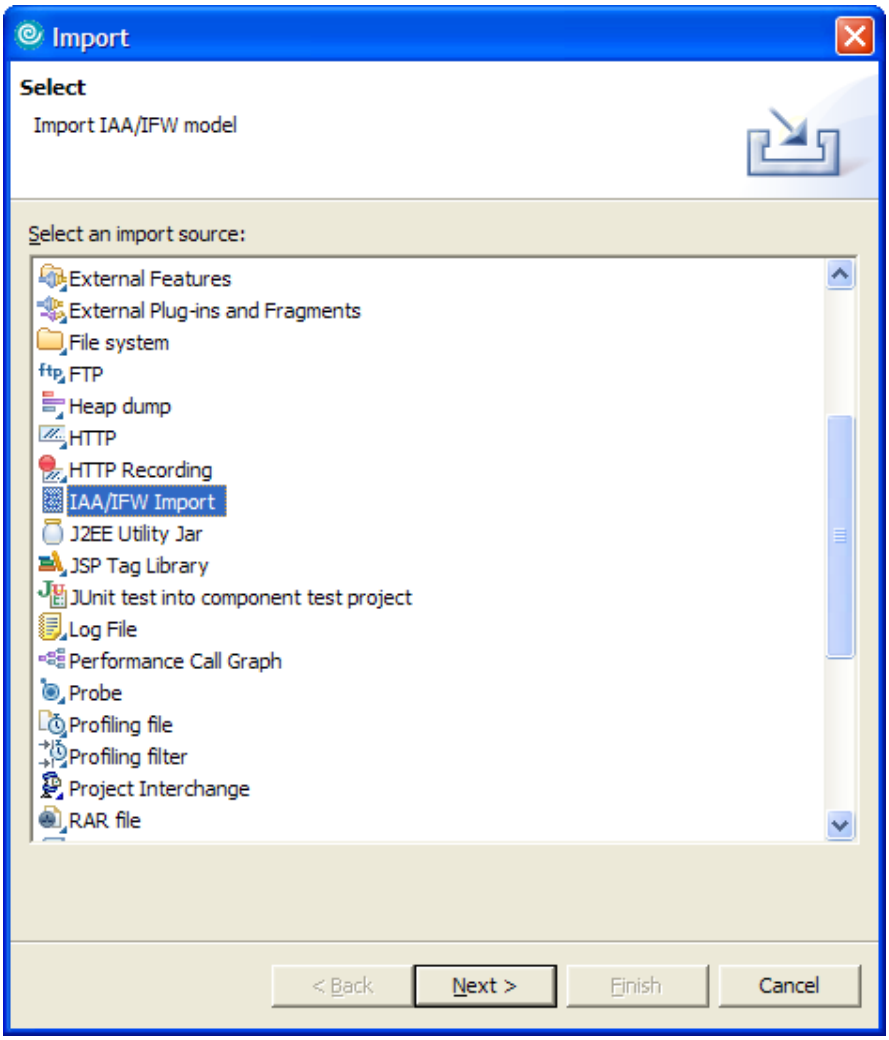
- **Add**xxx**BID** Specifies a *Class*, by its BID, to which the mapping refers
- **Add**yyy**BID** Specifies an *Attribute/Operation* to which the mapping refers

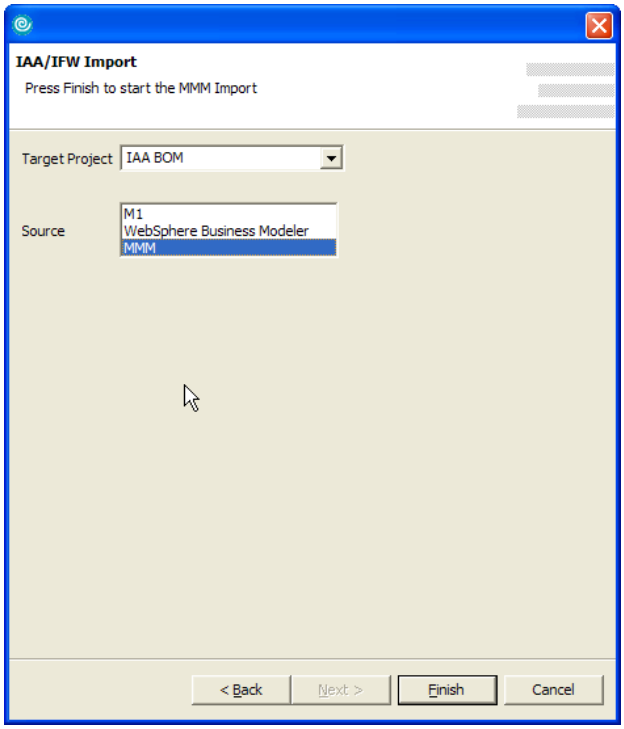
- **Add**xxx**Mapping** Specifies a *Class mapping* for the current Attribute (**)
- **Set**xxx**MappingNavigPath** Specifies the *Navigation Path*
- **Set**xxx**MappingTransRules** Specifies the *Transformation Rules*
- **Set**xxx**MappingNature** Specifies the *Nature* of the mapping

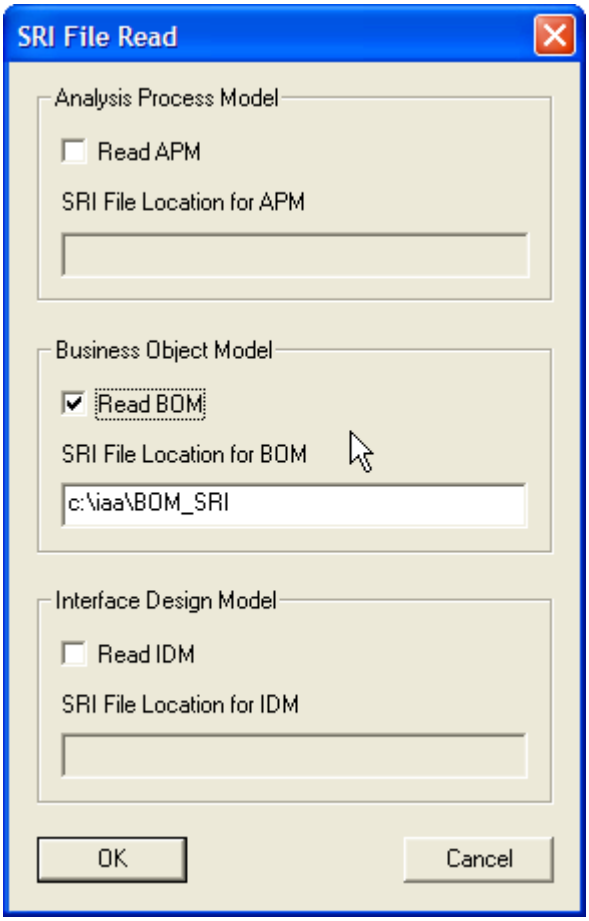
6.5 Import / Export Logical Model – RSA plugin

The SRI files are used to exchange the model content with Rational Software Architect (RSA / RSM) (and other CASE tools such as Rational Rose, Embarcadero Describe® and ER/Studio® that could implement similar bridges).
A set of RSA functions have been developed to import and export SRI files within a RSA model.

From the RSA, select File -> Import (or Export), then select the IAA/IFW Import (or Export) wizard:







Note that the directory should contains the set of SRI files, named as follow: Packages.sri, Associations.sri, Types.sri, Properties.sri, Mappings.sri

6.6 Import / Export Data Model – IDA plugin

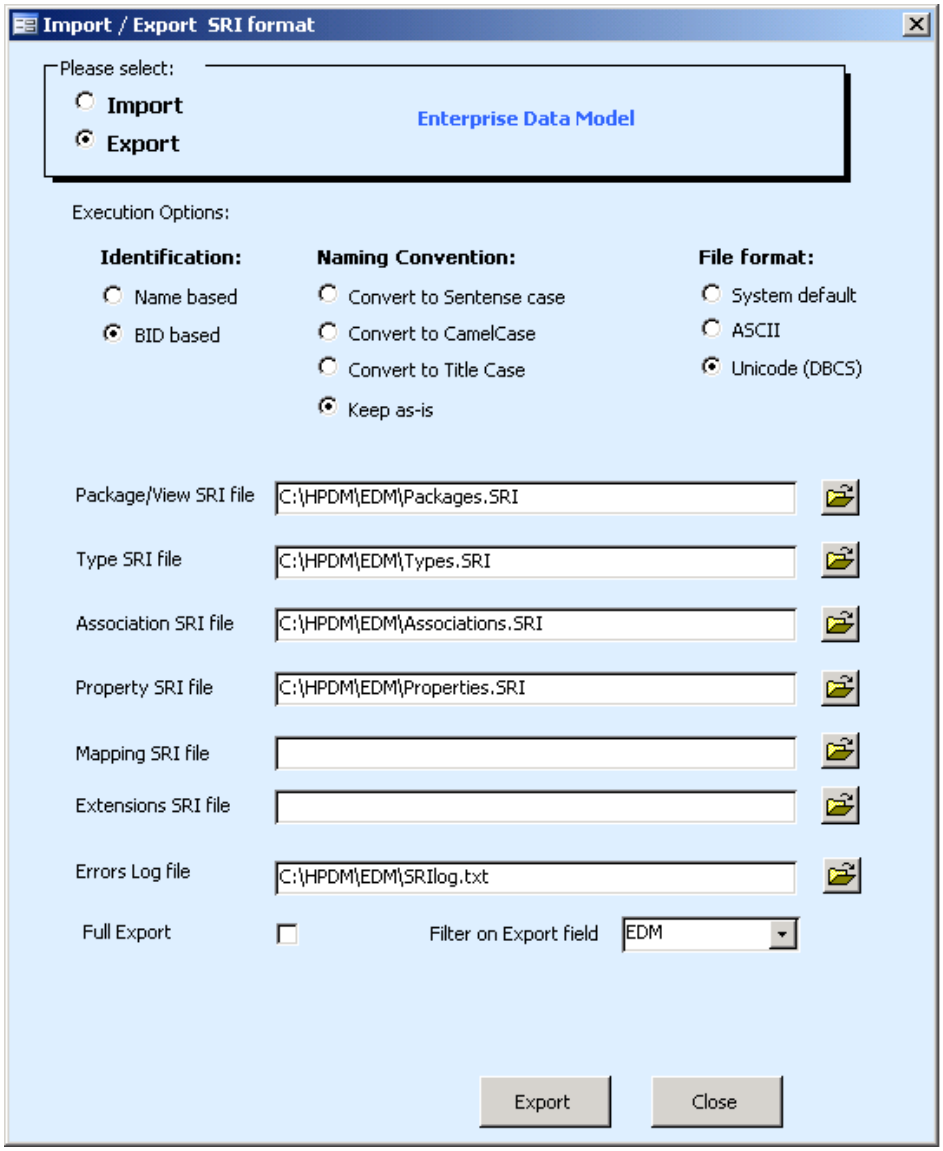
An Industry Models plugin enables the exchange of data model content between SRI files and IBM InfoSphere Data Architect (IDA).
SRI files containing a full data model can be imported and exported to and from an IDA logical data model.

For the Health Plan Data Model (HPDM), the following model imports are supported:

- Enterprise Data Model
- Core Warehouse Model
- Conformed Dimension Model

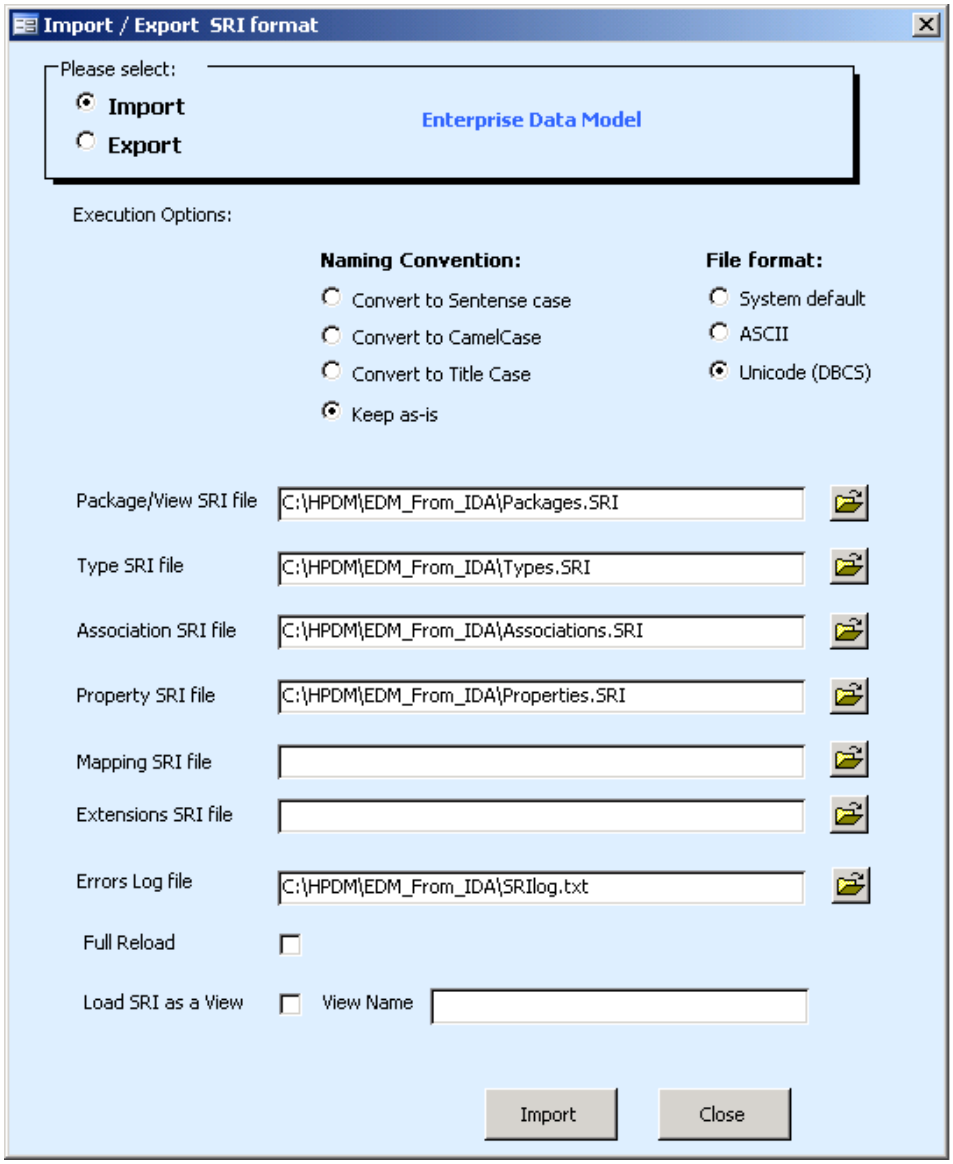
The HPDM Enterprise Data Model is used as an example to demonstrate how to exchange a HPDM data model between MMM and IDA.

When exporting from MMM to SRI files use the following options:



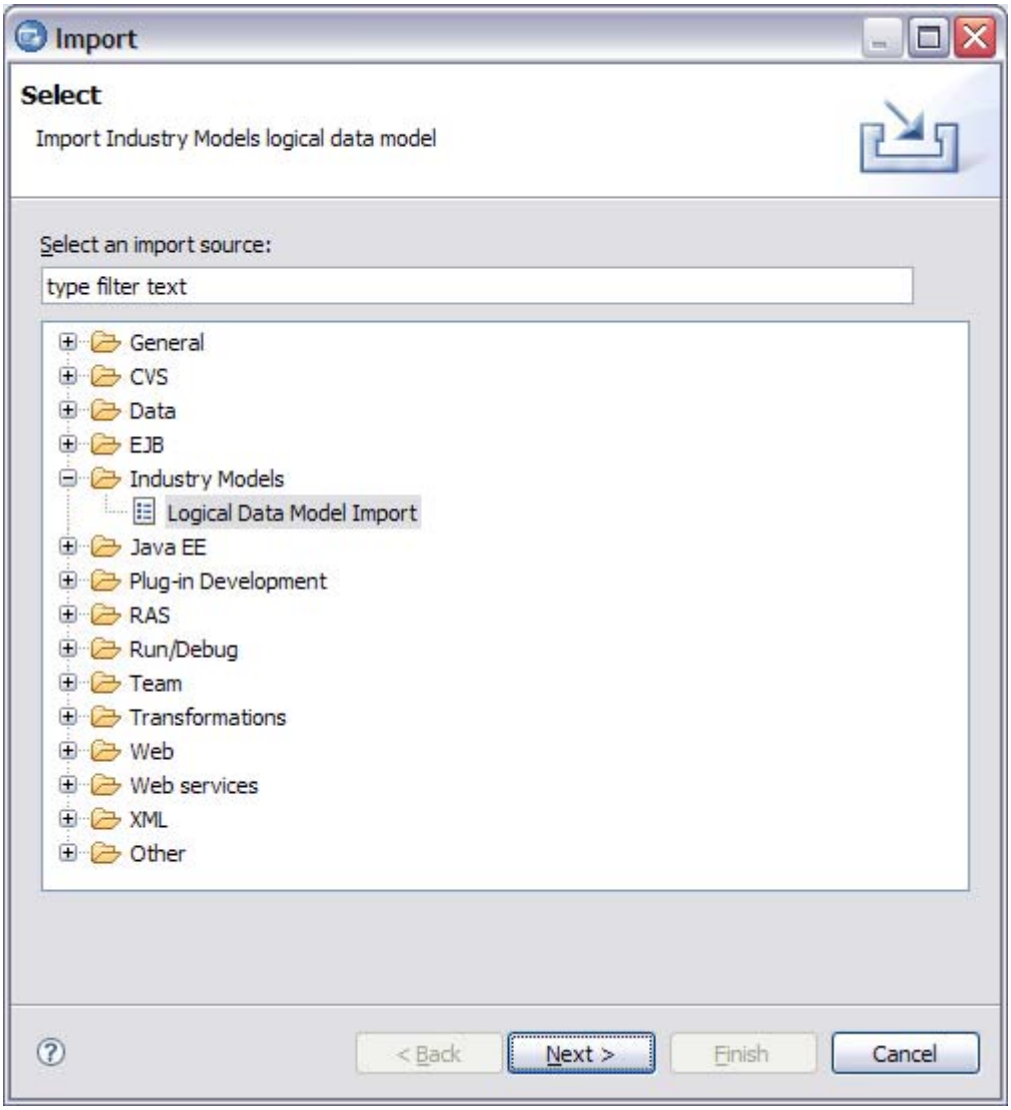
The IDA Bridge does not support the Mapping or Extensions SRI files, so these two options can be avoided. For the HPDM Enterprise Data Model the **Filter on Export field** option should be **EDM.** For the other model exports the option for **Filter on Export field** is **All**.

When importing a round tripped model back into MMM, use the following options:

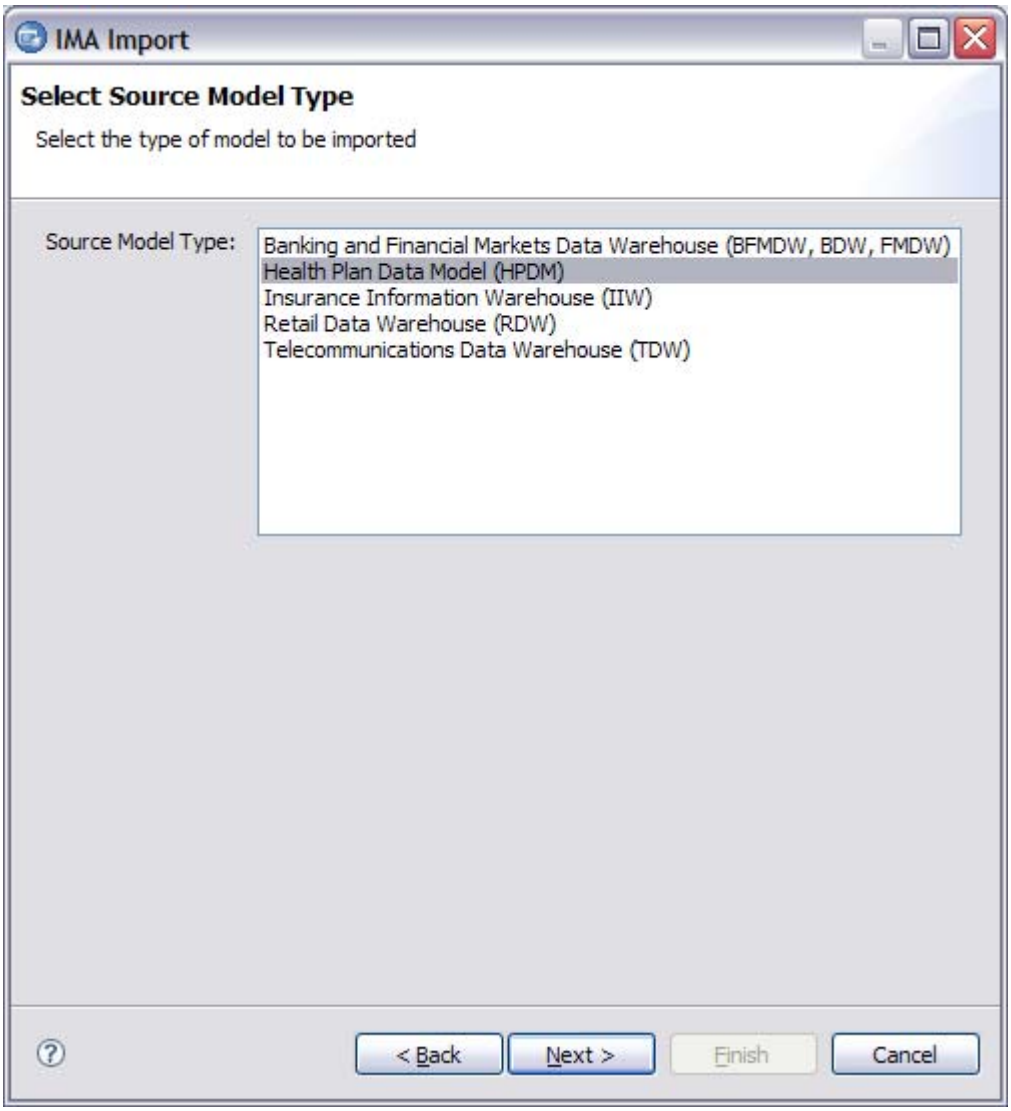


To import the SRI files into IDA:

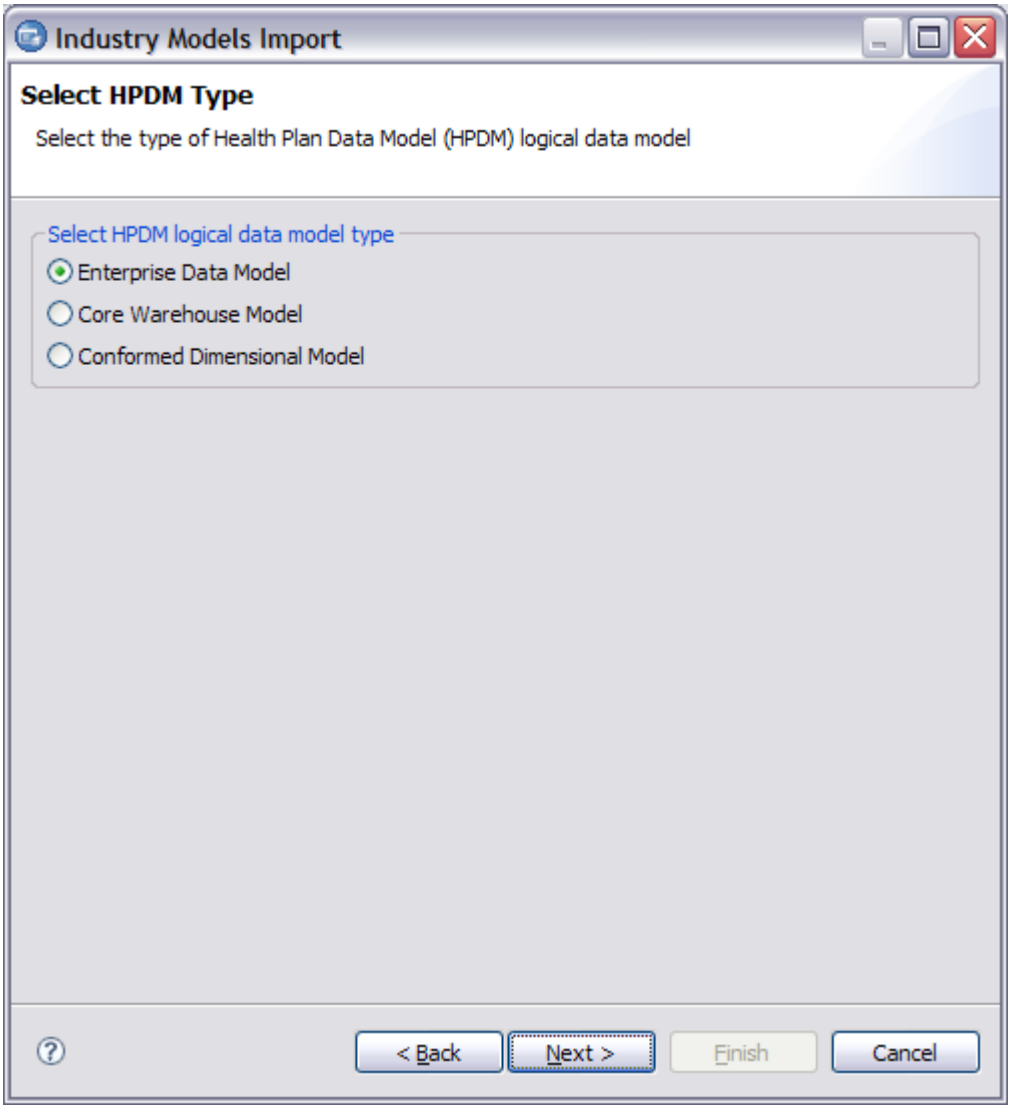
- Select the **File -> Import** from the IDA menu and choose **Industry Models -> Logical Data Model Import**.



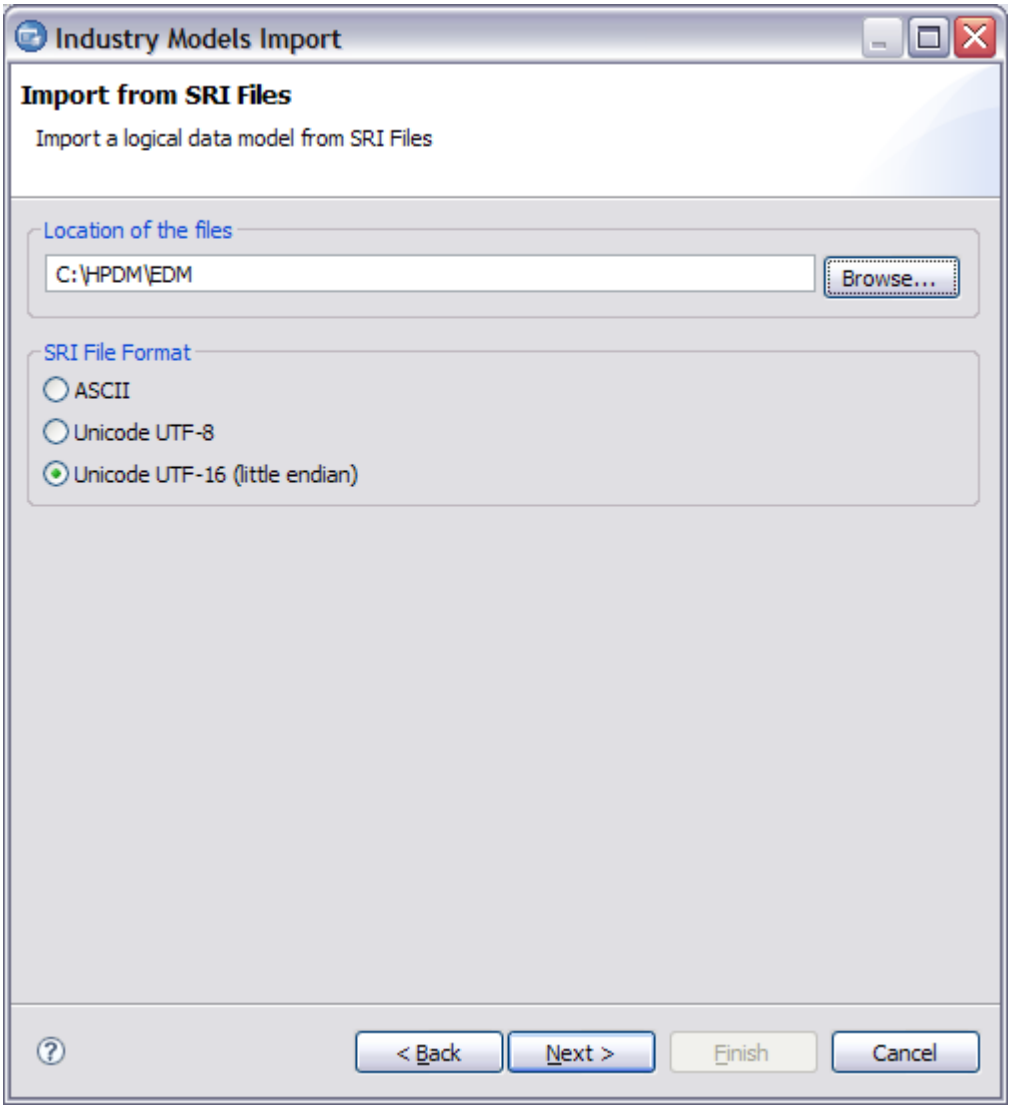
- Select **Health Plan Data Model (HPDM)** in the *Select Source Model Type* wizard page.



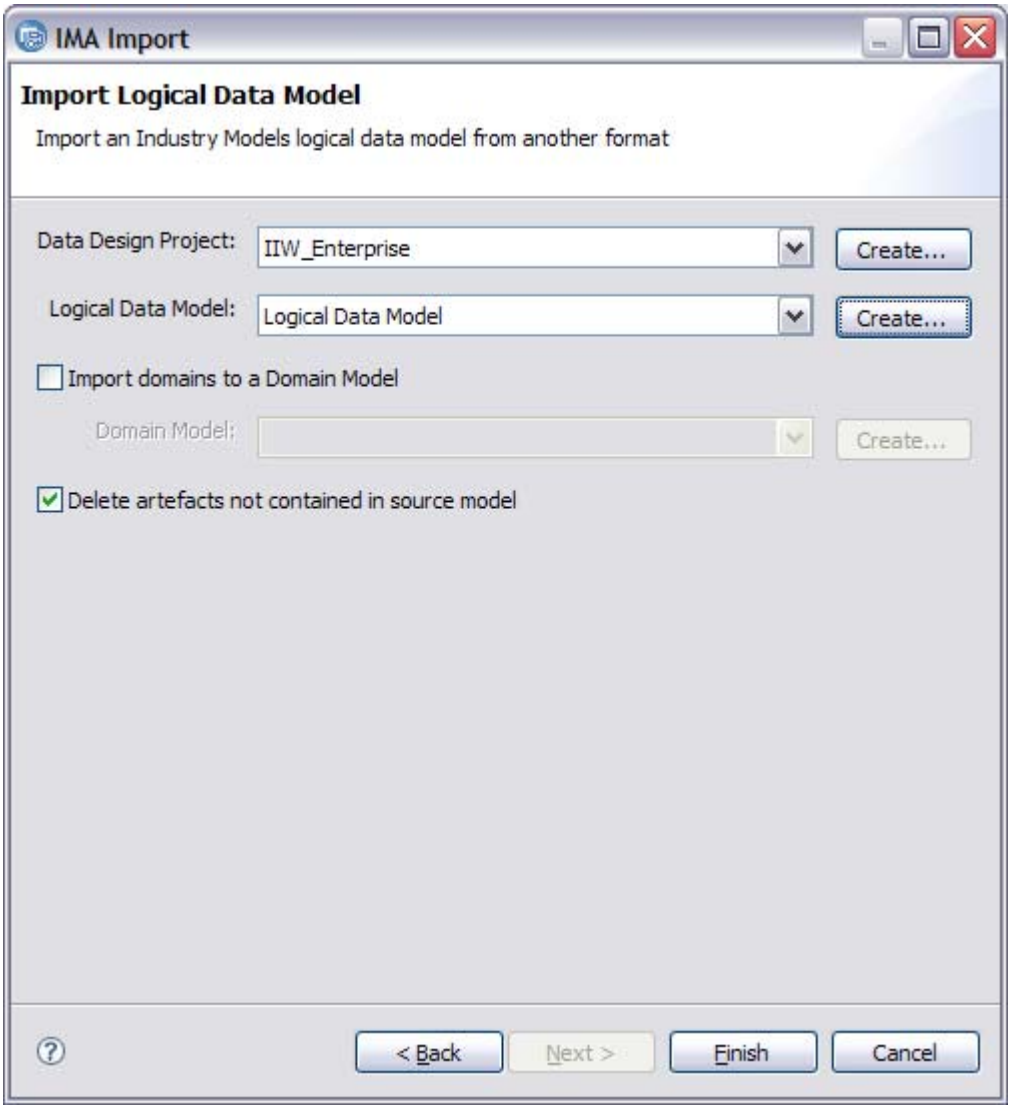
- Select the appropriate type in the Select HPDM Type wizard page (SRI files do not contain model information; hence this option):



- Browse to the location of the SRI files in the *Import from SRI Files* wizard page. Select UTF-16 as the SRI file format:



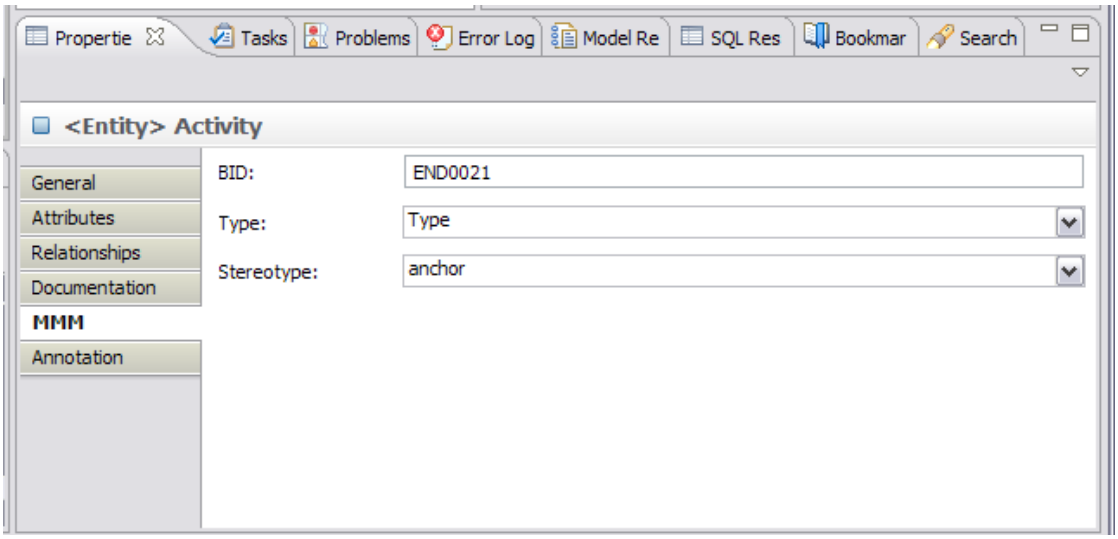
- Either use the drop down boxes to select an existing Data Design Project and Logical Data Model or else use the Create buttons to launch the IDA wizards to create a new data design project and a new logical data model. You can optionally import domains to a separate Domain Model which can also be an existing domain model or can be newly created. If you don't choose a separate domain model then domains will be imported as part of the logical data model.



The import function of the MMM-IDA bridge operates in two modes. In the first mode, model content can be imported into an empty logical data model. In the second mode of operation, the MMM-IDA bridge can be used to overlay source content with an existing logical data model. In this mode a matching algorithm is used to update/delete model content.

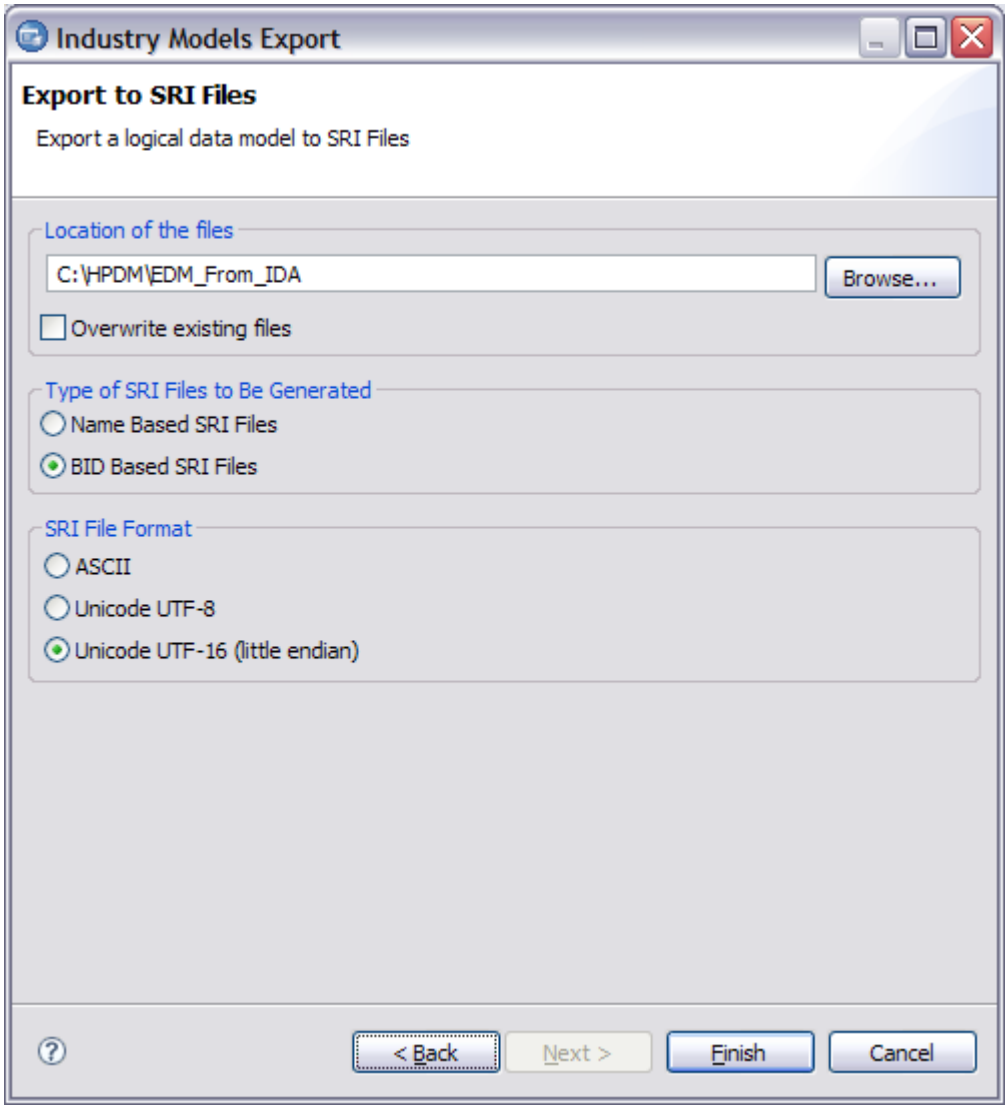
Delete artefacts not contained in source model option is used when importing on top of an existing model. When checked, objects in the existing model that are not in the SRI files being imported are deleted.

The IDA Bridge extends the IDA properties view to add a new MMM tab.



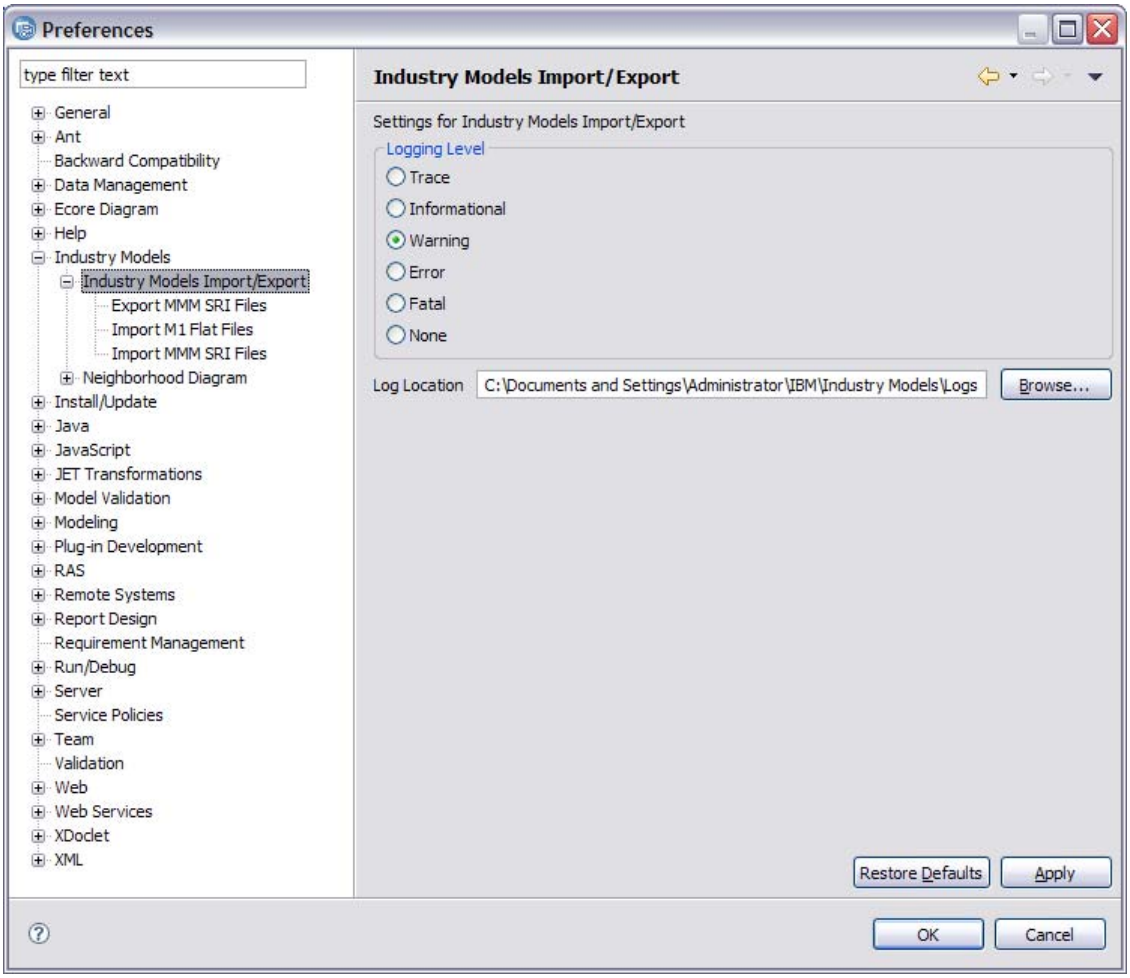
For newly created objects in IDA, the user should specify appropriate Type and Stereotype information. The combo boxes have context sensitive types and stereotypes preloaded. If the user does not set Type or Stereotype information, warnings will be issued on IDA Bridge export and default values will be used.

The IDA Bridge export wizard pages are similar to the import wizard pages. It is advisable to specify *BID Based SRI Files* on the *Export to SRI Files* wizard page.



IDA Bridge Preferences

The IDA Bridge preferences can be set by selecting the *Windows -> Preferences -> Industry Models Import/Export* menu item.

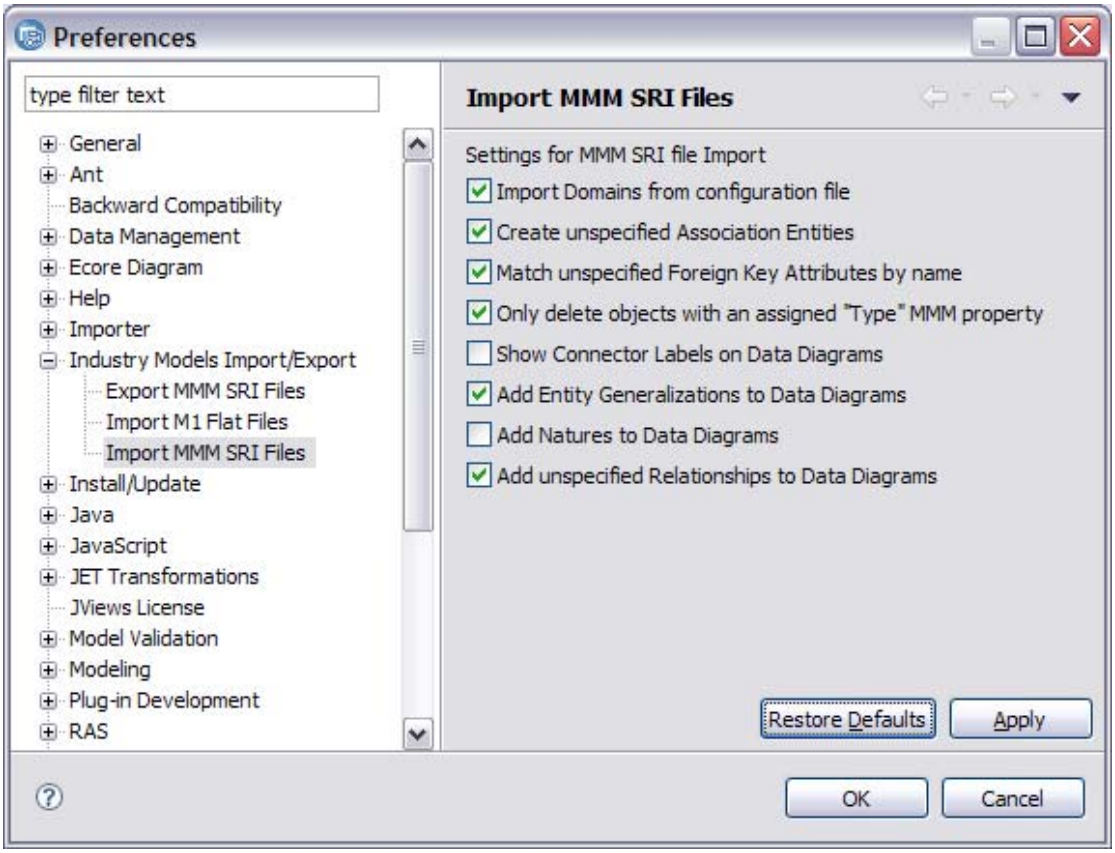


The Logging Level preference determines how much info is logged. The default level is **Warning**.

Depending on the Logging Level preference, the IDA Bridge may write to the log file IM_Tools.log in the directory specified in “Log Location”. The default is the “IBM\Industry Models\Logs” folder in the users home directory (e.g C:\Documents and Settings\Administrator\IBM\Industry Models\Logs).

Import Preferences

Preferences for importing MMM SRI files can be set by selecting *Windows -> Preferences -> Industry Models Import/Export -> Import MMM SRI Files*.



The *Import Domains from configuration file* preference controls whether the IDA Bridge should create atomic domains in the LDM being imported. If you chose to import the domains to the logical data model instead of a separate domain model then they will be imported to a package named **<< HPDM Domains>>** in the imported LDM.

When checked, the domains specified in *HPDM_Domains.xml* are used to create atomic domains. This XML file can be found in the installation folder for the *com.ibm.ima.imardasritools* plugin (e.g. *C:\ProgramFiles\IBM\SDP75Shared\plugins\com.ibm.ima.imardasritools_3.2.0.v20090316\config*).

When unchecked, domain types are mapped to an IDA Predefined Data Type using the mappings specified in the *Map_HPDM_Datatypes.properties* file (which can be found in the same location as HPDM_Domains.xml). Note that models created with this option are probably not suitable for exporting back out to MMM as domain information will be different.

The *Create unspecified Association Entities* preference controls whether an Association Entity is created when the Associations.sri file contains a definition for an “Association” relationship, but there’s no matching entry in the Types.sri file.

The *Match unspecified Foreign Key Attributes by name* controls whether the MMM-IDA bridge should attempt to match foreign key attributes not explicitly defined in the SRI files being imported (attributes are matched by name and domain type).

The *Only delete objects with an assigned "Type" MMM property* preference controls which objects are deleted from the destination model during an overlay import. When this option is selected, objects that have an empty MMM "Type" property are not deleted. The exception are Packages – they are deleted if they are not in the source model regardless of whether "Type" is empty (this is because the Type property cannot be assigned to Packages in MMM)

The *Show Connector Labels on Data Diagrams* preference controls whether connector names are added to relationship edges in imported diagrams.

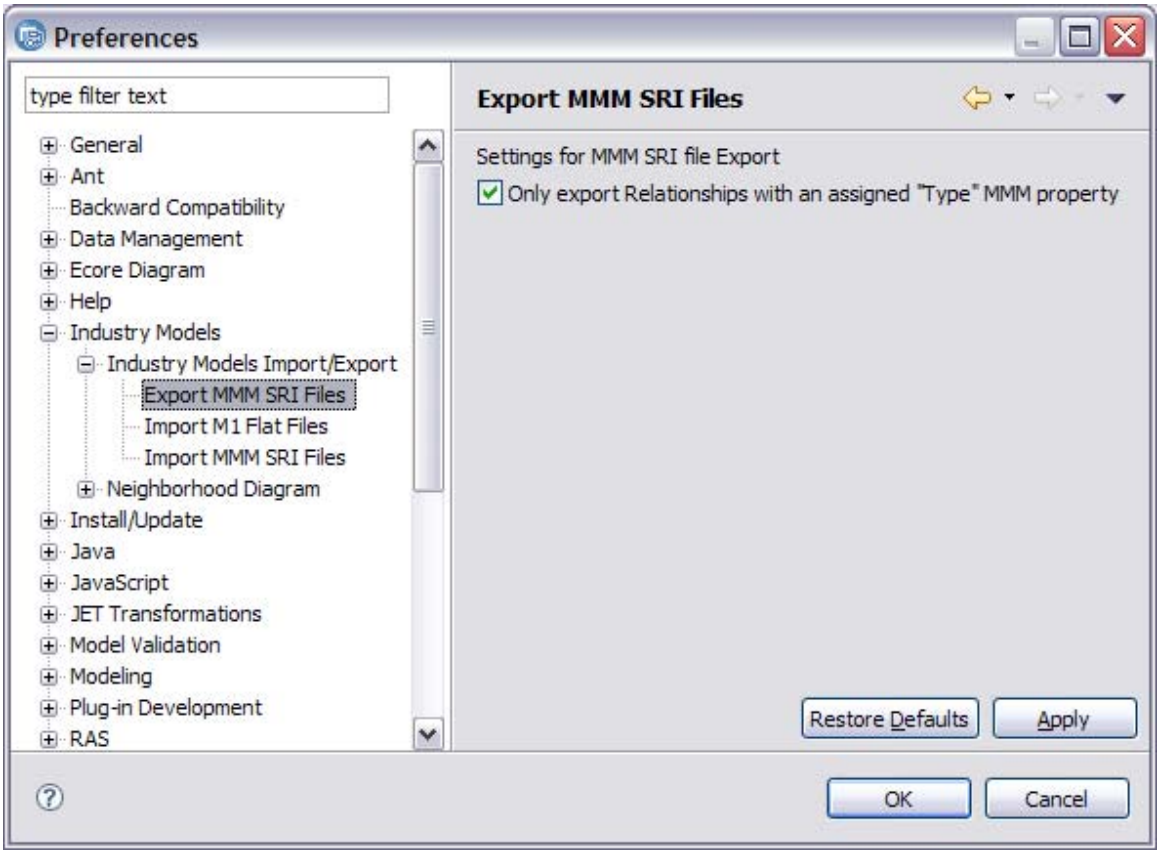
The *Add Entity Generalizations to Data Diagrams* preference controls whether generalization edges are created in imported diagrams. The SRI files define entity generalizations but do not explicitly specify which generalizations should be added to which diagrams.

The *Add Natures to Data Diagrams* controls whether relationship edges for Associative Entities that don't have supertypes are added to data diagrams.

The *Add unspecified Entities to Data Diagrams* preference controls whether entities that are not explicitly defined as being part of a diagram in the SRI files being imported, but that are referenced by relationships on the diagram, should be added to the diagram.

Export preferences

Preferences for exporting MMM SRI files can be set by selecting *Windows -> Preferences -> Industry Models Import/Export -> Export MMM SRI Files*.



The *Only export Relationships with an assigned "Type" MMM property* preference controls which relationships are exported to SRI files. When this option is selected, relationships that have an empty MMM "Type" property are not exported to SRI files.

CHAPTER 7: IMPORT / EXPORT ERWIN®

The MMM supports ERwin® 3.52 and ERwin® 4.1 and Erwin® 7 bridges for exchanging textual content (Names, Definitions, Examples, data type, BID, mapping information.)
This makes possible to synchronize the Entities/Attributes in ERwin® with the MMM models that use the inheritance of definitions based on the traceability: one change in the definition at Business level can be spanned down to Enterprise and datamart models.
The export function will add or update the object without altering the diagrams.
The identification can be either Name- or BID-based.

The ERwin® 3.5.2 bridge is a one-way export capability from MMM to an .ER1 file.
The ERwin® 4.1 and Erwin 7 bridges are two-ways import/export capability using the ERwin® XML export format.

7.1 Import ERwin®

From ERwin® v 4.1 or v 7
The Import into MMM reads an existing **.XML** model.
Note: The XML format of an ERwin® model is obtained by a "File" - "Save as…" ERwin® function. Choose either the Standard XML Format or the Standard XML Format with Minimum Info. Using the minimum format is faster but doesn't import derived documentation. The standard format is usually only required after foreign keys have been added to the ERwin® model.

7.2 Export ERwin®

To ERwin® v 3.5.2
The Export from MMM creates or updates an existing **.ER1** model.
Any newly created entity will appear in the top-left corner of the <Main Subject Area> in ERwin®.

Note: For performing the "Export ERwin 3.5.2" function, the ERwin® API (er2api32.dll) is needed. It is available with ERwin® 3.5.2 SP2 or higher.

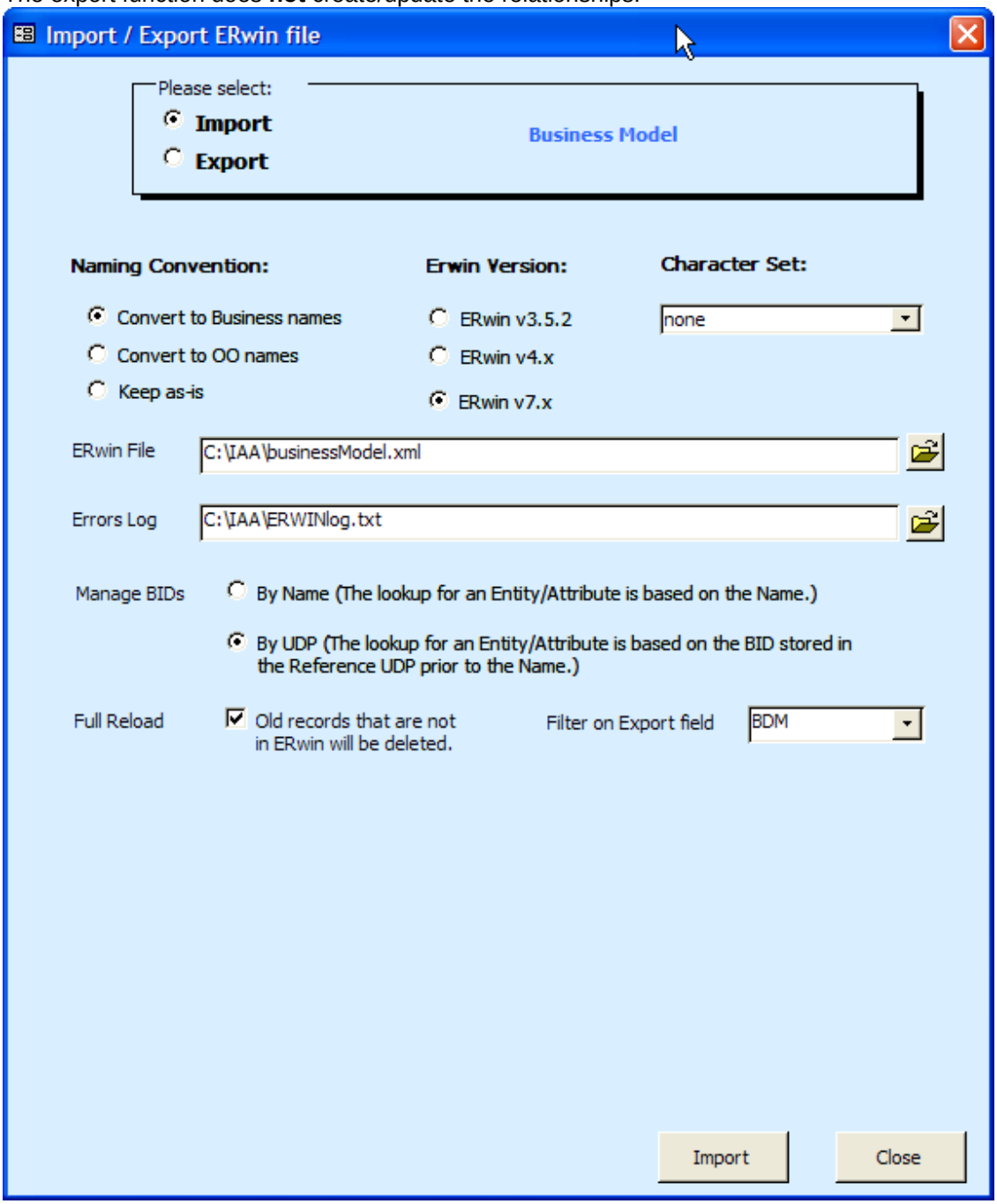
To ERwin® v 4.1 or v 7
The Export from MMM creates or updates an existing **.XML** model.
From a provided Xxxx.XML existing file, MMM will create a Xxxx_Updated.XML and let the Xxxx.XML file unchanged.
Note: The XML format can be loaded in ERwin® by a "File" - "Open…" ERwin® function.

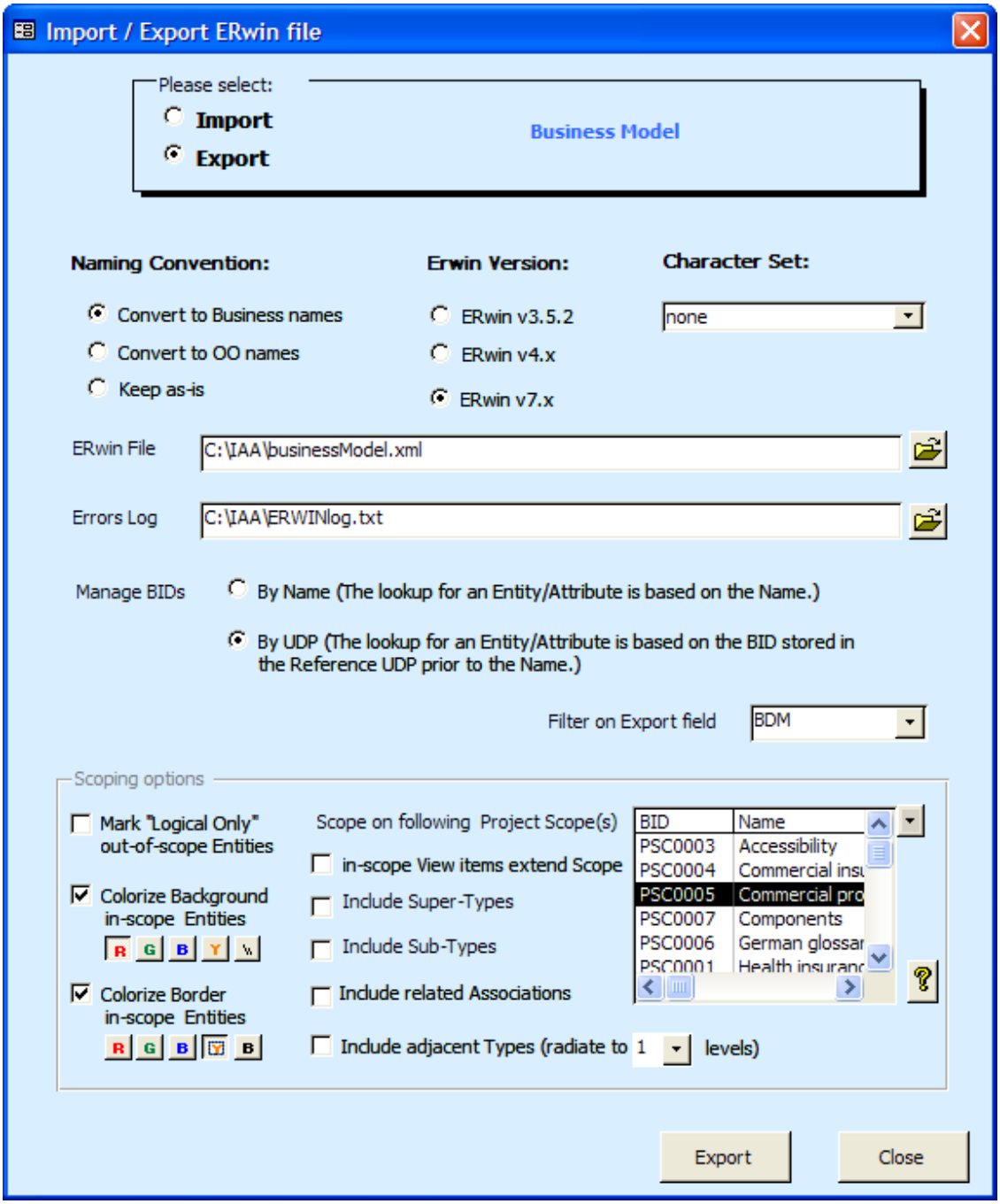
For both export versions, this is **NOT** a full export, but the purpose of this function is to update the following information into the ERwin® model for both Entities and Attributes:

- the Name
- the Definition (either specific or inherited)

- the Examples (either specific or inherited)
- the BID (“Reference” UDP)
- the Sequence
- the Mapping (“Source Reference” UDP)
- the Domains (v4.1 only)

The import function also imports the domains and relationships.
The export function does **not** create/update the relationships.





Naming Convention:

Object names can optionally be converted to Business Names or to the Object Oriented Syntax (see *Standards and Naming Conventions* Appendix)

Manage BIDs

If **By name** is selected, the program will try to find an entity or an attribute based on its name. If it does not find it, the program will create a new entity / attribute with this name. (Be especially careful if renaming has been done in the MMM !)

If **By UDP** is selected, the program will first try to find an entity or an attribute based on the BID that is stored in the "Reference" User-defined Property (UDP). If not found, it will try to find it using the entity's name. If it is still not found, the program will create a new entity / attribute with the name.
Important: In order to use this option, the "Entity Reference", "Entity Source Reference", "Attribute Reference" and "Attribute Source Reference" UDPs must exist on Entity and Attribute ERwin® objects.

Scoping Options:

Project Scope(s): One or more Project Scope view can be selected

Mark Logical-Only: Entities and Attributes can optionally be flagged Logical-Only if they do not belong to a given Project Scope. The Physical Erwin® view will only show Tables and Columns that are in scope.

Colorize Background: Entities can optionally be colorized (Red or Green or Blue or Yellow or reset to White) if they belong to a given Project Scope.
Entities for which only a few Attributes are in scope are differentiated by a lighter colour.

Colorize Border: Entity's borders can optionally be colorized (Red or Green or Blue or Yellow or reset to Black) if they belong to a given Project Scope.

Please refer to the ***Hints and Tips*** appendix for more explanation on how to use the *Filtering Options.*

7.3 Notes about Import options

For the Associations, there are two options when importing a model from ERwin® to MMM.

Let's take an example: PARTY as one fundamental entity, OBJECT as the second fundamental entity, and PARTY-OBJECT RLSHIP as the association between them.

When importing this in MMM we have two options:

- create one Type called "*PARTY*"
- create one Type called "*OBJECT*"
- create one Association called "*PARTY-OBJECT RLSHIP*" , with one parent on PARTY and one parent on OBJECT

but we could also do the following:

- create one Type called "*PARTY*"
- create one Type called "*OBJECT*"
- create one Type called "*PARTY-OBJECT RLSHIP*"
- create one Association called "*PARTY <role name> PARTY-OBJECT RLSHIP* ", with one parent on PARTY and one parent on PARTY-OBJECT RLSHIP
- create one Association called "*OBJECT <role name> PARTY-OBJECT RLSHIP*" , with one parent on OBJECT and one parent on PARTY-OBJECT RLSHIP

Actually, in MMM, the decision to take one option or the other is based on the role names.
If the role name is "**is parent of**", or **"is left parent of**" or "**is right parent of**", then option 1 is taken, otherwise, option 2 is taken.

There are also some role names that are not considered in MMM for creating associations since they would produce unnecessary complexity: "**is type of**" and "**is anchor of**".

Obviously, the option 1 can only be used if they are TWO and only two parents with the role name "**is parent of**".

CHAPTER 8: IMPORT / EXPORT XML

The MMM has an XML bridge for exchanging textual content (Names, Definitions, Examples, data type, BID, mapping information.)
This makes possible for instance to synchronize the Classes/Attributes in Rational-XDE with the MMM models that use the inheritance of definitions based on the traceability: one change in the definition at Business level can be spanned down to the IDM model.
The export function will add or update the object without altering the diagrams.
The identification can be either Name- or BID-based.

In order to allow different XML "dialects", a cross-reference table called *XML Templates* is used to map a XML tag to an MMM construct. By extending this table, the user can define its own mapping between any XML format and the MMM constructs he wants to map to.
Other CASE tools that have the XML import/Export capability can therefore be bridged to MMM for synchronizing the textual content.

8.1 Import XML

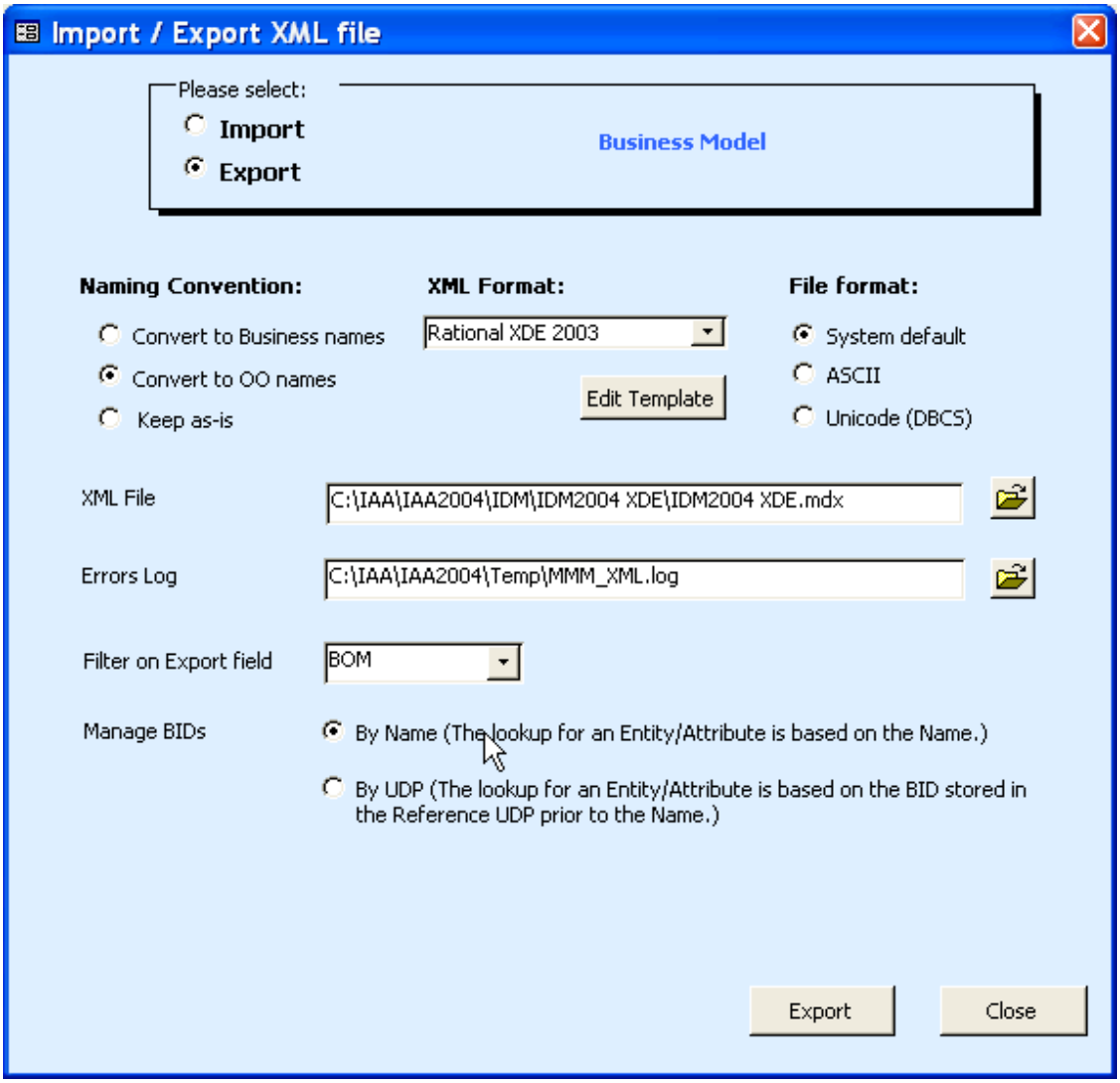
This import bridge is at BETA level, with the following capabilities and limitations:

- Imports by Name (import by BID not yet available)
- Creates / Updates Package, View, Type, Type Property
- Does not import Associations
- The XML Template for XDE currently defines tag mapping for Name, Definition, Stereotype, Parent id, Package id. (can be extended)

8.2 Export XML

This export bridge is at BETA level, with the following capabilities and limitations:

- Exports by Name (export by BID not yet available)
- Creates / Updates XML Elements and Attributes
- Creates all new elements in Root package
- The XML Template for XDE currently defines tag mapping for Name, Definition, Stereotype, Parent id, Package id. (can be extended)



Naming Convention:

Object names can optionally be converted to Business Names or to the Object Oriented Syntax (see *Standards and Naming Conventions* Appendix)

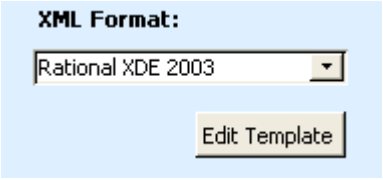
Manage BIDs

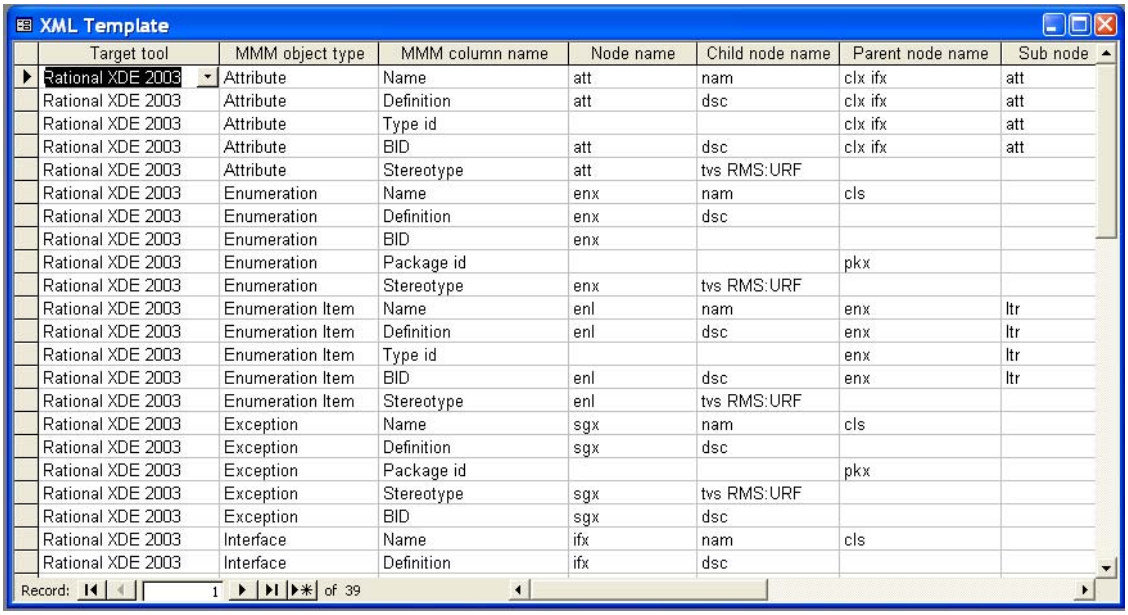
If **By name** is selected, the program will try to find an entity or an attribute based on its name. If it does not find it, the program will create a new entity / attribute with this name. (Be especially careful if renaming has been done in the MMM !)

If **By UDP** is selected, the program will first try to find an entity or an attribute based on the BID that is stored in the “Reference” User-defined Property (UDP). If not found, it will try to find it using the entity’s name. If it is still not found, the program will create a new entity / attribute with the name.

8.3 XML Template

The XML Template makes it possible to customise the import and export capabilities by managing a mapping between an XML tag and an MMM construct.





Target tool:

The name of the Target XML tool or XML file. Can for example be: XDE, XMI, …

MMM object type + **MMM column name**:

The MMM construct to map to the equivalent XML tag

Node name + **Child node name** + **Node attribute**:

The XML tag mapped to the equivalent MMM construct

Parent node name :

Used to retrieve or create an XML element

Sub node name:

Used to retrieve or create the intermediate XML elements if any

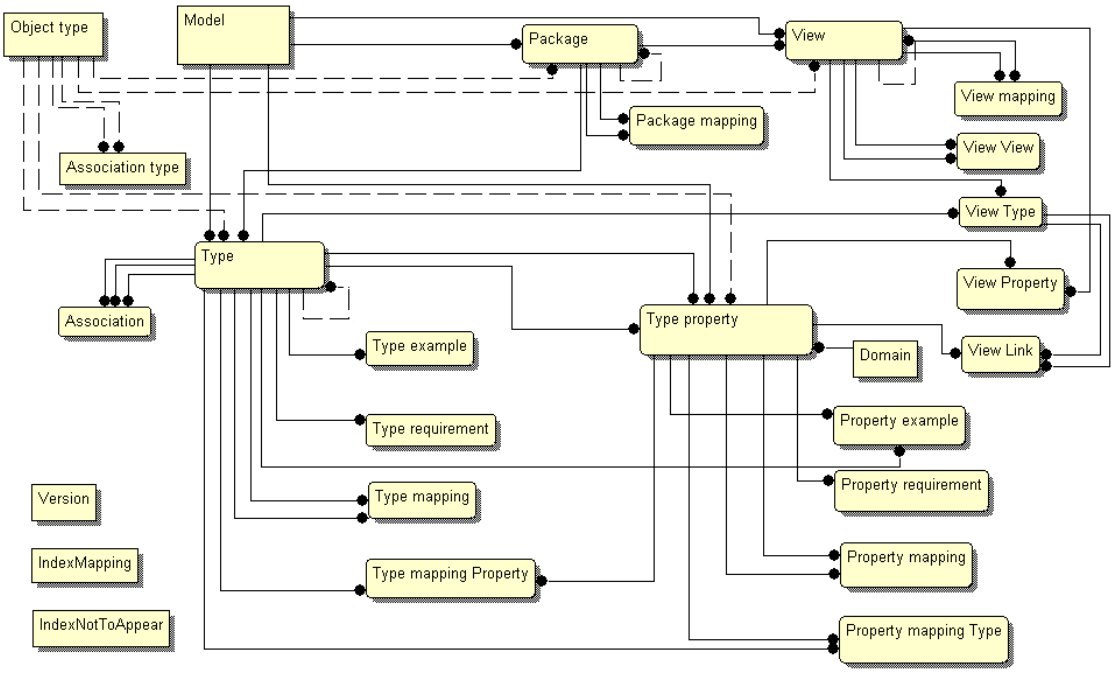
Import function and **Export function**:

Indicates a specific function (customer defined program) to be executed just before the insert/update. It can be used for example to translate code values, or format a text in a certain shape.

CHAPTER 9: TECHNICAL INFORMATION

9.1 Tables Structure

Because all models are stored in the same set of tables the **Model id** foreign-key is present in all tables.
Each object has its own Identifier (**ID** as primary key), which is unique across all of the models. This is an auto-generated number allocated by MS-ACCESS.
Note that the Business Identifier (**BID**) is not unique, as several versions of a model can reside in the MMM tables.



The relationship constraints are defined in the MMM database. They assure the Referential Integrity of the relations between the models (for example: delete cascade of all related Properties when deleting a Type).

The relationships can be edited by clicking on Tools > Relationships, or on  from the main toolbar.

MMM tables that can be shared or accessed remotely:

- Version
- Association
- Association type
- Domain
- IndexMapping
- IndexNotToAppear
- Model
- Object type
- Package
- Package mapping
- Property example

- Property mapping
- Property mapping Type
- Property requirement
- Type
- Type property
- Type example
- Type mapping
- Type mapping Property
- Type requirement
- View
- View Property
- View Type
- View View
- View Link
- View mapping

MMM technical tables that reside with each local MMM database:

- DefinitionsError
- FixLog
- HashTableProperty
- HashTablePackage
- HashTableType
- HashTableView
- HyperlinksThemes
- Index
- ObjectHelper
- Reports
- XML Template

9.2 Installation in a multi-user environnent

The MMM is delivered as one single **.mdb** file that contains both the User Interface and the Repository, populated with the Industry Models.

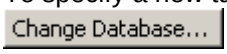
When installing the MMM, there is an option to split the User Interface from the Repository. The split between the user interface and the repository allows easier maintenance, for example, with new releases of the MMM tool, without affecting the actual content of the repository.

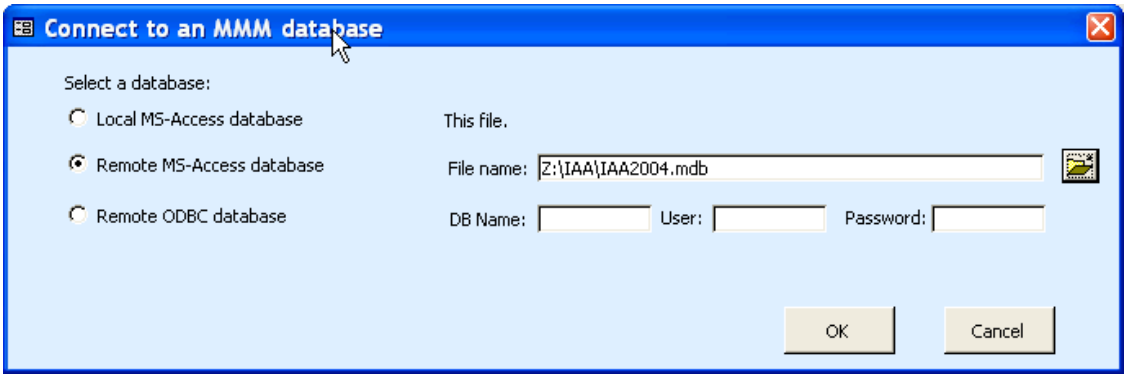
To do this from the main toolbar, go to: Tools > Database utilities > Database Splitter and follow the wizard dialog boxes.

- The **MMM User Interface** (a MS-Access database that contains the Forms, Queries, Reports, Macros and Modules needed to manage the MMM Repository).
- The **MMM Repository** (a MS-Access database that contains a set of tables, populated with the Industry Models).

According to the user's corporate standards, the repository can also be deployed on UDB or other RDBMS, by using an ODBC connection with the MMM user interface.
A file that contains the definitions of the MMM meta tables (MMM_DB2.DDL) is available on the installation CD-ROM, along with a ReadMe.txt that documents the installation steps.

To specify a new target database, open the **Select Model** window, and press on



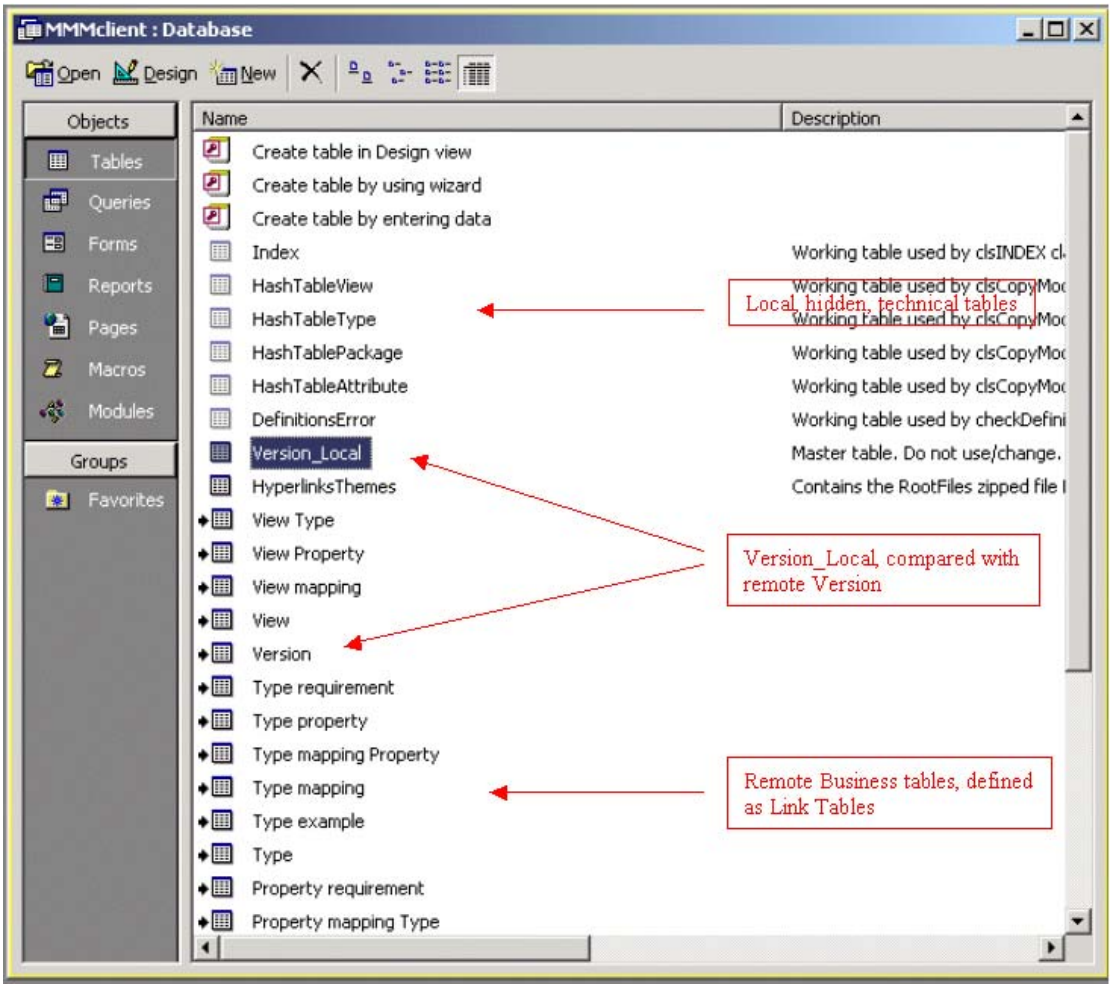


Alternatively, the MMM .mdb file (user interface + repository) can be placed on a shared disk, or a LAN to be accessed by several users at the same time. Locking will be managed at Record level.

9.3 MMM Client

A “light” version of MMM is also provided, that can be used to access a remote database.

This version of MMM only contains the code and the technical tables, but no Business tables, and is therefore much smaller and faster to load.
The MMM Client has an extra table, **Version_Local**, which is used to recognise the MMM as being a light version, and to check its code being at the right level to be used with the remote database.



9.4 Setup a DB2 database

- WARNING – The usage of MMM with an ODBC connection to an external database has not been fully tested. Performance issues may be encountered. This feature is provided on an "AS-IS" base.

Basic steps are as follows:

1) Setup the DB2 environment:
- Create the database (Create database MMM)
- Connect to the database (Connect to database MMM)
- Create the tables (db2 -tf MMM_DB2.DDL)
- Register the database to ODBC
- Setup permissions

2) First time you connect with MMM to the database, you must use the full MMM (instead of MMM Client). This will allow the meta-data to be copied from MS-Access MMM tables to the DB2 tables.

3) Next time you connect to the DB2 MMM database, you can use MMM Client.

4) To populate Business content (Industry models):
- Exporting (SRI format) the models from the MS-Access MMM database
- Importing (SRI format) the models to the DB2 MMM database

9.5 Handling concurrent updates

In a multi-user environment more than one person might be working with the same record at the same time. Because more than one user can change or even delete the same data at the same time, users will occasionally overlap with each other as they work. For this reason, MS-Access uses a technique called optimistic record locking to handle record contention.
If while modifying a record, and someone else updates that record before you save it, MS-Access displays a message box and warns you that if you save the record, you will overwrite the changes that the other user has made. In this case, you can do one of the following:

- Press **Save Record** to save the record and overwrite the other user's changes.
- Press **Copy to Clipboard** to first examine the other user's changes so that you can reconcile them with yours.
- Press **Drop Changes** to ignore your changes and accept the other user's changes.

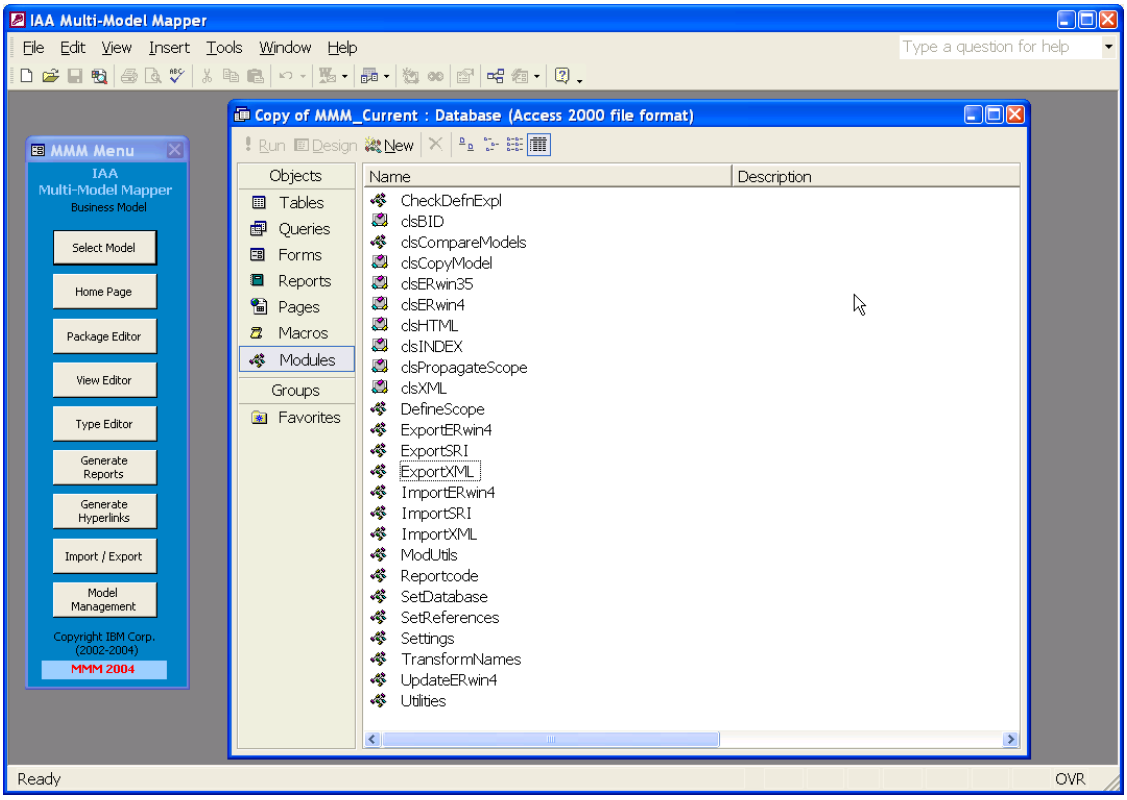
Because other users may add, modify, or delete records that you are also accessing, you should periodically refresh the display of data in your screen, using the **Refresh** button.

9.6 VB Modules and Source code

All of the MMM functions have been developed with Visual Basic, and can be accessed from the **Modules** window by clicking on the  button from the main Toolbar.

Important: This code is not part of any standard IBM product and is provided to you solely for the purpose of assisting you in the development of your applications. The code is provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample code, even if they have been advised of the possibility of such damages.

When changing the code, please be sure to create separate modules, in order not to interfere with future releases of MMM.



Appendix A: Standards and Naming conventions

Requirements model

This section provides the naming conventions that apply to Requirements Models in Industry Models, and is assigned to the model elements.

The Business Identifier (BID) is a reference that uses the Industry Models naming convention and has the following format: **TTcnnnn** where

TT identifies the modeling element:

• BD = Business Direction
• FA = Focus Area (implemented as an MMM-View)
• UC = Use Case (implemented as an MMM-View)
• AD = Activity Diagram (implemented as an MMM-View)
• BP = Business Process
• BA = Business Activity
• EA = External Activity
• SS = System Service
• SY = System
• MT = Metric
• CS = Atomic (Core) Subject Area (implemented as an MMM-View)
• DE = Atomic Data Element
• AS = Analytical Subject Area (implemented as an MMM-View)
• ME = Measure Data Element

c determines the model:

- A = Glossary
- B = Requirements model (IIW)
- H = Requirements model (HPDM)
- Z = Glossary (HPDM)

nnnn is used as a meaningless sequential number.

Please refer to the **Model Editor** and **Object Type Editor** chapters for governance on how to extend/customise these conventions.

Business and Design models

This section provides the naming conventions that apply to the Industry Models models.

The Business Identifier (BID) is a reference that uses the Industry Models naming convention and has the following format: **TTxxxxx** where

TT identifies the modeling element:

– EN = Entity / Type / Class / Association Class / Interface
– RL= direct Relationship
– AT = Attribute
– OP = Operation
– PK = Package

xxxxx depends on the model:

• **cnnnn** for model elements related to the **Business** and **Design** models, where

c determines the model:

– 0 = originates from IAA Ed3 Data Model
– C = Business model (IAA/IIW)
– D = Enterprise model (IAA/IIW)
– E = Interface Design model (IAA/IIW)
– F = Specification Framework (IAA/IIW)
– G = Product model (IAA/IIW)
– I = Conformed Dimensional model (HPDM)
– J to K = Reserved for customer
– L = Enterprise Data Model (HPDM)
– L to P = ACORD models (IAA/IIW)
– Q to V = Reserved for customer
– W = Core Warehouse model (HPDM)
– X to Z = Reserved for IAA / IIW

nnnn is used for a meaningless sequential number.

• **ccnnn** for model elements related to the **data marts,** where

cc is numeric and determines the data mart:

– 01 to 59: Reserved for IIW with:
– 10 = Campaign Management
– 11 = Segmentation Discovery and Management
– 12 = Segmentation (IM4RM table)
– 20 = Customer and Prospect Optimizer
– 21 = Intermediary Performance Analysis
– 22 = Sales Forecast Analysis
– 30 = Underwriting Profitability Analysis
– 31 = Financial Reporting
– 32 = Claim Efficiency Analysis
– 40 = Profitability Analysis for Motorcycle
– 50 = Risk Pricing Analysis
– 51 = Overall Profitability Analysis
– 60 to 99: Reserved for customer data marts

nnn is used for a meaningless sequential number.

Please refer to the **Model Editor** and **Object Type Editor** chapters for governance on how to extend/customise these conventions.

Appendix B: Hints & Tips

Export Filter and Project Scope

A project very often relies on a subset of the models, from Requirement, to Business, to Design and then Implementation.
In the MMM, the suggested way to mark the objects in scope is by using a combination of the Export field and the Project Scope view.

The Export Filter field

The purpose of the **Export Filter** is to manage what content has to be filtered out when exporting a model to a specific downstream environment, such as a CASE tool.

The Export field is present on the following tables:

- Package
- View
- Type
- Type property.

It is used as a filter in the following functions:

- Export SRI
- Export ERwin®
- Export XML

There are two ways to work with the filter:
- **By Exclusion**: all entities are considered as part of the scope (default filter value is "All") and you explicitly exclude the ones you don't want to keep in the scope by giving them the "No" value
- **By Inclusion**: all entities are considered as not being part of the scope (default filter value is "No") and you explicitly include the ones you want to have in the scope by giving them a specific Export Filter value.

A "Reset Filter" option exists in the Model Editor. It allows to reset the filter to "No" when the "by Inclusion" mode has been chosen.
For properties, reset the “All” filter and change it to:

- “ “ if you want all properties to be in scope when selecting a Type
- “**No**” if you want to individually select properties to be exportable.

By default, all objects have an Export field initialized to the value “All”.

“**All**” has the special meaning of preventing an object from being filtered out. (i.e. The object will always be exported or printed).

“**No**” has the special meaning of always filtering the object out. (i.e. The object will never be exported or printed).

Of course, if a Type is not selected, none of its properties is selected, even if the filter is “All” or is the specified value.

In the context of the **IAA Business model**, the Export field has been used to make filtering possible on objects which are specific to an object model, and should appear on an OO modeling tool (such as RSA), or that are specific to a data model and should appear on an E/R modeling tool (such as ERwin®).
For example “**BDM**”
“**BOM**”
“**BDM BOM**”

The Project Scope view

By creating a **View** (of object type "*Project Scope*") you can easily define the scope of a project.

When a Type or an Association is referenced in the **Types Tab** or **Associations Tab** of this view, all its properties (Attribute and Operations) are also part of the project, unless you chose to reference at least one property of this Type or Association in the **Attributes Tab** or **Operations Tab** of this view.

TIP: It is not necessary to specify the Super-types and Sub-types of a Type to be in scope, this is an available option of the Copy Model function to copy them automatically.

Views that are referenced in the **Views Tab** of this view are also part of the project.

Packages do not need to be referenced, as they are implicitly part of the project. (The packages that contain Types or Views in scope are part of the scope.)

The Project Scope view is used as a filter in the following function:

- Propagate Scope
- Copy Model
- Reference Manual

There is a clear distinction between Scoping rules, Propagation rules and Model Copy rules.

The Scoping rules

- A Type specified in the Project Scope View has all its Properties "in scope".
- As soon as one or more Properties are specified in the Project Scope View, all other Properties of the same Type are excluded from the scope.

When creating a Project Scope view, always keep in mind the options described hereafter that can minimize the data entry.

The Propagation rules

- For each Type and/or Property "in scope", the mapping (of a given mapping type) is traced in order to select the Types and/or Properties to scope in the target model.
- The 4 different mapping kinds are taken into consideration: Type-Type, Type-Property, Property-Property, Property-Type.
- When **the *Include Associations*** is selected, all Associations for which the two parents are "in scope" will be "in scope".
- When the ***Include related Types*** is selected, the Interfaces of the target model that are associated ("realizes" and "best implements") to "in scope" Types will be "in scope", as well as the Dimensions for "in scope" Fact tables will be "in scope".

The Model Copy rules

- Every Type and/or Property "in scope" will be copied into the (empty) target model.
- Every View "in scope" will be copied, but will only reference Types/Properties/Views that are "in scope", unless the ***in-scope View items extend scope*** option is selected.
- When the ***Copy Super-Types*** or ***Copy Sub-Types*** option is selected, the Super-Types or Sub-Types of a Type or a Property in the scope will be "in scope".
- When the ***Copy Associations*** option is selected, the Associations for which the two parents are in the scope will be "in scope".
- Packages for which a Type or a View is "in scope" will be copied.
- The new target model will always comply with the referential integrity.

Summary

There are two complementary features used for filtering:

- The Export Filter field is mainly at a higher level of business modeling decision, regardless the definition of a specific project scope.
- The Project Scope view is mainly at project management level, allowing the definition of a specific project's boundaries.

To setup an environment for defining the subset of the model that is in scope, you can perform the following steps:

1. Provide a new model prefix for the BID allocation that distinguishes the original Industry Models object from customized objects.

   This can be done in the Model Editor.
   See the *Model Editor* chapter and the *Standard and Naming Conventions* appendix.

2. Choose the way you will scope at corporate level: Filter (by Inclusion of Exclusion) and review the filter values.

3. In the context of a specific project, for each object (View, Type, and possibly Type property) in scope, reference it to your *Project Scope* view.
   This can be done via the View Editor, or from the Type Editor and Property Editor.

4. Use the Copy Model function with Project Scope filtering, to create a small model that represents your subset. This subset model will then be useful for documenting the project. From this model, you can easily generate the Hyperlinks.
   See the *Copy Model* chapter for more detail.

5. Use the Propagate Scope function to create the equivalent scope view in a downstream model to define the equivalent subset, according to the mapping.
   See the *Propagate Scope* chapter for more detail.

Customising the Hyperlinks

- MMM is preloaded with a set of default files that defines the appearance of the Hyperlinks. If you want to put your company logo, change the background image, the icons, the navigation bar etc, you can do so by defining your own ***Theme***.
  Refer to the section *4.3 Defining Themes*.

- If you want to manually create a “home page” per model, called here “Main view”, you can do so by placing a file called **Main_body.bak** in the hyperlinks model directory before re-generating the hyperlinks. This “body” will be embedded in the generated Main_view.htm page.
  This file can contain any valid html code to be inserted inside a <body> …</body> section.

- If you want to place some additional pictures in an HTML file (for example a Class Diagram in a View), place the picture in the hyperlinks model directory **before** re-generating the hyperlinks. The picture file (**.gif .jpeg .wmf**) should have the same name or BID as the html page where you want to see it.
  For example, place SAC0001.GIF into the c:\Hyperlinks\IAA\Business directory.
  Note: This feature is available for Packages and Views only.

- The same logic applies to external documents (**.doc**) that are placed in the directory.

- If you want to preserve the content of an HTML file from being re-generated (for example you made some manual changes to its content), copy/rename the file to the same directory with a **.bak** extension.
  When re-generating the hyperlinks, the .bak files will overwrite the freshly re-generated files.

Migrating models from a previous MMM version

The migration steps from a previous version of MMM **tooling** to a new version of MMM is as follows:

1) For each model: Export SRI all the content from the previous MMM (Full Export option)
(packages.SRI, types.SRI, assoc.SRI, attributes.SRI, mapping.SRI, extensions.SRI)

2) Create the models in the new MMM if they don't exist yet

3) For each model: Import SRI all the content to the new MMM
(packages.SRI, types.SRI, assoc.SRI, attributes.SRI, extensions.SRI)

4) Once all the models are loaded in the new MMM database:
For each model: Import SRI all the mapping to the new MMM (mapping.SRI)

This approach avoids any problem linked to table design changes, different internal identifiers, etc.

The migration steps from a previous version of an MMM **model** to a new version of that model are described in section *3.4 Copy Model.*

Upgrading a customised model to a newer Industry Model content

The migration steps from a previous version of **model** to a new version of an Industry Model is as follows:

1. use the delivered Delta spreadsheet to identify the changes and assess the impact on the existing customized model
2. whenever you decide to incorporate changes into your customized model, you can use the Project Scoping facility to create a temporary model that can then be exported in an SRI format and re-imported into the customized model.

MS-Access Editing Tips

How to modify data content?

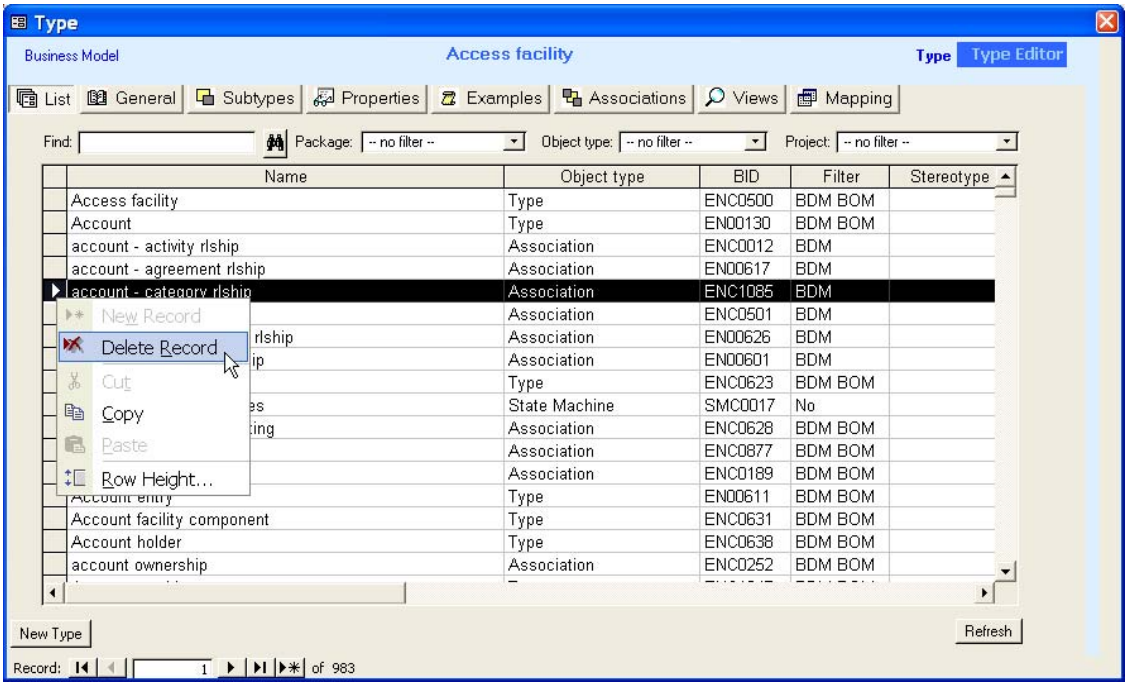
Any changes entered on a data field is directly stored. There is no “Save” or “Apply” or “Commit” button.

How to add new rows?

Rows available for data are marked by an * in the left margin.

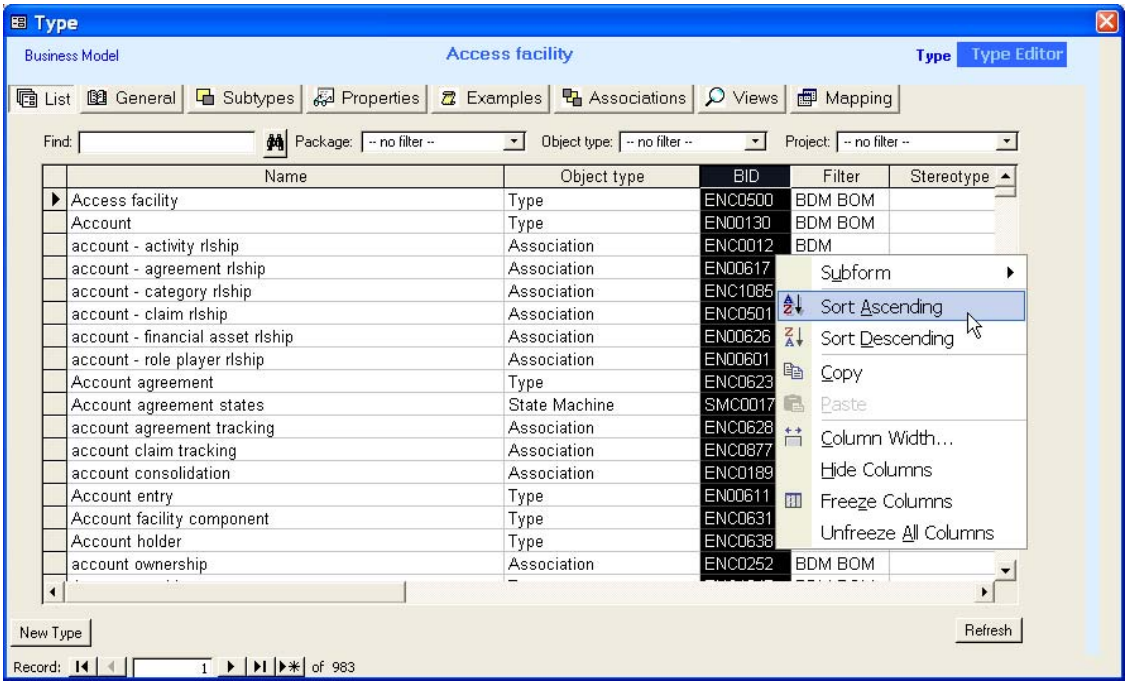
How to delete rows:

Right-click on the row margin (where a black arrow appears in front of the selected row) and choose *Delete Record*



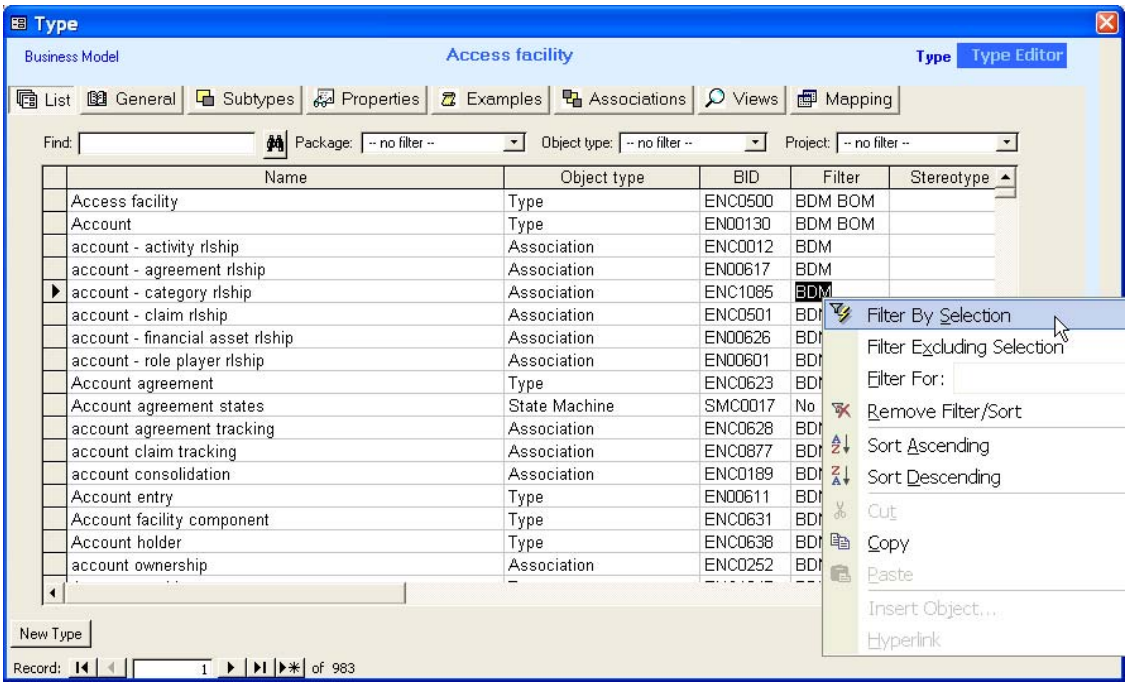
How to sort on a specific column?

Right-click on the column header, and then choose *Sort Ascending* or *Sort Descending*. This enables the user to **sort** by the values of the rows in that column into useful groups. Below is an example of the Type Editor's columns being sorted.



How to filter on a specific column value?

Right-click on the cell containing the value on which you want to filter, and then choose *Filter By Selection* or *Filter Excluding Selection*. This enables the user to **shorten** the list to the rows that have (or not) the specified value. Below is an example of the Type Editor's columns being filtered.

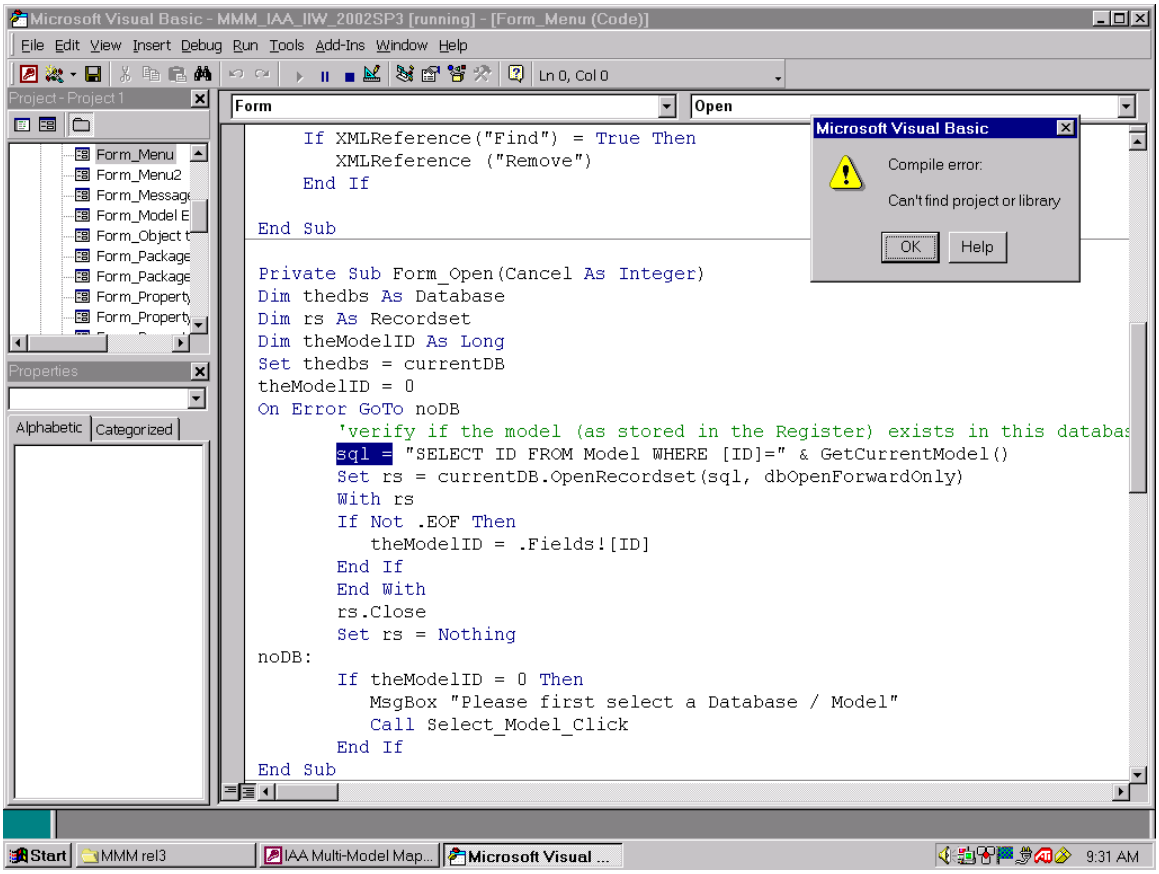


MMM Installation problem

For performing advanced functions, MMM needs to have some additional files installed on the machine.
Usually, those files are installed as part of the MS-Office components, but this may depend of the installation options taken when installing MS-Office.

If missing, they can be downloaded from the http://support.microsoft.com site.

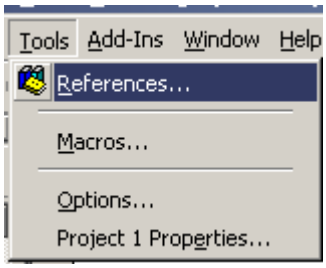
When MMM starts, it tries to locate the files, and if it doesn't find them, or if a file is incompatible with other installed components, you can see such a window appearing:



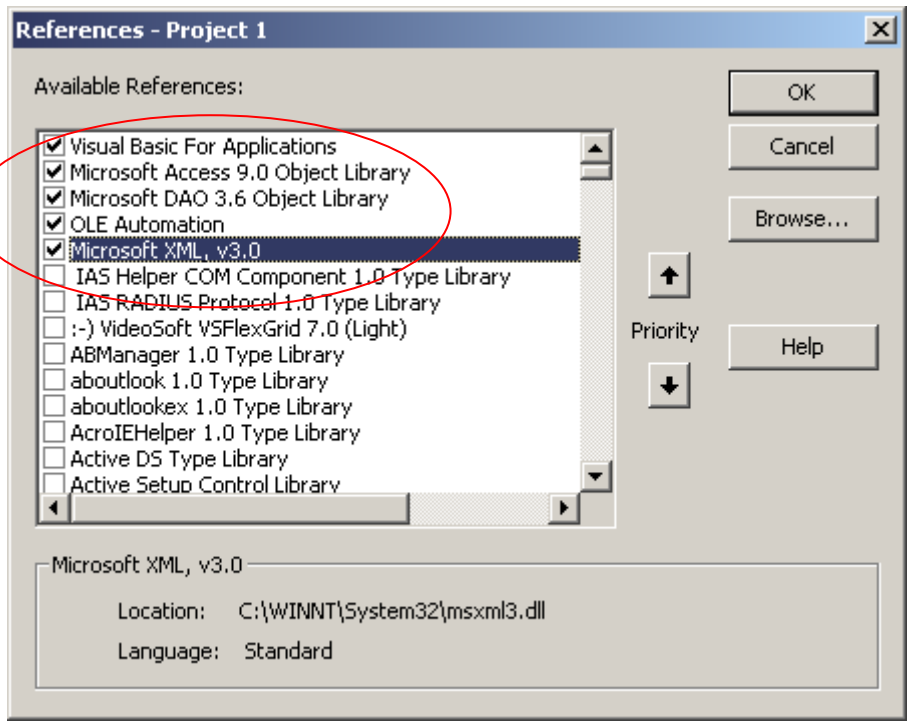
How to fix the problem:

1. Check that the file is well referenced by MS-Access:

   - Open MMM by holding the SHIFT key while double-clicking on MMM
   - Press ALT-F11 to open the MS-Visual Basic window
   - Go to Tools -> References



   - Verify that you well have the following six available References; and without the "***missing***" warning.



   - If one of them is ***not selected***, look down in the list to select it, or if ***missing***, use the Browse button to add the dll as Reference. They are respectively:

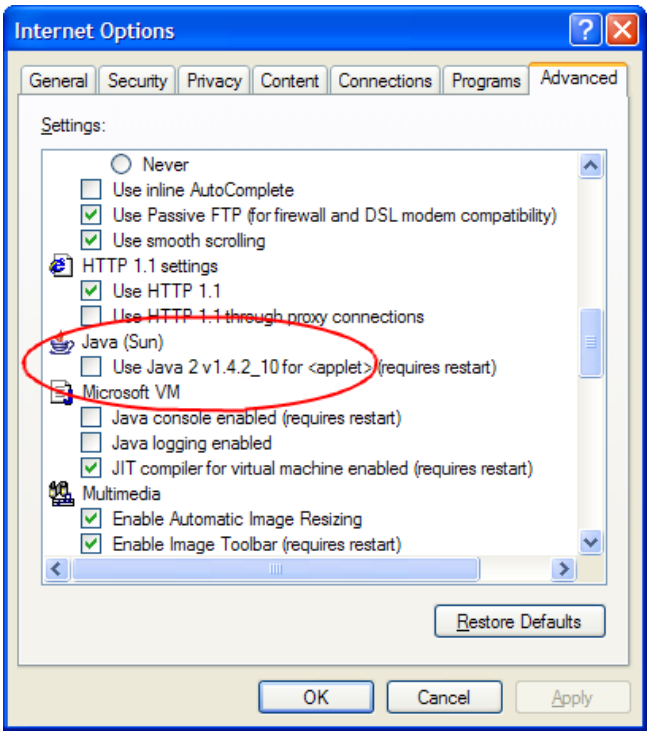
     - c:\Program Files\Microsoft Office\Office\MSACC9.OLB
     - c:\Program Files\Common Files\Microsoft Shared\DAO\DAO360.DLL
     - c:\WINNT\System32\stdole2.tlb
     - c:\WINNT\System32\msxml3.dll

     or later versions.
   - If you are using the ERwin® 3.5 export function, the er2api32.dll is also required.

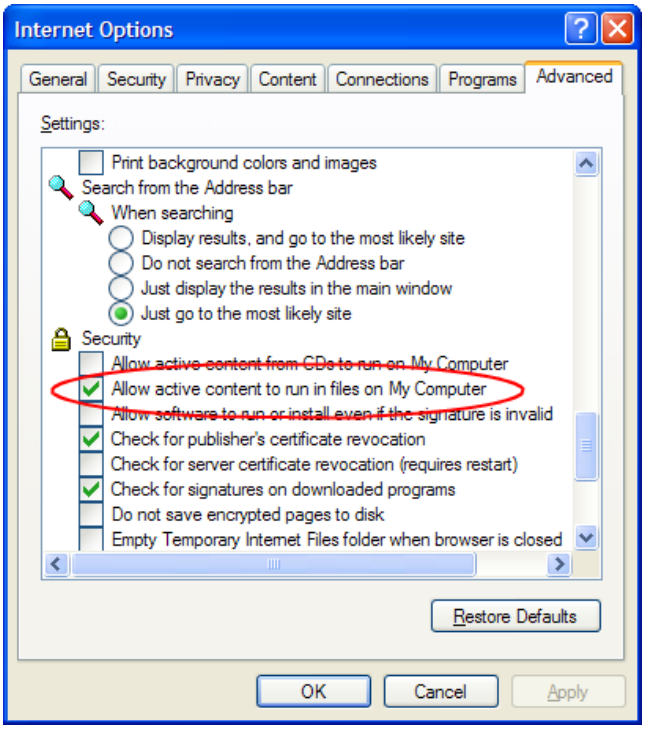
3. Close MMM and restart it the normal way.

Hyperlinks Tree-view problem

The Hyperlinks tree views are implemented by applets and require Java.
The Sun Java 1.5.0_11 has been tested successfully.
If any trouble encountered while opening the Hyperlinks tree views (applet), it can be due to an incompatibility with another Java installation. In such a case, deactivate the option in Internet Explorer:



Also, since the applet runs from the local Hyperlinks files stored on your computer, the following security option should be activated:



END OF DOCUMENT