Dokument Typ:
Document Type: Manual

Titel:
Title:

DADI–MA Reference Manual

Lieferbedingungs-Nr.: DRL/DRD No.:	N/A	**Klassifikations Nr.:** Classification No.:	N/A
Produktgruppe: Product Group:	PR 1216401	**Konfigurationsteil-Nr.:** Configuration Item No.:	1216401
Schlagwörter: Headings:	N/A	**Produktklassifizierungs-Nr.:** Classifying Product Code:	8–QABA
		Freigabe Ordnungs-Nr.: Release Order No.:	N/A

Bearbeitet: Prepared by:	MDA – Team	**Abteilung:** Department:	RIO 62	**Firma:** Company:	DASA RI
Geprüft: Agreed by:	I. Lenz	**Abteilung:** Department:	RIO 62	**Firma:** Company:	DASA RI
Genehmigt: Approved by:	P. Athmann	**Abteilung:** Department:	RIO 63	**Firma:** Company:	DASA RI
Genehmigt: Approved by:		**Abteilung:** Department:		**Firma:** Company:	

DOCUMENT CHANGE RECORD

Issue / Rev.	Issue date	Pages / Section Affected	Remarks
(1/ –)	(05.02.95)	(all)	(initial version)
(1/A)	(28.07.95)	(all)	(update and extension of rev. –)
(2/–)	(01.12.95)	(all)	new document style new Main Menu line and pull down menues new section Composite Aggregate Definition new section Category Report Definition new section List Composite Aggregates new section Data Type Definitions update section Print Reports function Change Object Name deleted
(3/–)	(04.04.96)	(all)	Updated screen forms
(4/–)	(01.09.97)	(all)	
(4/–)	(31.03.2000)	(5)	new menu Cross Reference Constraints new section Cross Reference Constraints

This page is intentionally left blank.

1 INTRODUCTION

1.1 Identification and Scope

This part is the Reference Manual of the MDA DADI–MA Users and Operations Manual. It provides a reference to the most fundamental and commonly used features of the Data Dictionary Maintenance Application called DADI or DADI–MA.

1.2 Purpose

The Mission Database Application (MDA) constitutes the set of utilities which support or enable various activities typically performed during mission preparation. As such its main objective is to prepare for and support the development of space segments and missions.

As part of MDA the Mission Database (MDB) is viewed as the central repository of information about flight configurations. In order to perform flight configuration operations this kind of information is stored and manipulated in the database together with Hardware and Software configuration information about flight elements, Payloads and Ground Support Equipment.

The Data Dictionary Maintenance Application that is described from a usage and operations point of view in this book of the Users & Operations–Manual does provide the capabilities to define MDB table structures representing the End–Items and by that representing the data structures which make up the Mission Database itself.

2 APPLICABLE AND REFERENCE DOCUMENTS

	Document No. **Document Title**	**Issue / Revision**	

2.1 Applicable Documents

2.1.1	SPE 1216 401 002 MDA Requirements Specification	2/C	05.11.1993
2.1.2	COL–RIBRE–ICD–0015–00 System to MDA Interface Control Document	3/–	28.02.1997

2.2 Reference Documents

2.2.1	ADD 1216 401 002 MDA Architectural Design Document	2/A	18.06.1993

3 OVERVIEW OF DADI

The Data Dictionary – Maintenance Application DADI–MA is an editor for the Mission Database Data Dictionary and a generation tool for exporting the Data Dictionary information to the MDB. This manual shall give an overview of how to operate DADI–MA, how to implement a new MDB Data Dictionary and how to export the Data Dictionary (DD) to instantiate a new version of the Mission Database (MDB).

DADI–MA is an Oracle Forms application. Everyone who uses DADI–MA should be familiar with Oracle Forms Application handling (e.g. function keys, short keys etc.) and Motif applications including the Motif window manager. Also helpful for completing this manual are the according Oracle manuals: Getting started with Oracle Forms (Part No. A11986–1) and Oracle Reports Operators Manual GUI Version 2.0 (Part–Nr. A14002)

☞ Please note the term DADI, DADI–MA are used synonymously within this Document because the former version of the application was called DADI.

3.1 Reference Manual Structure

The user manual includes an overview about the implementation concept of DADI–MA, the files and different examples which show step–by–step the use of DADI–MA for editing the Data Dictionary.
As an aid for self training, anyone who is familiar with all the topics mentioned here will gain an overall understanding of DADI–MA necessary to complete his work outside of the examples presented.

3.2 Conventions used in this Manual

This manual uses certain format and style conventions. This section also shows how general key names used in this manual relate to keys and controls on your keyboard.

- Entries in a pull down menu are shown like this:
  File–>Select Version
  representing the main menu topic ”File” handling with the subitem ”Select Version” has to be selected.
- Buttons are shown like this:
  Exit
  representing that the Exit button has to be pressed.
- Relations to data entry fields are shown like this:
  Password
  representing that the field password has to be selected and filled by the user.
- Mouse clicks are indicated by the term ”Select”.
- Essential remarks are indicated in italic like this: *mandatory*
- Essential Notes are written in this style with a hand in front of it.

This page is intentionally left blank.

4 CONCEPTS SUMMARY

This section explains the data structure within a Data Dictionary Version. A detailed description of the concepts relying on DADI–MA is within the Introduction manual of the DADI U&O–Manual.

The DADI–MA maintains Oracle table structures representing the data dictionary for the Mission Database. Working on this table structures the user (type administrator) can edit the definition of the MDB Data–Types also call End–Item–Types.

The logical data design and data structure handling follows the rules listed below.

- Data Dictionaries can exist in Versions. This allows that DD Versions can be maintained independently from each other.
- The data structure breakdown is as follows within one DD Version.
  - Data Type Domain

    Data Types can be grouped in so called Domains. Domains can share (re–use) Data–Types.
  - Data Type Definition

    A Data Type Definition is constructed out of Data Aggregate Definitions. Data Aggregates can be shared (re–used) between Data Type Definitions.
  - Data Aggregate Definition

    A Data Aggregate Definition is constructed out of Data Attribute Definitions.

The DADI–MA delivery does provide pre–defined Data Types needed by Applications accessing MDA/ MDB at run–time. Those Data Types shall not be modified or deleted.

☞ Predefined Data–Types should not be changed because other applications are relying on the existence of these Data–Types.

4.1 DADI S/W Architecture

The DADI–MA Software was build using Oracle 7, PL/SQL, Oracle Forms 4, Oracle Reports 2 and SQL*Plus scripts. It consists of mainly four parts:

- the graphical user interface
- the report definitions
- the SQL installation
- the MDB export scripts.

This page is intentionally left blank.

5 DADI–MA OPERATIONS AND USAGE

This chapter gives an overview of the main topics for using the DADI tool. After you have completed this chapter, you should be able to define a new **Data Dictionary** and related **End–Item–Types**, to print reports and to export the Data Dictionary to the MDB.

The Usage of the DADI Maintenance Application is explained in a logical order starting with the operations on Data Dictionary Level, End–Item Domain grouping, followed by the common DD content browsing operations, the End–Item–Type related operations – and so forth, down to the Data–Attribute level.

Using the browser and the main menu, the Data Dictionary and End–Item–Type administrator can:

- Browse top–down through the Data Dictionary
- Select Objects (Domain, Type, Aggregate, Attribute) for editing
- Delete Objects (Domain, Type, Aggregate, Attribute)
- Create object relations
- Commit changes and refresh the screen
- Export Data Dictionary versions

5.1 DADI–MA User Interface

5.1.1 Main Menu Line and Pull Down Menues

The DADI–MA command are collected into logical groups. The menu names appear in the menu line in the upper part of the DADI–MA application window. The Main Menu line has the following entries.

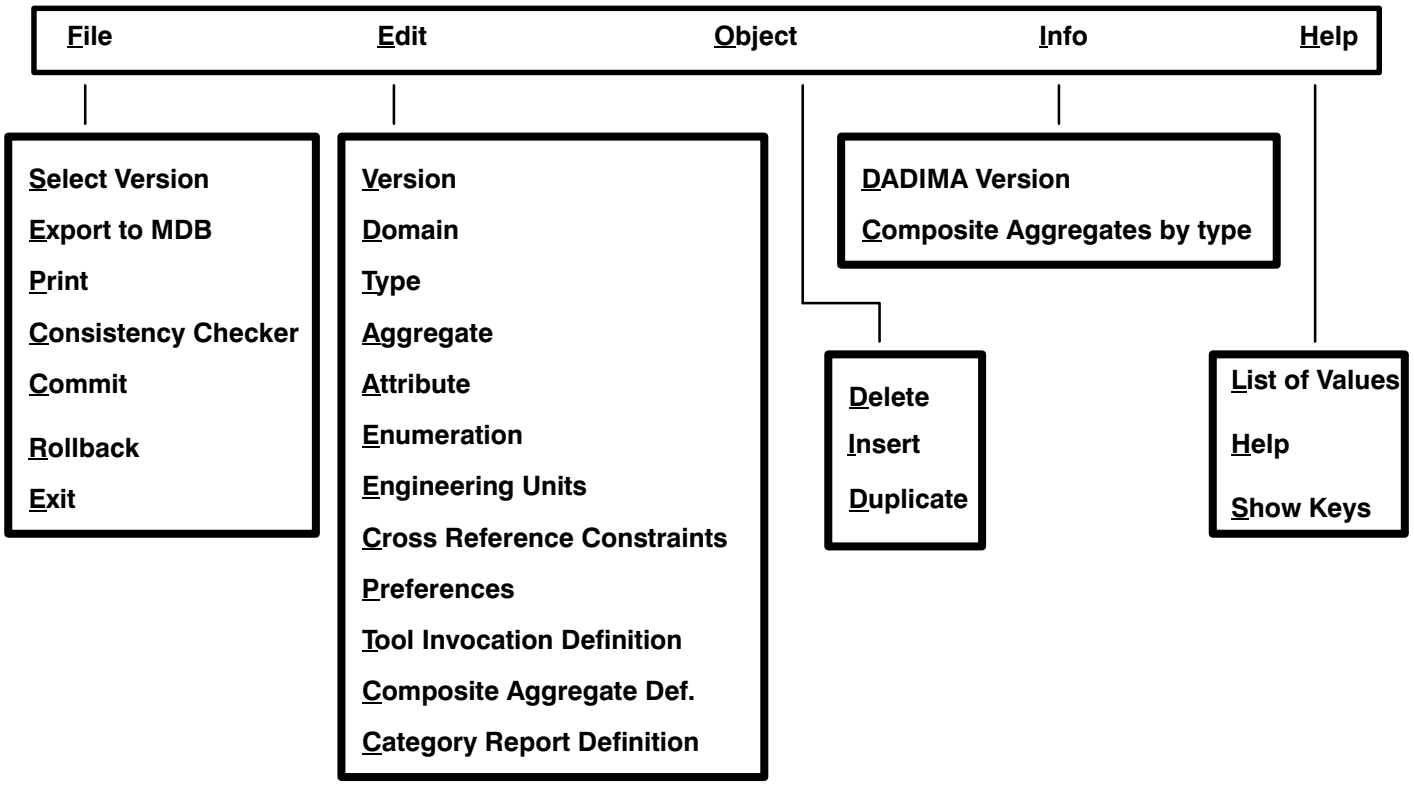

Figure 1. Main Menu Line Entries

The Pull Down menue items related to the Main Menu line allow to perform the following commands:

File :

Select Version :	Selection of a specific data dictionary version
Export to MDB :	Exports the definitions performed with this tool to the MDB
Print :	Report capability to print several kinds of reports
Consistency Checker	check the consistence of the data.
Commit :	In a database transaction sense commit makes table changes persistent, meaning data modifications are stored in an end user sense.
Rollback :	In a database transaction sense rolls–back modifications to the last Commit point, meaning the modifications are dropped and the old status re–loaded.
Exit :	Quits the DADI–MA tool

Edit :

Version :	Creation and modification of a data dictionary version
Domain :	Creation and modification of a domain
Type :	Creation and modification of a type
Aggregate :	Creation and modification of an aggregate
Attribute :	Creation and modification of an attribute
Enumeration :	Creation and modification of an enumeration
Engineering Units :	Creation and modification of engineering units
Cross Reference Constraints :	Creation and modification of cross reference constraints
Preferences :	Selection of the default values when entering DADI–MA
Tool Invocation Definition:	Attachment of user application tools to CDU versions, CCU versions, or End–Items by the flexible tool invocation
Composite Aggregate Def. :	Definition of composite aggregates
Category Report Definition :	Definition of category Reports

Object :

Delete :	Deletion of an object record
Insert :	Insertion of an object record
Duplicate :	Creating of a new object by copying an existing one

Info :

DADIMA Version :	Information about the actually running DADIMA version CCU versions, or End–Items by the flexible tool invocation
Composite Aggregates by types :	Listing of the defined composite aggregates related to a specific type

Help : List of Values : Shows a list of available values for an entry field

Help:	Informs case sensitive about expected actions the user should take to complete the current operation.
Show Keys :	invokes a window showing the key bindings to functions.

5.1.2 The Browser Window areas and fields

All you need to work with the DADI–MA tool is either on the screen or on your finger tips. The following illustration shows the most important parts of the DADI–MA screen where the user will perform the work. It is the main menu line with the pull down menu items and the browser window.

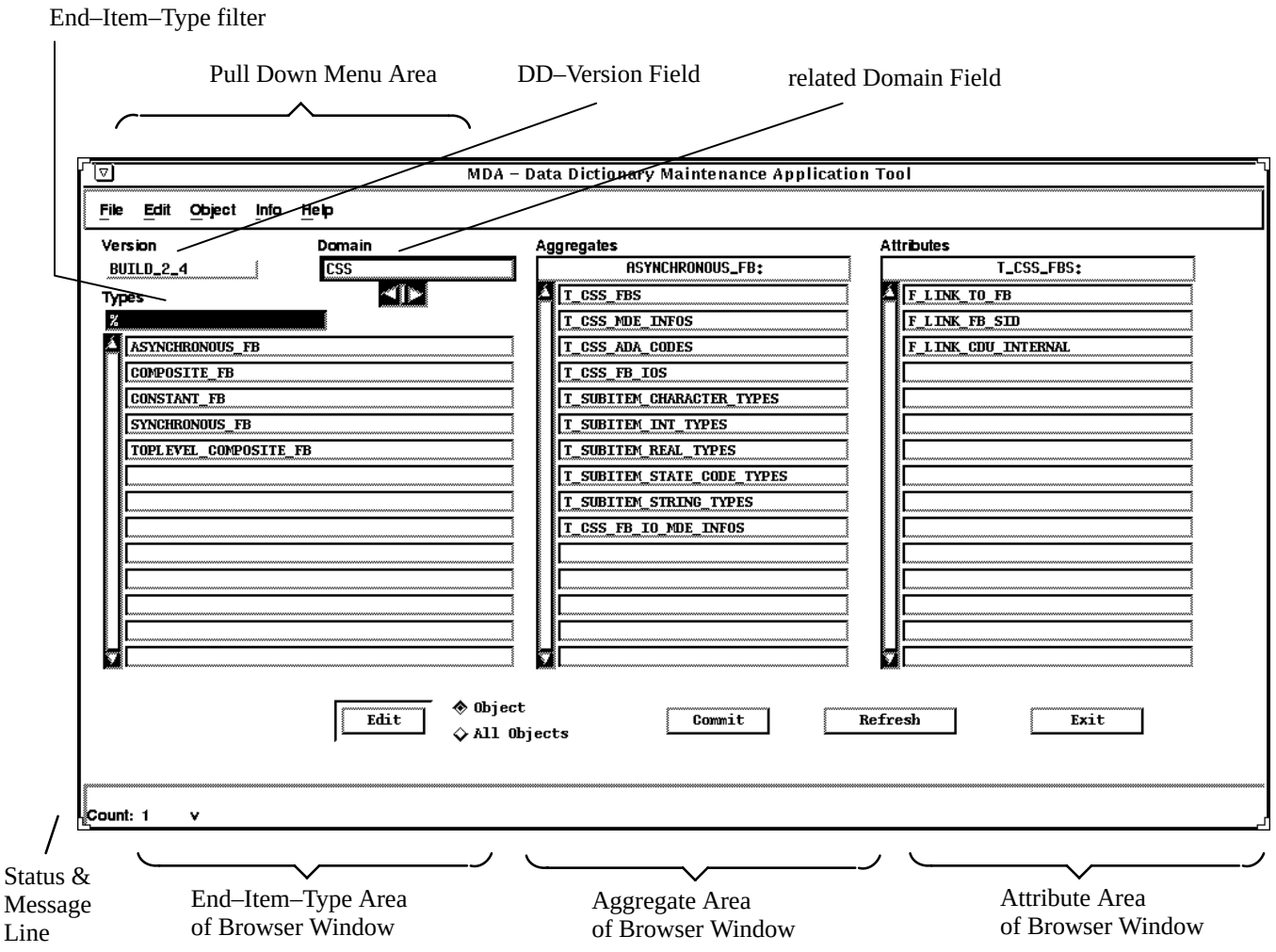

Figure 2. DADI Browser Window

5.2 Startup and Login

To enter the DADI–MA Tool it is prerequisite that you have been locked in the Unix environment and started Open Windows. .

Within a command shell type ”$MDA_HOME/util/dadi/bin/start_dadima” .

The scripts asks the user to enter username and password of the DADIMA–Tool:

Enter OWNER NAME of the DADIMA account [DADIMA] :
Enter OWNER PASSWORD of the DADIMA account [DADIMA] :
[1] [pid]

DADIMA–Tool started

The tool is started in batch mode and after a short time the graphical user interface is displayed at the screen.

5.3 Data Dictionary (DD) Versions

You can define, edit, delete, import, export and select versions of a MDB Data Dictionary. You can work only with *one version* at a time. If you want to use more than one version you can start another DADI–MA tool in parallel.

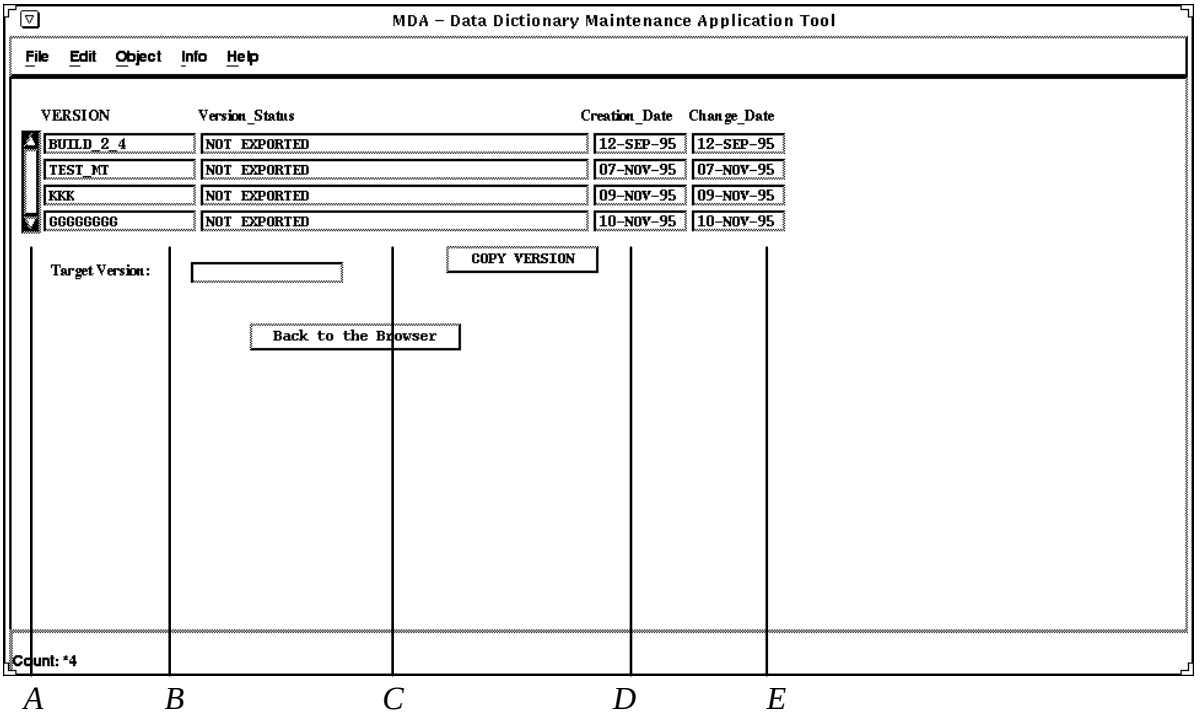

Figure 3. Data Dictionary Version Editor

A *Version Scroll List*

Allows to access the different available Data Dictionary Versions

B *Version Name List*

Displays all available Data Dictionary Versions of the actually installed database.

C *Version Status List*

Displays the actual status of the related version. If the version has just been created and is within the development, the status is *not exported*. If a development milestone has been reached, the version status can be changed to *baseline*. The third option for the version status is *exported*, indicating that the Data Dictionary version has been exported to the MDB and that the version has been installed The version status has to be changed by the operator.

☞ *Note: DADIMA–Tool does not prevent the user from changing data in baselined or exported MDB Version. The status field is for information only.*

D *Creation Date List*

Displays the date when the Data Dictionary version has been created. The date is created automatically during the version creation.

E Change Date List

Displays the date when the version status has been changed the last time.

5.3.1 Creation of a new DD Version

You can create a new Data Dictionary version at any time while you are working in DADI–MA. By this creation new empty data tables will be generated.

To create a new DD Version

1. Select **Edit–>Version** from the Main Menu line
   The version form shows up in the main browser window.
2. Move with the cursor to the Version scroll list and select an empty version field.
3. In the ***Version*** field type in the name of the new version, e.g. 'My_New_Version'.

The creation date and change date have default values which can not be changed.

4. The ***Version Status*** has a default value as well. To change the Version Status do the following:
   - press the **Return** key *or* **Ctrl+l** to invoke the Version Status Window.
   - Select the appropriate value and press **Ok**.
5. Press button **Back to Browser** to close the version form.
6. Press **Commit**
   or
   File–>Commit
   to Commit the newly created DD version (empty).
7. Press the button **Refresh** to update the version field.

5.3.2 Copying a DD Version

You can create more than one version of a DD Version within the database for safekeeping. The copy capability is also useful in the case of creation of a revised DD version. Each version can be saved under a different name.

To copy a DD Version

1. Select **Edit–>Version** from the Main Menu line.
   The version form shows up in the main browser window.
2. Select the ***Source DD Version*** you want to copy from within the ***Version*** list.
3. Position the pointer in the ***Target Version*** text entry field.
4. Enter the new target version name e.g. 'My Target Version'.
5. Press Button **Copy Version** to start the copy process.
6. You have to confirm the start copy operation by selecting the **Start** button.

☞ Please note , that the target version must not exists in the Version List.

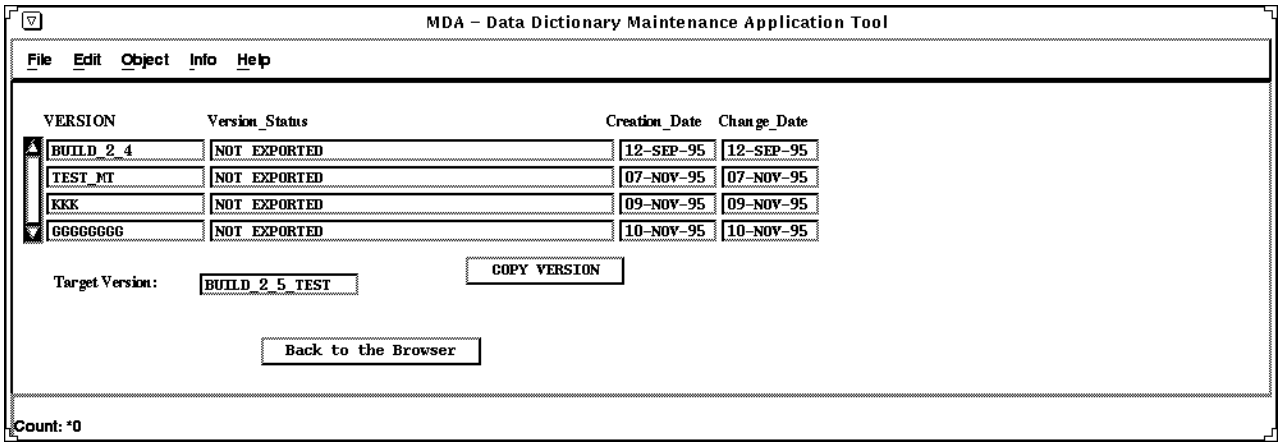

Figure 4. Copy from an existing DD–Version into a target DD–Version

5.3.3 Selecting a DD Version

If you are working on different Data Dictionary versions in parallel it is essential to have the capability of switching between different versions. You can quickly open any of the existing DD versions using the select command.

To select a DD version

1. Select **File–>Select Version** from the main menu line.
   The version selector box shows up.
2. Select the DD version within the version scroll list by the **Up / Down Arrow key**
   or
   use the Scroll bar.
3. Perform a **double left** mouse click
   or
   press the **Ok** button to use the selected DD version and all related data.

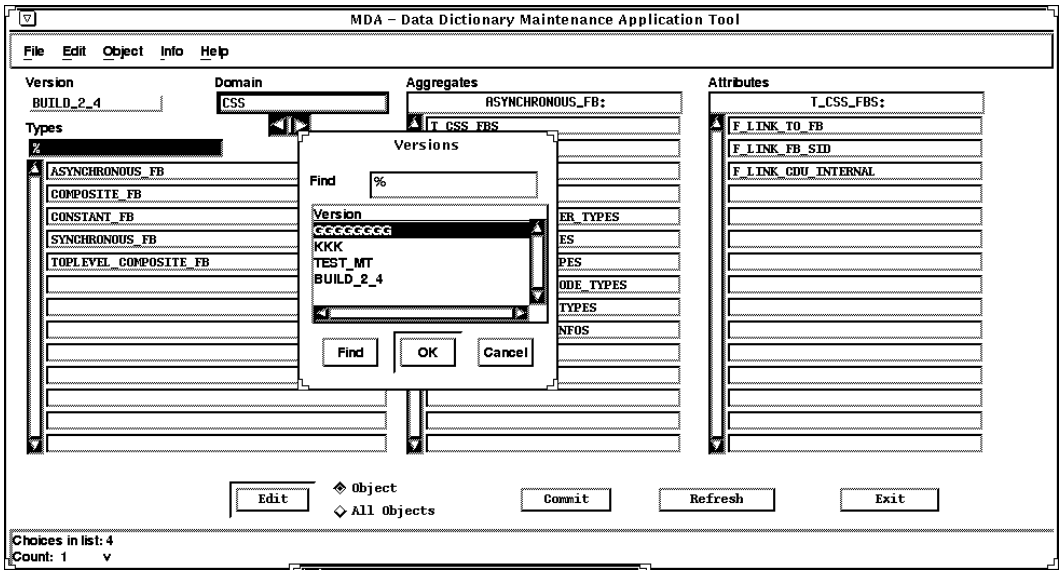

Figure 5. Select a Data Dictionary Version

5.3.4 Selecting a Default DD Version

The default Data Dictionary version will be loaded initially when starting DADI–MA. This version name is shown in the version field of the main browser window.

To select a default DD version

1. Select **Edit–>Preferences** from the Main–Window menu to enter the Preferences option window. The preferences view shows up in the main browser window.
2. Select the field ***Default Version***.
3. Press **Ctrl+l** to enter the version selection window.
4. Select a new default version and press the **Ok** button in the selection list.
5. Press **Apply** to activate the new preferences.
6. Press **Back to the Browser** to go back to the main browser window.
7. The browser window is updating with the contents of the default version.

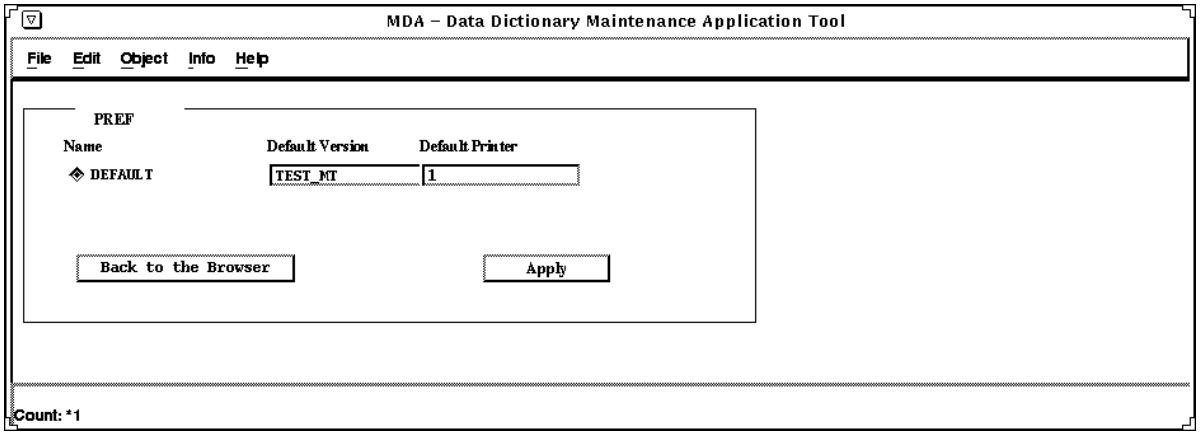

Figure 6. Define a default DD–Version

5.4 Using the Browser

The browser allows you to look at the data defined in your Data Dictionary, e.g. Versions, Domains, End–Item–Types, Aggregates and Attributes. Each time when selecting one of these objects within your Data Dictionary a browse on the database will be done.

To select a DD Version

1. Select **File–>Select Version** from the main window.
   The version selector box shows up.
2. Select the DD version within the version scroll list by the **Up / Down Arrow** key
   or
   use the Scroll bar.
3. Perform a double left mouse click
   or
   press the **Ok** button to use the selected DD version and all related data.

To select a Domain

1. Use the scroll bar arrows ◀ || ▶ right beyond the ***domain field*** to select the domain.
   All existing domains within the selected DD version are selectable.

   When you have changed the domain, the other blocks in the browser (Type, Aggregate, Attribute) are updated correspondingly.

To filter End–Item–Type Names

DADI–MA provides a filtering function to select subsets of data. Wildcards can be used as the Oracle SQL wildcards. If you write a %–char in the End–Item–Type filter field above the type block, all types corresponding to the selected domain are shown in the type block. This is the default configuration.

1. If you want to select all Types starting with S, define ”S%” in the type filter.
2. Press **Refresh** to make the selection persistent. All types beginning with S are shown.

To select an End–Item–Type, Aggregate, or Attribute

To select one of these objects you can use :

- the corresponding scroll bars
- the Up / Down arrow keys
- single click on the specific object you want to use.

To edit Single Objects

If you want to edit a domain, type, aggregate or attribute you have to perform the following steps:

1. Be sure that the radio button right from the **Edit** button has the value ”***Object***”.
   If it has the value ”All objects”, not the selected object will be presented but all objects of this type.
2. Select the object you want to edit and then press **Edit** on the bottom of the browser.
   The selected object editor window is exposing.

To edit Multiple Objects

If you want to edit several objects of the same category you have to perform the following steps:

1. Set the radio button to the value ”***All Objects***” by selecting on the button **All objects** .
2. Select the object type you want to edit and then press **Edit** on the bottom of the browser.
   The selected object editor window is exposing with a list of objects that can be edited.

To delete a Object Relations

The object relations are:

- Attribute to Aggregate relations
- Aggregate to End–Item type relations
- End–Item type to Domain relations

It is suggested to delete the object relations within the browser by the following steps:

1. Select the attribute, aggregate or type whose relation shall be deleted within the browser.
2. Execute **Object–>Delete** to delete the object relation.
3. Press **Commit** to make the attribute deletions persistent
4. Select the desired attribute, aggregate or type sroll list in the browser.
5. Press **Refresh** to update the browser scroll lists.

Within the browser, the attribute, aggregate or type is no longer visible. The records containing the attribute, aggregate or type data are still there and may be accessed with Edit–All Objects.

To delete an Attribute

An attribute can only be deleted if attribute constraints have been deleted prior to the attribute deletion.

1. Select the ***Attribute*** you want to delete within the browser and press **Edit** to enter the ’End–Item Type Editor.

If any constraint is defined, the next three steps have to be executed for each constraint.

2. Select the constraint Name field and
3. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the constraint.
4. Press **Commit** to make the constraint deletions persistent
5. Select the **Attribute** to be deleted in the browser.
6. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the constraint.
7. Press **Commit** to make the attribute deletion persistent
8. Select the attribute scroll list in the browser.
9. Press **Refresh** to update the browser scroll lists.

To delete an Aggregate

An aggregate can be deleted by the following steps:

1. Delete all attributes of the aggregate to be deleted by the previously mentioned steps.
2. Select the ***Aggregate*** you want to delete within the browser and press **Edit** to enter the ’Aggregate Editor’.

If any relation is defined, the next three steps have to be executed for each relation.

3. Select the ***Related Types*** field where the relations are defined..
4. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the relation.
5. Press **Commit** to make the relation deletions persistent
6. Select the ***Name*** field.
7. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the aggregate.

8. Press **Commit** to make the aggregate deletions persistent
9. Select the aggregate scroll list in the browser.
10. Press **Refresh** to update the browser scroll lists.

To delete a Type

A type can be deleted by the following steps:

1. Delete all attributes of the aggregate to be deleted by the previously mentioned steps.
2. Select the ***Aggregate*** you want to delete within the browser and press **Edit** to enter the 'Aggregate Editor'.

If any relation is defined, the next three steps have to be executed for each relation.

3. Select the ***Related Types*** field where the relations are defined..
4. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the relation.
5. Press **Commit** to make the relation deletions persistent

If any CGS type mapping is defined, the next three steps have to be executed for each mapping.

6. Select the ***Mapped to CGS–Type*** field where the mapping is defined..
7. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the relation.
8. Press **Commit** to make the relation deletions persistent
9. Select the ***Name*** field.
10. Execute **Object–>Delete**
    or
    Ctrl+Del to delete the aggregate.
11. Press **Commit** to make the aggregate deletions persistent
12. Select the aggregate scroll list in the browser.
13. Press **Refresh** to update the browser scroll lists.

To delete a Domain

A domain can be deleted by the following steps:

1. Delete all attributes, aggregates and types of the domain to be deleted as previously mentioned.
2. Select the ***Domain*** you want to delete within the browser.
3. Select the ***Domain*** you want to delete within the browser and press **Edit** to enter the 'Domain Editor'.
4. Select the ***Domain Name*** field.
5. Execute **Object–>Delete**
   or
   Ctrl+Del to delete the aggregate.
6. Press **Commit** to make the aggregate deletions persistent
7. Press **Refresh** to update the browser scroll lists.

To delete a Version

It is not possible within this DADI–MA version to delete a Version.

To refresh the Object–List

To refresh (re–query the Oracle database) an object list:

1. Select an item within the list you want to refresh.
2. Press **Refresh** to re–query the database

5.5 Creation of Data–Dictionary Objects

DADI–MA allows you to create different data objects in your Data Dictionary to create and extent the dictionary. The following objects can be created:

- end item type domain,
- end item type
- end item type aggregate
- end item type attribute

5.5.1 Creation of Domains

Domains are groups of End–Item–Types. A Domain refers to a set of related End–Item–Types, where the relation may be logical, functional or operational depending on a specific environment or a specific purpose.

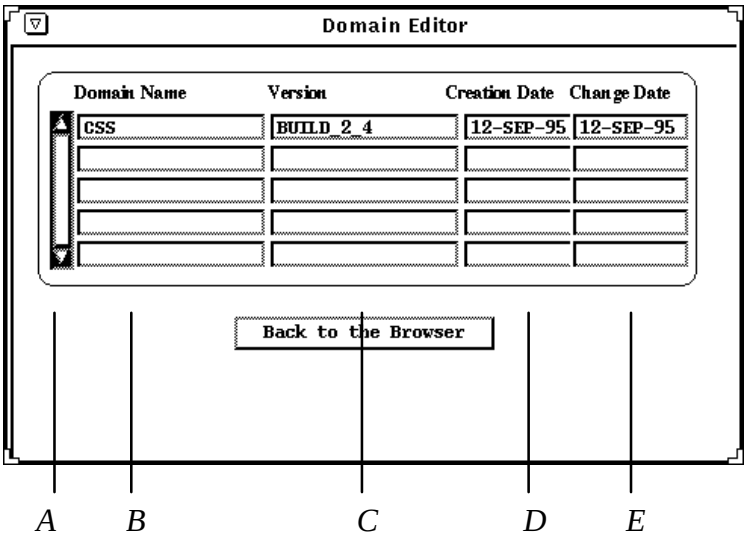

Figure 7. Domain Editor

A *Domain Scroll List*

Allows to access the different available Domains

B *Domain Name List*

Displays all available Domains of the actually installed database.

C *Version List*

Displays the related version for the domain.

D *Creation Date List*

Displays the date when the domain has been created. The date is created automatically during the domain creation.

E *Change Date List*

Displays the date when the domain has been changed the last time. The change date update is performed automatically.

To create a new Domain

1. Select the ***Domain*** field and press **Edit** to enter the 'Domain Editor' window.
2. Select within the ***Domain Name*** scroll list an empty field.
   The new domain name NEW_DOMAIN will be inserted by default when selecting an empty field.
3. Press **Ctrl+u** to delete the default value and enter the domain name.
   The version, creation date and change date have default values which can not be changed.
4. Press **Back to the Browser** to return to the browser window.
5. Select **File–>Commit** from the main menu
   or
   press the **Commit** button in the browser window to make your changes persistent.
6. Select the ***Domain scroll list*** and press **Refresh** to update the domain scroll list.

5.5.2 Creation of End–Item–Types

Each End–Item is of a given type, called End–Item–Type. An End–Item–Type may be related to a specific domain to create sets of types for a specific purpose. An End–Item–Type is comprised of one or more Aggregates.

User defined consistency check procedures can be related to an End–Item–Type. The consistency check procedure have to be defined by the user and in DADI–MA the connection of the procedure to the End–Item–Type will be created.

If an End–Item–Type shall be mapped to a different type, not known in the MDB, this can also be defined. The mapping procedures must be created externally by the user.

A mapping to the standard CGS End–Item–Types can be defined directly. For this kind of mapping it is not necessary to create a mapping procedure.

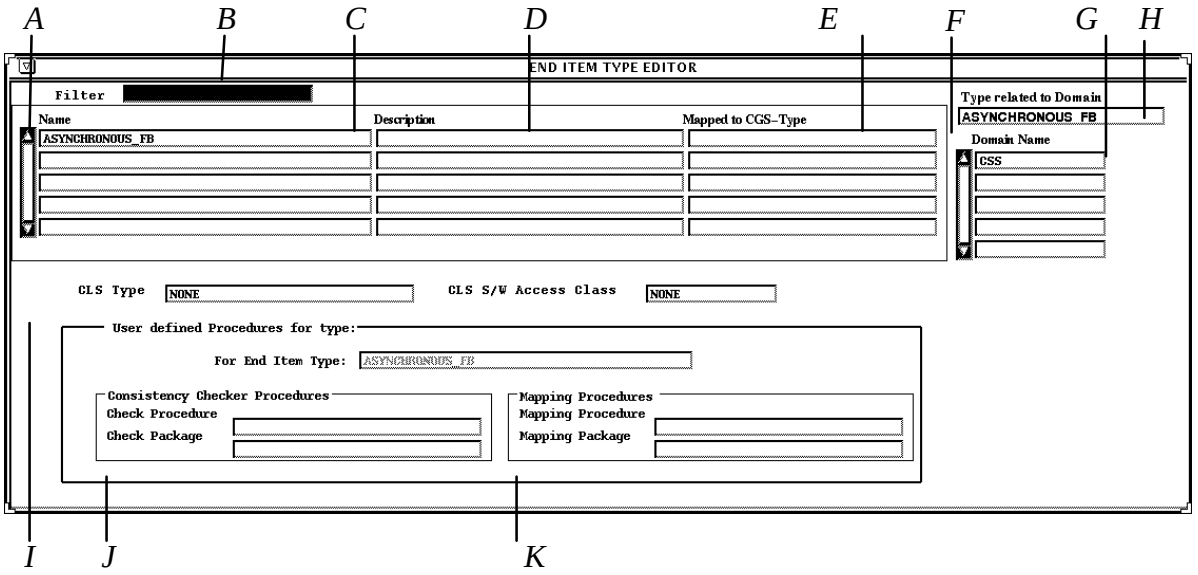

Figure 8. End–Item–Type Editor

A *End–Item–Type Scroll List*

Allows to access the different available End–Item–Types.

B *Filter Function*

Allows to perform a selection of End–Item–Types, starting , ending or including a specified substring.

C *End–Item–Type Name List*

Displays all available End–Item–Types of the actually installed database and allows to implement new End–Item–Types.

D *Description List*

The fields contains comments from the user describing the definition he did.

E *Mapped to CGS–Type List*

Defining the mapping of an End–Item–Type to a standard CGS End–Item–Type. The standard CGS End–Item–Types are predefined and must be available in the actual installation.

F *Domain Scroll List*

Allows to access the different domains which are related to the actually selected End–Item–Types.

G *Domain Name List*

Contains the domain names which are related to the actually selected End–Item–Type. All relations to one type are shown once.

H *Type related to Domain*

When creation a relation to domain, the selected End–Item will be displayed.

I *CLS Type Definition*

This field is for the definition of a specific End–Item type, the CLS Type, which is a Columbus Ground Software (CGS) data type.

J *Consistency Check Procedures*

A user can create his own consistency check procedures, which extends the standard consistency check. It will be specified which consistency check procedure in which consistency check package shall be used for a specified type.

K *Mapping Procedures*

A user can create his own mapping procedures to map his own data type structures to the standard data type structures. It will be specified which mapping procedure in which mapping package shall be used for a specified type.

To create a new End–Item–Type

1. Select an ***End–Item–Type*** field within the End–Item–Type scroll list and press **Edit** to enter the the 'End–Item–Type Editor' window.
2. Select within the ***Name*** scroll list an empty field.
   A new End–Item–Type with the name NEW_TYPE will be inserted by default when selecting an empty field.
3. Press **Ctrl+u** to delete the default entry and enter the End–Item–Type name.
4. A short description to explain the effect of the End–Item–Type can be included. For this, select the ***Description*** field and enter a description.

If the End–Item–Type shall be mapped to an CGS End–Item type, the next two steps have to be executed. The result of the mapping is the mapping of the CGS type Aggregates to the user defined End–Item. After definition, the CGS End–Item type aggregates are also includes in the browser aggregate scroll list.

The definition of CGS type mapping is optional. For a mapping definition the steps NO TAG to NO TAG have to be executed.

5. Select the ***Mapped to CGS–Type*** field and press **Ctrl+l** to pop up the CGS–Type list.
6. Select the CGS–Type by a **double left** mouse click on the related type
   or
   select the type and press **Ok**.

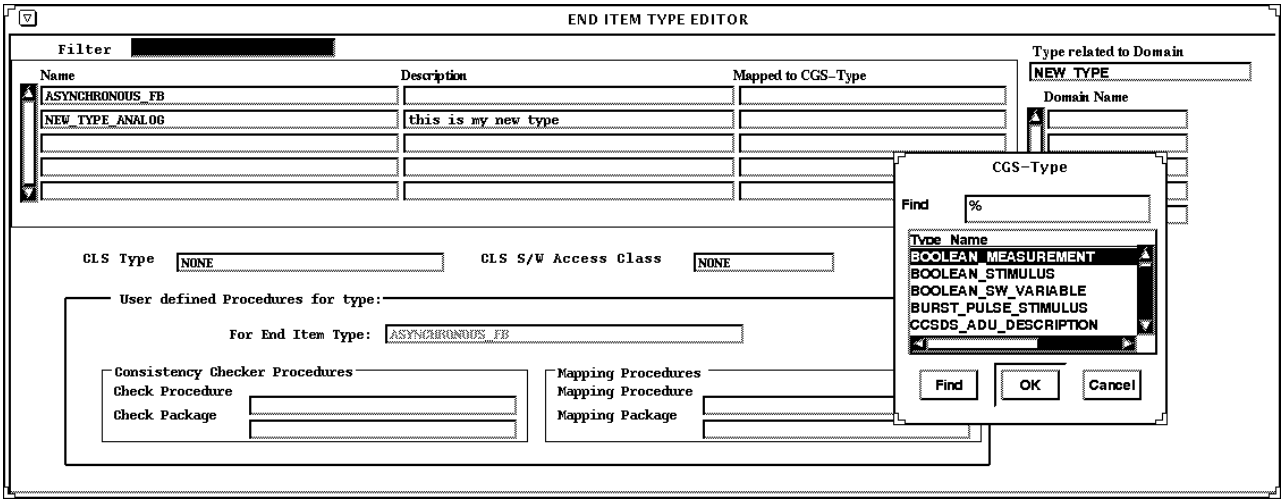

Figure 9. Create an End–Item–Type

The definition of CLS informaton is optional. For CLS definition the step NO TAG has to be executed.

7. Additional CLS–Information can be inserted in the according two ***CLS Type*** fields.
Type **Ctrl+l** to use the corresponding selection list.
Select the appropriate value.
Press **ok** to get back to the end–item–type window.

☞ Please note that the CLS Type and CLS S/W Access Class field are only of interest for CGS Data Type Administrators. CGS does provide pre–defined Aggregate dealing with above CLS parameters. Those data types shall not be modified by other users.

Field CLS–Type : provides a mapping to CLS internal data representation as they are

- NONE, STRING_TYPE, STATE_CODE_TYPE, INTEGER_TYPE, REAL_TYPE, BOOLEAN_TYPE, BITSET_TYPE, CHARACTER_TYPE, WORD_TYPE, PATHNAME_TYPE, TIME_TYPE, COMPLETION_CODE_TYPE, UNSIGNED_INTEGER_TYPE,LONG_REAL_TYPE

Field S/W Access Class : as they are

- NONE, READ, READ_WRITE, EXECUTE, SEND, IMPORT, PATH_SELECT, NODE_SELECT

The definition of user defined Consistency Check procedures is optional. For a user defined Consistency Check definiton the steps NO TAG to NO TAG have to be executed.

8. Select the **Check Procedure** field and enter the name of the user defined consistency check procedure related to the End–Item–Type shown in the field ”User defined Procedures for Type:”.
9. Select the **Check Package** field and enter the name of the package where the user defined consistency check procedures are included.

The package name field is not mandatory. Procedures might be stored outside a package.
DADI–MA does not check if the specified user defined consistency check procedure exists. The procedures may be written later and the MDA Consistency Checker will check if the procedures are available.
The procedure name and package name might be written in lower or upper or mixed letters. The names must be unique.

The user defined Consistency Check Packages and Procedures shall be copied to the directory:

$MDA_HOME/config/mdb/install/user_defined_procedures for automatic installation at MDB ’ininitialize time.

The filenames must end with **.sql** (e.g. check_measurement.sql) and have to contain the **EXIT** statement at the end of the procedure bodies.

The definition of user defined Mapping procedures is optional. For a user defined Mapping definiton the steps NO TAG to NO TAG have to be executed.

10. Select the **Mapping Procedure** field and enter the name of the user defined procedure for End–Item–Type mapping related to the End–Item–Type shown in the field ”User defined Procedures for Type:”.
11. Select the **Mapping Package** field and enter the name of the package where the user defined mapping procedures are included.

The package name field is not mandatory. Procedures might be stored outside a package.
DADI–MA does not check if the specified user defined mapping procedure does exist. The procedures may be written later and the MDA Consistency Checker will check if the procedures are available.
The mapping procedure name and package name might be written in lower or upper or mixed letters. The names must be unique.

The Mapping Packages and Procedures shall be copied to the directory:

$MDA_HOME/config/mdb/install/user_defined_procedures for automatic installation at MDB ’ininitialize time.

The filenames must end with **.sql** (e.g. mapp_measurement.sql) and have to contain the **EXIT** statement at the end of the procedure bodies.

12. The ***Related Domains*** field is for the definition of an End–Item–Type to Domain relation. How to perform a relation definiton is explained in detail in chapter NO TAG.
13. Select **File–>Commit** from the main menu
    or
    press the **Commit** button in the browser window to make your changes persistent.
14. Select the ***End–Item–Type scroll list*** in the browser area and press **Refresh** to update the End–Item–Type scroll list.

5.5.3 Creation of Aggregates

Aggregates are a further refinement of the End–Item–Types. Definitions for the entry to the Attributes of End–Items in I_MDB can be defined. Aggregates may be related to End–Item–Types.

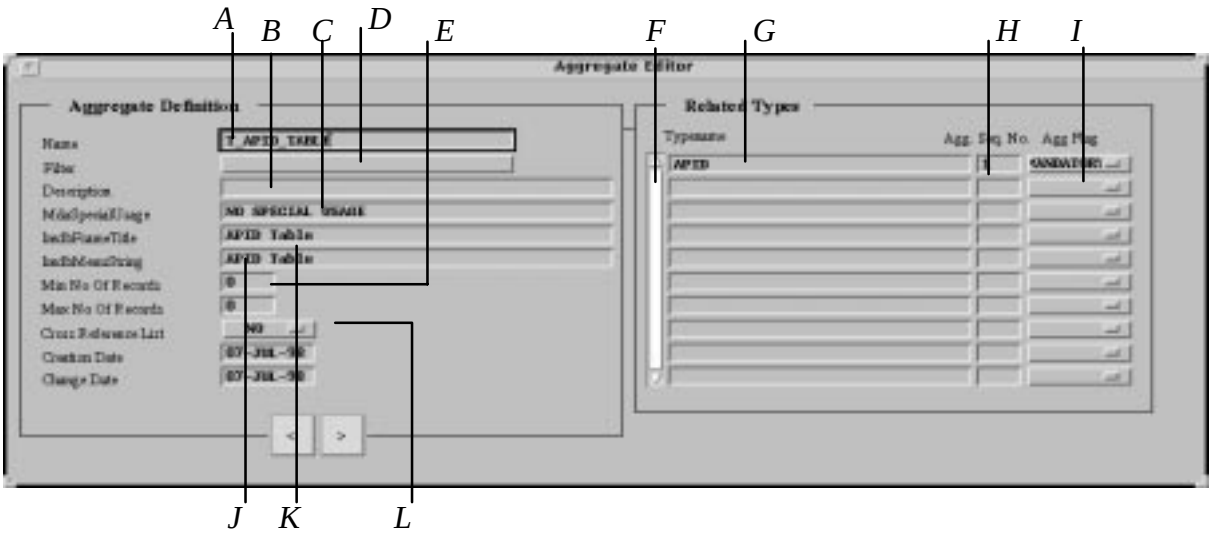

Figure 10. Aggregate Editor

A *Aggregate Name*

Specifies the aggregate name. The aggregate itself is defined by the content of all other fields in the aggregate editor.

B *Description*

The fields contains a comment from the user which explains the definition he did.

C *MDA Special Usage*

An aggregate may have a special usage when operating with the MDA. When operating with foreign keys to perform a cross reference between the MDB and other databases, it may be specified here.

D *Filter Function*

Allows to perform a selection of displayed aggregates, starting , ending or including a specified substring. The scroll bar defines the view when the ’all object’ mode is selected and the scroll is performed by the scroll buttons at the bottom of the window.

E *Number of Records*

The minimum and maximum number of records for an aggregate can be specified. If the maximum number is greater than one, the aggregate is called a multi record aggregate.

F *End–Item–Type Scroll List*

Allows to access the different types which are related to the actually selected aggregate.

G *End–Item–Type Name List*

Contains the End–Item–Type names which are related to the actually selected aggregate.

H *Aggregate Sequence Number*

The aggregate sequence number defines the ordering of the aggregate belonging to the End–Item–Type.

I *Aggregate Flag*

The aggregate flag defines if the aggregate has to exist or not. If the flag set to mandatory, the consistency checker performs a check.

J *I_MDB Menu String*

The menu string is the entry menu title of the end item aggregate editor in I_MDB.

K *I_MDB Frame Title*

The frame title is the window title of the end item aggregate editor invocation in I_MDB.

L *Cross Reference List*
This flag defines if the aggregate is a cross reference list or not. The default value is NO.

To create a new Aggregate

1. Select an ***Aggregate*** field within the Aggregate scroll list and press **Edit** to enter the 'Aggregate Editor' window.
2. Use **Object–>Insert**
   or
   the scroll arrows ◀ || ▶ at the bottom of the window to create a new entry.

A new domain Aggregate with the name NEW_AGGREGATE will be inserted by default when selecting an empty field.

3. Select the ***Name*** field and press **Ctrl+u** to delete the default aggregate name.
4. The ***Creation Date*** and ***Change Date*** have default values which normally shall not be changed.

The change date will be updated each time when you change the aggregate and execute commit.

5. Select the ***Description*** field and enter a description explaining the sense of this aggregate.

6. Select the ***MDA Special Usage*** field to define the MDA special usage.
7. Type **Ctrl+l** and select the appropriate value from the selection list.
   - NO SPECIAL USAGE: Default value of MDA special usage field
   - CLS_FORMAL_PARAMETER: Shall not be used by other than CGS Data Administrators.

☞ Please note that the CLS Formal Parameter field is only of interest for CGS Data Type Administrators. CGS does provide pre–defined Aggregate dealing with above CLS parameter. Those data types shall not be modified by other users.

- MDA_VERY_LONG_RAW:
  Is used to indicate that the aggregate represents a byte image e.g. S/W Executable or Display definition etc.

This type is related to the attribute data type LONG_CHAR in case of a text and is related to the attribute data type LONG_RAW in case of an image.
The data type will be used to start tool like MatriXx or UCL.
Only *one* attribute is allowed to be defined for this aggregate data type.

☞ Please note that the ”Max No. of Records” field has to be set to ”0” to indicate that as many records as needed shall be used!

- FOREIGN KEY: Is used to indicate that this aggregate is a foreign key.

8. Select the ***IMDB Frame Title*** field and define the window frame title that IMDB shall use at run–time for the End–Item–Aggregate editor.

9. Select the ***IMDB Menu String*** field and define the menu entry title that IMDB shall use at run–time for the End–Item–Aggregate editor invocation.

10. Select the ***Min No. of Record*** field and define the minimal number of records for a multi record aggregate

11. Select the ***Max No. of Record*** field and define the maximal number of records for a multi record aggregate.

If the maximum number is greater than one, the aggregate is called a multi record aggregate. A maximum value of zero defines an unlimited multi record aggregate.

E.g. for a point pair calibration aggregate with maximal five point pairs the entry has to be set to 5.

☞ Please note that for Byte Images e.g. Pictures, Source Code etc. the ”Max No. of Records” field has to be set to ”0” to indicate that the number of records is not limited.

If a multi record has been defined, it is necessary to set the multi record flag for at least one attribute. Not defining the MRF flag would lead to displaying three dots within I_MDB.

12. The Min Records field must be less or equal to the Max Records field.

13. Change the cross reference list item to YES if the aggregate should be a cross reference list.

14. Select **File–>Commit** from the main menu
or
press the **Commit** button in the browser window to make your changes persistent.

15. The ***Related Types*** field is for the definition of an aggregate to End–Item–Type relation. This relation is optional, but in general it makes sense to create it. How to perform a relation is explained in detail in chapter NO TAG.

16. Select the ***Aggregate scroll list*** in the browser area and press **Refresh** to update the Aggregate scroll list.

5.5.4 Creation of Enumerations

An Enumeration is a data type that can be defined and used for an attribute.

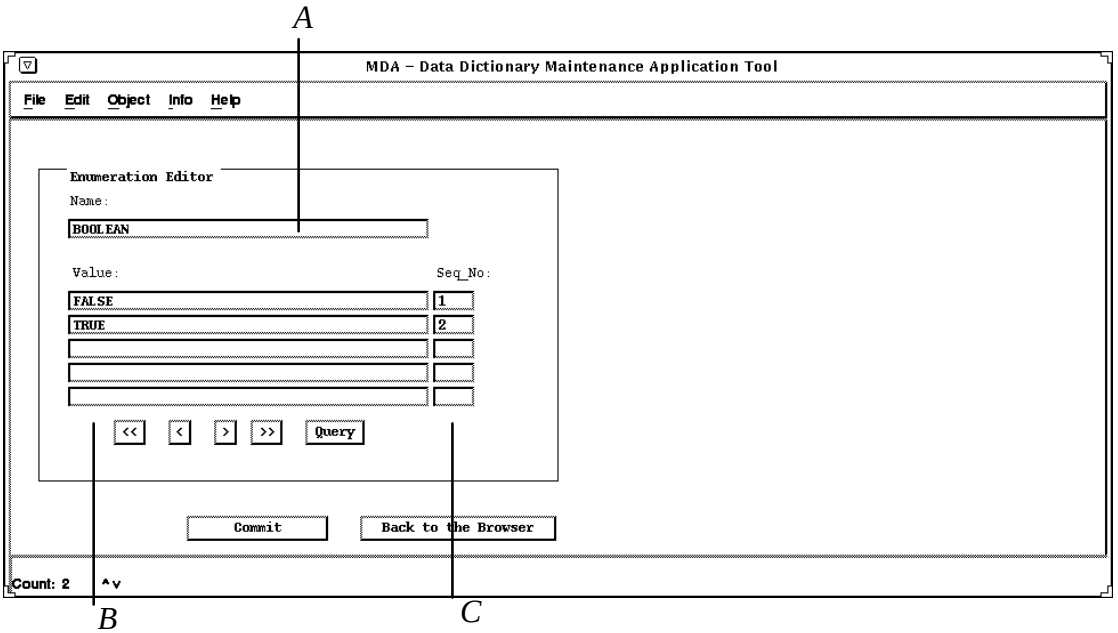

Figure 11. Enumeration Definition Form

A *Name*

Specifies the Enumeration Name. The Enumeration itself is defined by the contents of value fields and the sequence numbers. An enumeration has

B *Value*

The Enumeration Value is a string which defines the value of the enumeration itself.

C *Sequence Number*

The Enumeration Sequence Number defines the ordering of the enumeration in the MDB.

☞ *There is one enumeration which has a special handling, i.e. UNITS. Enumeration values and Sequence Numbers for this enumeration are created by using the function* ***Edit–>Engineering Units****.*

To create a new Enumeration

1. Select **Edit–>Enumeration** from the main menu to enter the Enumeration Editor.
2. Scroll to an empty enumeration record with **scroll right ▶**
   or
   use the menu **Object–>Insert** to get an empty scroll list entry.
3. Select the ***Name*** field and enter the enumeration name.

4. Select the first ***Value*** field and enter the enumeration value.
5. Select the ***Seq No*** field and enter the enumeration sequence number.

The sequence number defines the ordering within the enumeration value selection list in I_MDB.

6. Create additional enumeration values for this enumeration according step NO TAG and step NO TAG
7. Additional enumerations shall be created by continuing at step NO TAG
8. Press **Commit** to make the enumeration definition persistent.
9. Press **Back to the Browser** .

5.5.5 Creation of Engineering Units

Engineering Units are a special kind of enumerations. The enumeration name for engineering units has to be 'UNITS'. The enumeration name is created following section NO TAG, but without any enumeration values. To create the enumeration values, the menu item **Edit–>Engineering Units** has to be executed. A screen form is displayed allowing the definition of the enumeration values and their representation (e.g. km = 1000 m).

The sequence of the engineering units is important, all units that are used for the representation of a new enumeration unit have to be defined first.

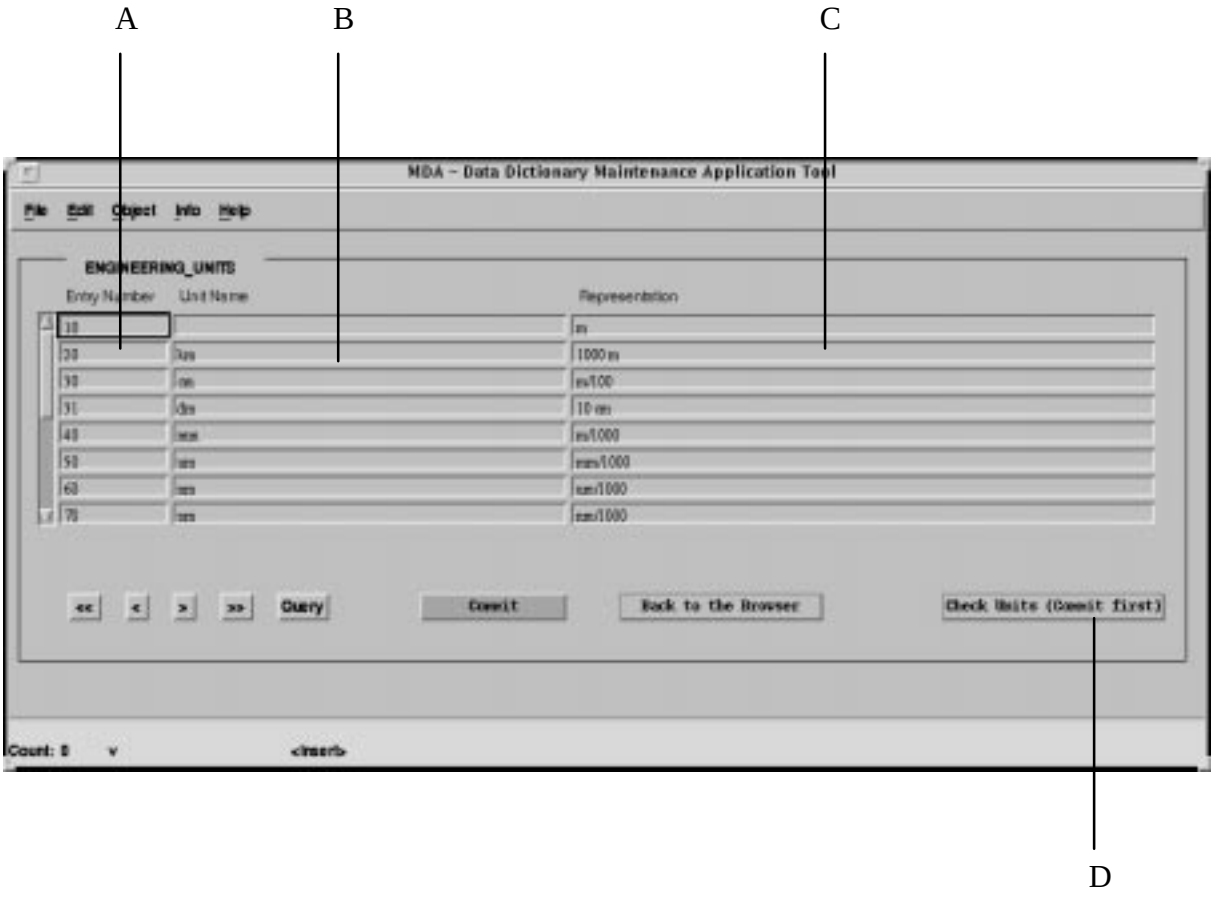

Figure 12. Engineering Units

A *Entry Number*

The Entry Number defines the position of the engineering unit.

B *Unit Name*

Name of the engineering unit. This field is optional. In case not a new unit but a term of units shall be defined, this field is left empty.

C *Representation*

The representation defines a new unit name or describes a term of units.

D *Check Units*

The Check Units button invokes the engineering unit checker in order to check if all engineering units and representations are correctly defined. The result of the check is displayed at the screen.

☞ *The standard CGS MDB Version already defines engineering units following the International Standard ISO 1000 ”SI Units and recommendations for the use of their multiples and of certain other units”.*

☞ *All engineering units defined here can be handled by all CGS products (e.g. CLS, CSS etc.).*

5.5.6 Creation of Attributes

Attributes are the further decomposition of aggregates. All attributes belonging to an End–Item are grouped according to the aggregate structure. An attribute is in any case related to an Aggregate.

5.5.6.1 General Attribute Data Definition

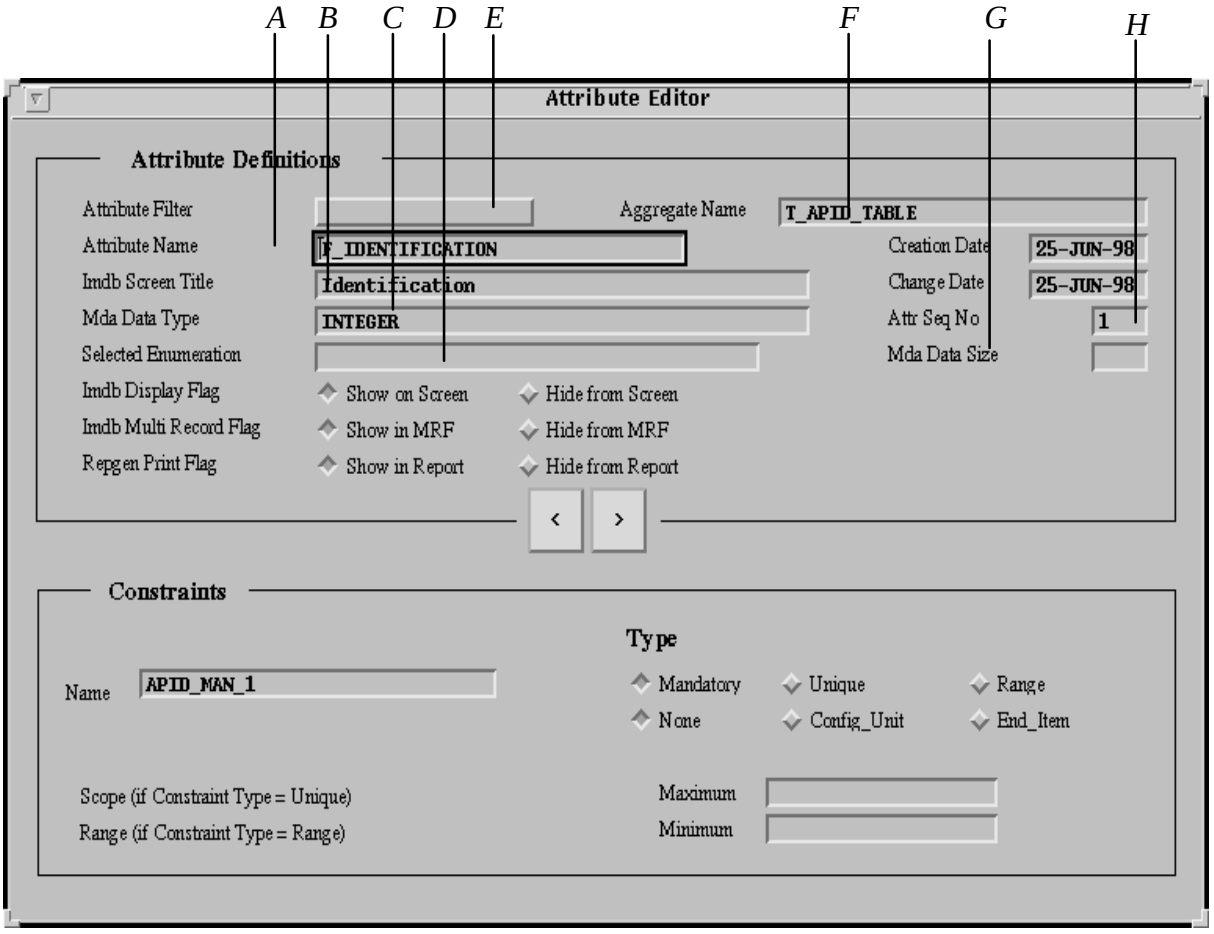

Figure 13. Attribute Editor

A *Attribute Name*

Specifies the attribute name. The attribute itself is defined by the content of all other fields in the attribute editor.

B *Imdb Screen Title*

The Screen Title is the description of the attribute within the end item aggregate editor in I_MDB.

C *Mda Data Type*

The field is for the definition of the attribute data type. All predefined and available MDA basic data types can be selected from a list.

D *Selected Enumeration*

If the data type Enumeration has been selected for the attribute, the enumeration has to be specified in this field. A selection from the predefined enumeration list is necessary.

E *Filter Function*

Allows to perform a selection of displayed attributes, starting , ending or including a specified substring. The scroll bar defines the view when the 'all object' mode is selected and the scroll is performed by the scroll buttons at the bottom of the window.

F *Aggregate Name*

Each attribute has to be related to an aggregate. It is a one to one relation. The related aggregate can be selected from a list of all available aggregates.

G *MDA Data Size*

The MDA Data Size is related to the selected Data Type. The maximum number of e.g. characters of a string can be specified.

H *Attribute Sequence Number*

The attribute sequence number defines the ordering of the attribute belonging to the aggregate.

To create a new Attribute

1. Select an ***Attribute*** field within the attribute scroll list and press **Edit** to enter the 'Attribute Editor' window.
2. Use **Object–>Insert**
   or
   the scroll arrows ◀ || ▶ at the bottom of the window to create a new entry.

A new Attribute with the name NEW_ATTRIBUTE will be inserted by default when selecting an empty field.

3. Select the ***Attribute Name*** field and press **Ctrl+u** to delete the default attribute name.
4. The ***creation date*** and ***change date*** have default values which may not be changed.

The change date will be updated each time when you change the attribute and executed commit.

5. Select the ***IMDB Screen Title*** field and define the screen title that IMDB shall use at run–time for the detailed data window.
6. Select the ***Attr Seq No*** field and define the order of the attributes that IMDB shall use at run–time for the detailed data window.
7. Select the ***MDA Data Type*** field and press **Ctrl+l** to invoke the selection list for the Attribute Data Type field.
8. Select a data type from the list and press **Ok**.

The assignment of an enumeration will only be executed if the attribute has as enumeration data type. For this, the next two steps have to be performed.

9. Select he ***Selected Enumeration*** field and press **Ctrl+l** to invoke the selection list for the existing enumerations.
10. Select an enumeration from the list and press **Ok**.

The definition of enumerations is explained in detail within the chapter NO TAG.

11. Select / Deselect the ***IMDB Display Flag*** field to define whether the attribute shall be shown in the IMDB detailed data window or shall be hidden.
12. Select / Deselect the ***IMDB Multi Record Flag*** field to define which attribute field out of the aggregate definition shall be used as an IMDB menu entry point to the IMDB detailed data editor window.

When selecting the multi record flag a record will be created which generates a scroll list in I_MDB. This scroll list contains all attributes marked by this flag. The items in this scroll list are entry points to the data editor window in I_MDB.

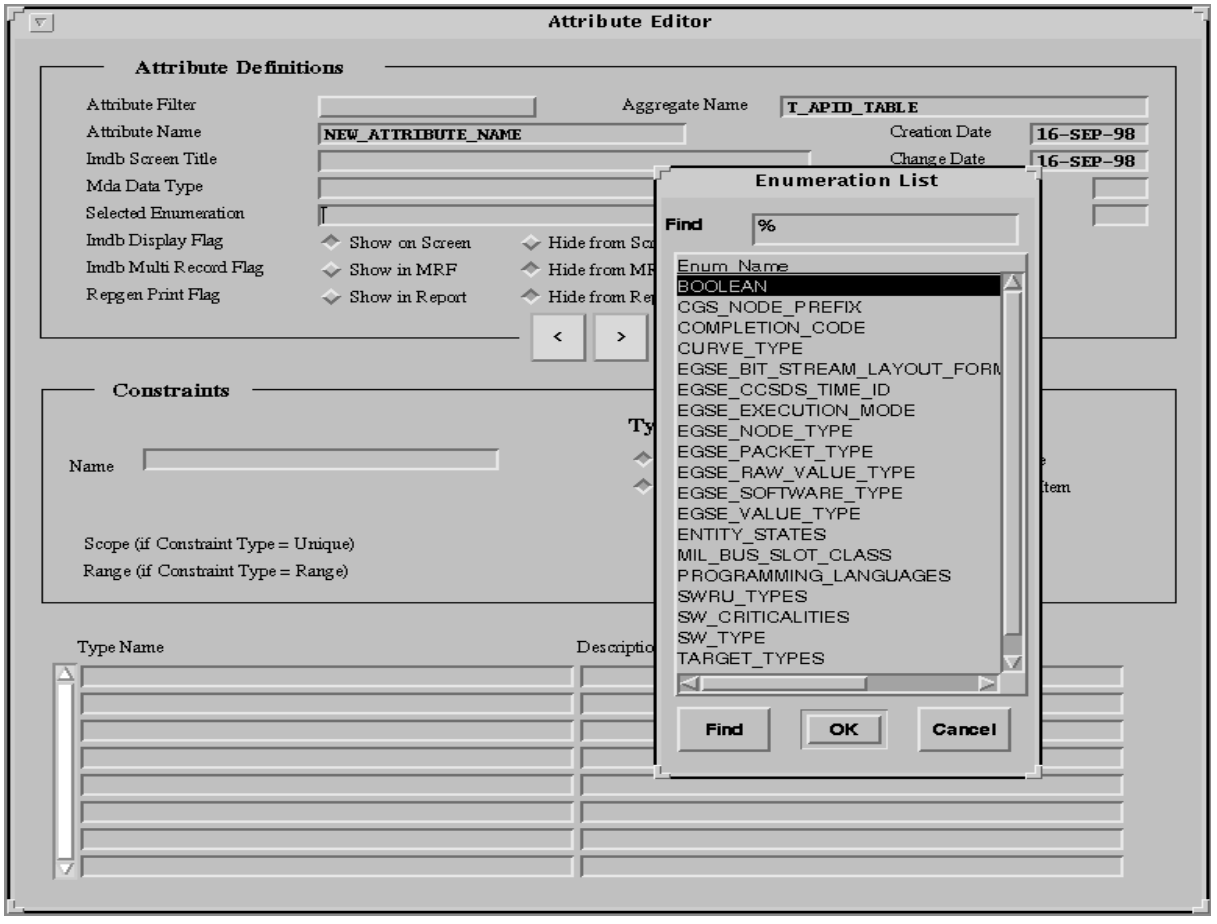

Figure 14. Enumeration specification within attribute definition

13. Select / Deselect the ***Repgen Print Flag*** field to define whether the attribute shall be printed in the ”all–details–report” or not.
14. If the attribute is of the supertype VARSIZE, the ***MDA Data Size*** field has to be filled–in. Values between 1 and 255 are allowed.
15. The definition of ***Constraints*** is described in the next section more in detail.

16. The ***Related Types*** field is for the definition of an attribute to aggregate relation. This relation is mandatory. It is explained in detail in chapter NO TAG.

17. Select **File–>Commit** from the main menu
or
press the **Commit** button in the browser window to make your changes persistent.

18. Select the ***Attribute scroll list*** in the browser area and press **Refresh** to update the Attribute scroll list.

5.5.6.2 Definition of Constraints

For all Attributes you can specify constraints. A detailed description of the constraint is given below. As also indicated below, the characteristics of the constraints are 'MDA data–type' dependent. For the constraints of STRING, PATHNAME and ENUMERATION examples are given as part of the detailed attribute definition.

- Constraint for the STRING type: you can specify the mda data type size.
- Constraint for the PATHNAME type: you can define the allowed end item type and a description of the path.
- Constraint for the ENUMERATION type: you can select an available enumeration or define a new one. How to define an enumeration is described below.

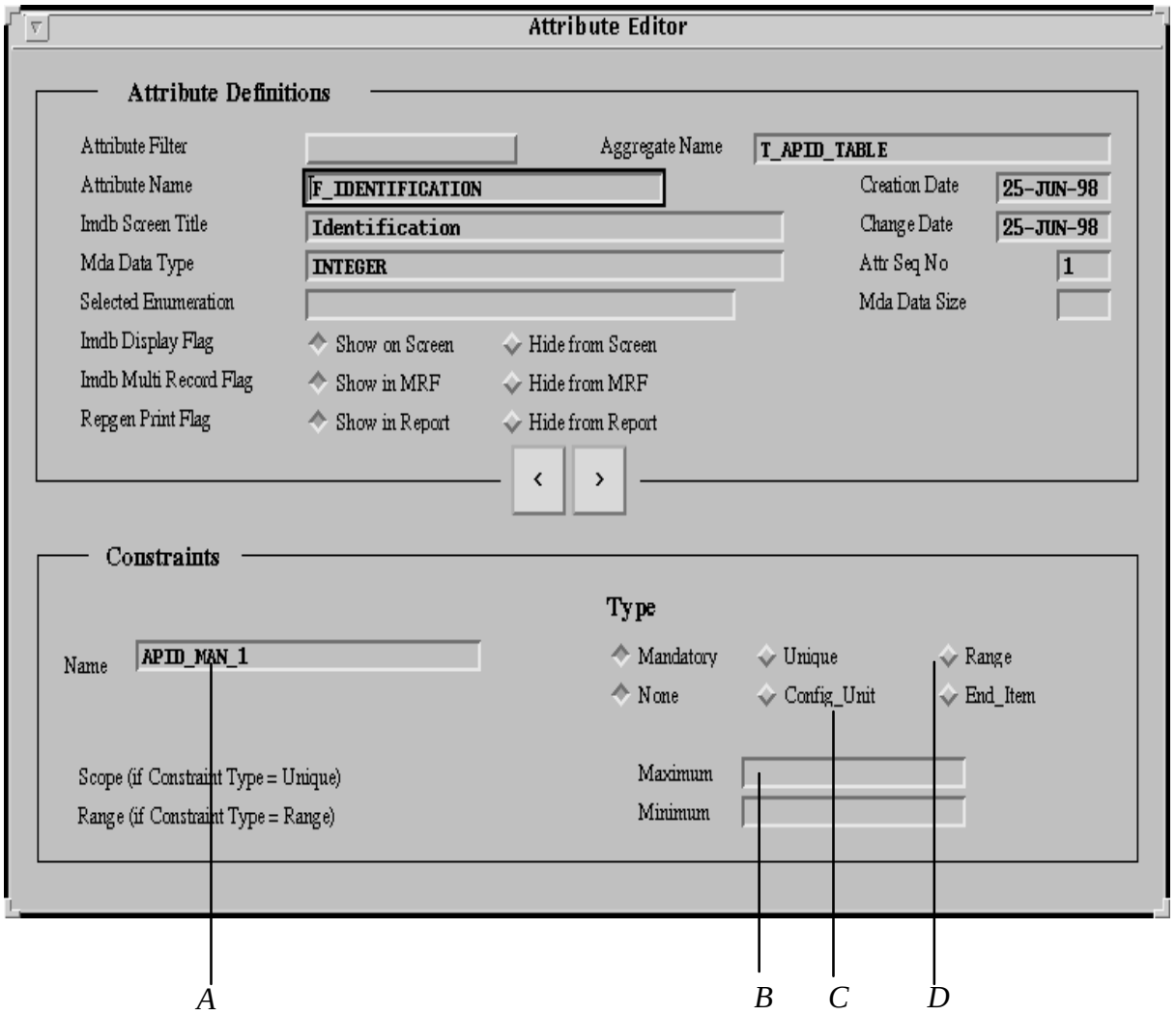

Figure 15. Constraint Definition in the Attribute Editor

A *Name*

Specifies the Constraint name. The constraint name must be unique.

B *Minimum and Maximum Range*

If a range constraint has been defined for the attribute, the minimum allowed value and the maximum allowed value input within I_MDB has to be defined.

C *Scope*

It can be specified if an attribute shall be unique within a Configuration Unit or within an End–Item.

D *Type*

Three different constraints may be defined for an attribute. Mandatory means, that the attribute has to exist, Unique means, that the attribute is unique within a scope, and Range means, that the attribute value must be entered in a predefined range.

To define Constraints

It is prerequisite that the attribute editor is on the screen.

1. Select the constraints ***Name*** field and define the name of the constraint that IMDB shall use at run–time in order to have meaningful error messages e.g constraint ”heat–sensor raw value range”.

2. Select one of the three ***Type*** flags to define which constraint type depends on the ’MDA data type’ .

- Constraint – ***Mandatory***: defines that this attribute field has to be filled–in. The Consistency Checker does perform a mandatory check based on this attribute constraint definition.

- Constraint – ***Unique***: defines that this attribute field has to be unique within the selected scope. The scope can either be a Configuration Unit or End–Item.

  Set ***Scope*** flag to ***Config_Unit*** or ***End_Item*** depending on your application if the constraint unique has been selected.

  The End–Item scope is used in relation to the multi record aggre–gate. It defines that an End–Item value is unique, e.g. the X_VALUE aggregate has been as a multi record aggregate.

X_VALUE	Y_VALUE	
1	10	–>valid
2	15	–>valid
3	30	–>valid
3	38	–>not valid, constraint violation

- Constraint– ***Range***: defines that the range of the attribute field shall be limited to the Maximum and Minimum as filled–in. The range information is used by IMDB to check/ensure that only the defined range can be assigned to the attribute at input time.

  Define the ***Range Minimum*** value and the ***Range Maximum*** value if constraint range has been selected.

3. Select **File–>Commit** from the main menu
or
Press the **Commit** button in the browser window to make your changes persistent.

For one attribute up to three different constraints can be defined, mandatory, unique and range. If more than constraint shall be defined, perform the next steps:

4. Select the constraint ***Name*** field. It is prerequisite that on constraint has been created.

5. Press the **arrow down key** to get an empty constraint record.

6. Define the constraint.

7. Select **File–>Commit** from the main menu
or
Press the **Commit** button in the browser window to make your changes persistent.

A switching between different attribute constraints is possible with the arrow up / down key. The constraint name field must be selected for this operation.

5.5.7 Attribute Definition Examples

5.5.7.1 Attribute definition example : Integer

The following figure gives an example how to define an attribute–type (MDA data type) INTEGER (see figure below). A range constraint with the name COL_INTEGER_1 has been defined with the values Min=5 and Max=10.

All other data fields are described in the above chapter ”General Attribute Data definition”.

Figure 16. Attribute definition example INTEGER

5.5.7.2 Attribute definition example : Hexadecimal

The following figure gives an example how to define an attribute–type HEXADECIMAL. All data fields are described in the above chapter ”General Attribute Data definition”.

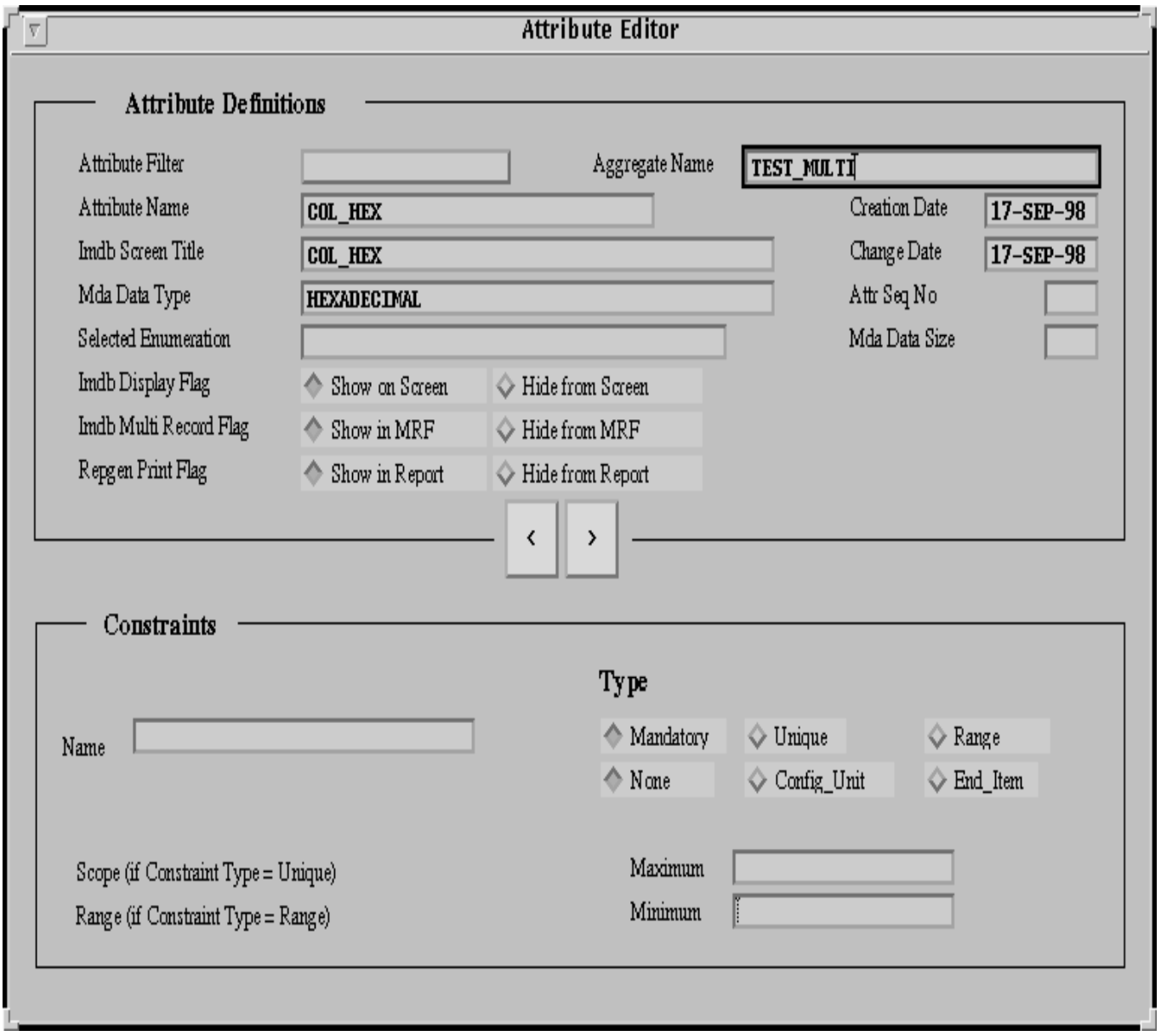

Figure 17. Attribute definition example HEXADECIMAL

5.5.7.3 Attribute definition example : Bitset

The following figure gives an example of how to define an attribute–type BITSET (see figure below). All data fields are described in the above chapter ”General Attribute Data definition”.

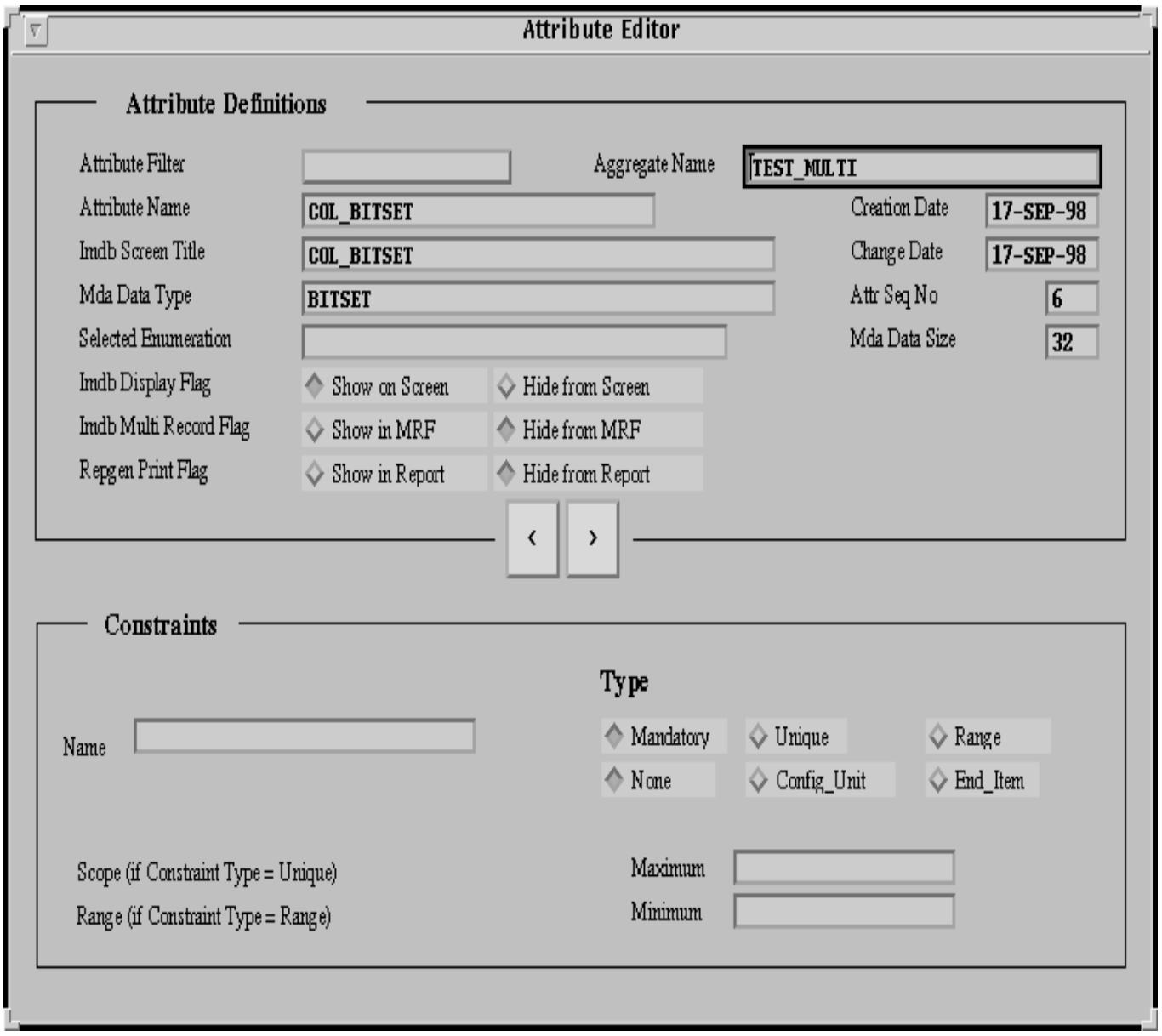

Figure 18. Attribute definition example BITSET

5.5.7.4 Attribute definition example : Enumeration

The following figure gives an example of how to define an attribute–type ENUMERATION (see figure below). An enumeration called BDE_STATE_CODES has been defined with the contents as shown in figure ”Enumeration definition form”.

1. An enumeration can be assigned by typing **Ctrl+l** within the attribute editor. A selection list is exposing and you can select a pre–defined enumeration and press **Ok** .

Another way to define an enumeration is to edit an existing or to define a new one.

2. Select the enumeration field and use the menu **Edit–>Enumeration**.

The enumeration edit form is exposing.

3. Scroll to the one you want to modify,
   or
   use **Query,**
   or
   use the menu **Object–>Insert** to get an empty scroll list entry.
4. After you have made your changes press **Commit** to make your enumeration definition persistent.
5. Press **Back to the Browser**.

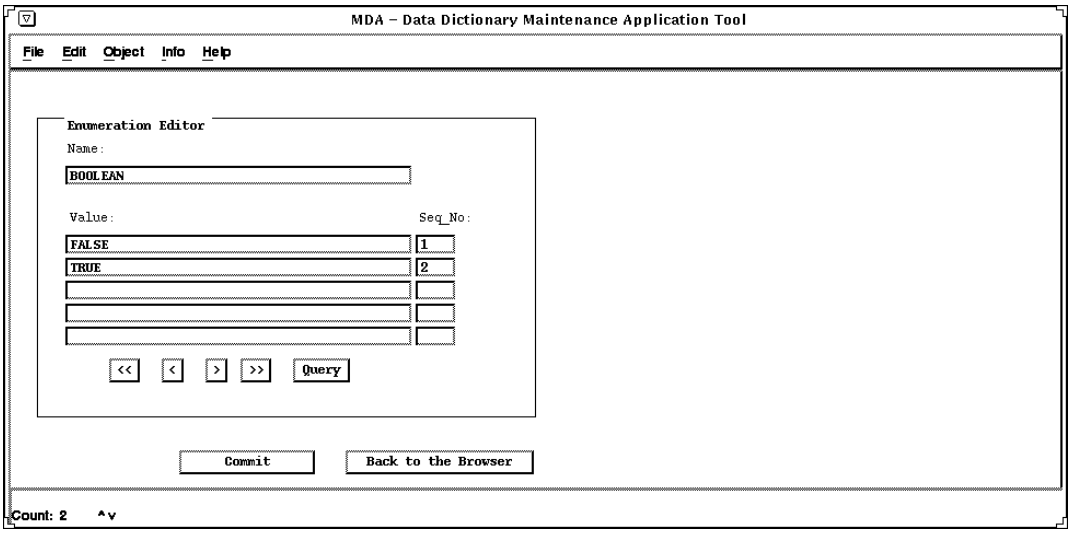

Figure 19. Enumeration definition form

All other data fields of the attribute are described in the above chapter ”General Attribute Data definition”.

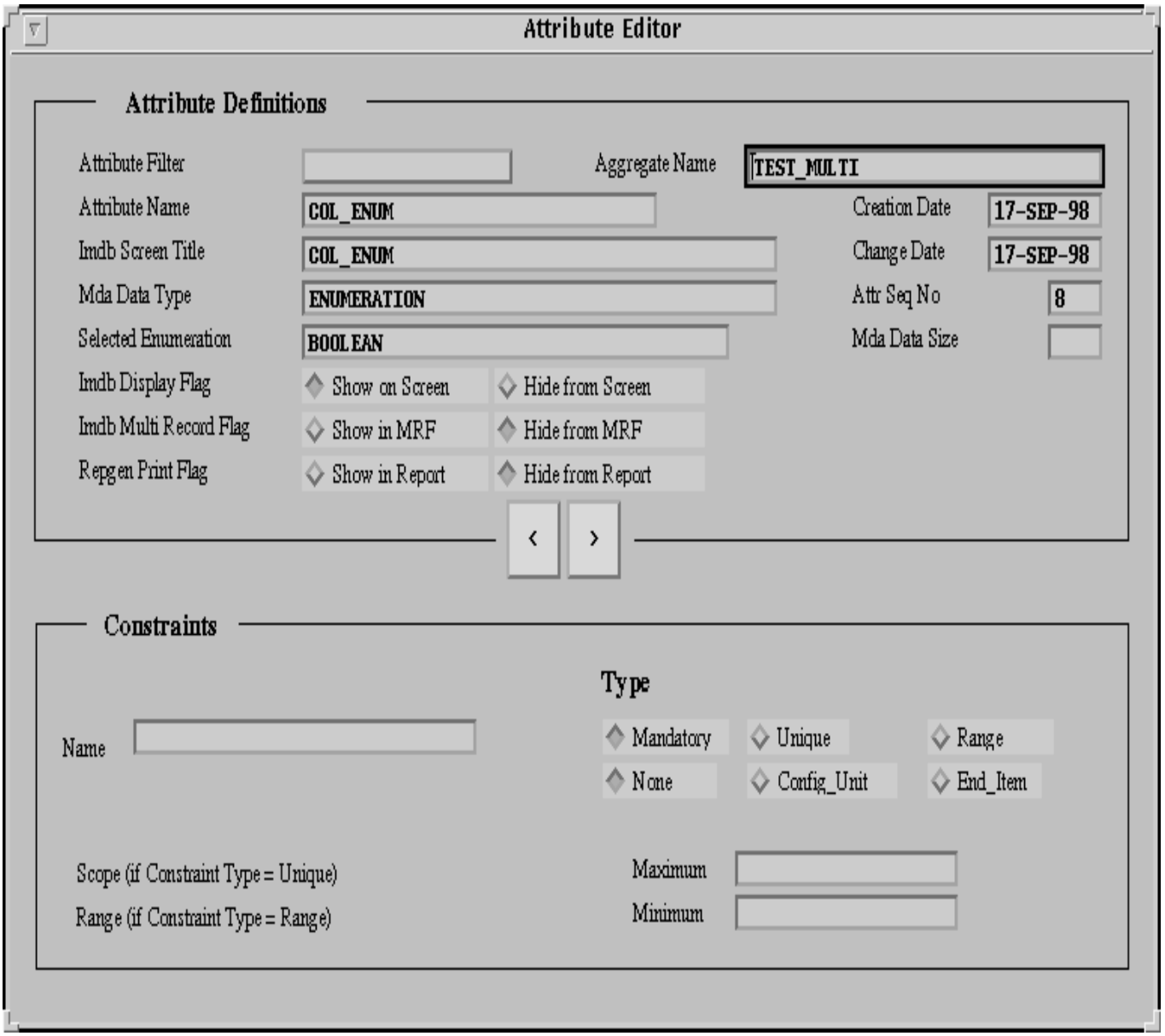

Figure 20. Attribute definition example ENUMERATION

5.5.7.5 Attribute definition example : Pathname

The following figure gives an example of how to define an attribute–type PATHNAME (see figure below). One or more End–Item Types may be assigned to a pathname attribute definition.

1. Select the ***Mda Data Type*** field within the attribute editor.
2. Type **Ctrl+l** to invoke the Mda data type list and select PATHNAME.
   end–item type selection list
3. Select the ***Show Details for:*** field.
   The attribute definition form will appear where type names and description may be added.
4. Select an empty type name field, press **Ctrl+l** to invoke the end–item type selection list and select an End–Item–Type from the list. Press **Ok** .
5. Select the related ***Description*** field and define a description if you want.
6. If you want to assign more than one End–Item–Type to this pathname attribute definition repeat the steps 4 and 5.
7. **Commit** the changes.

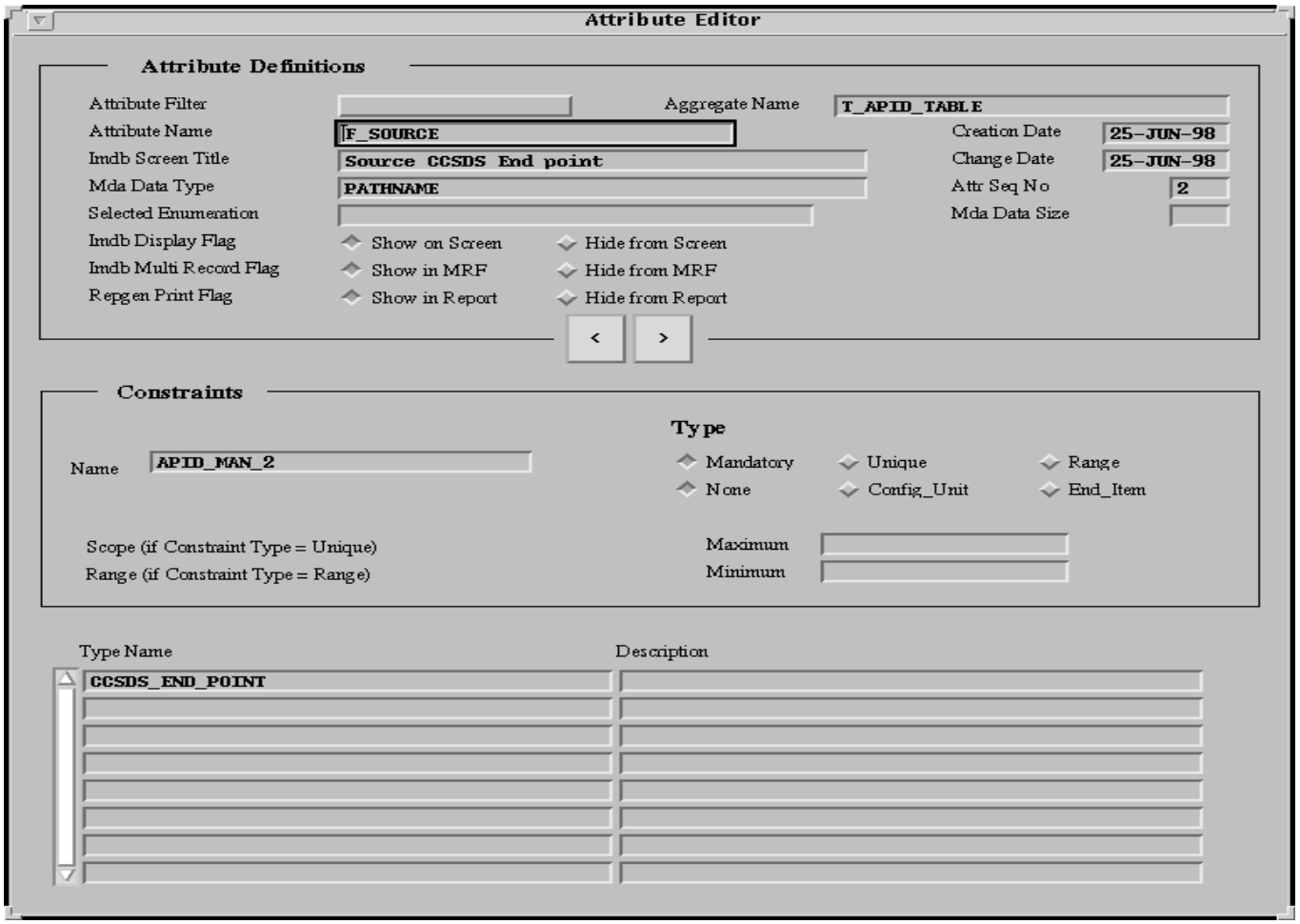

Figure 21. Attribute definition example PATHNAME

All other data fields are described in the above chapter ”General Attribute Data definition”.

5.5.8 Creation of Cross Reference Constraints

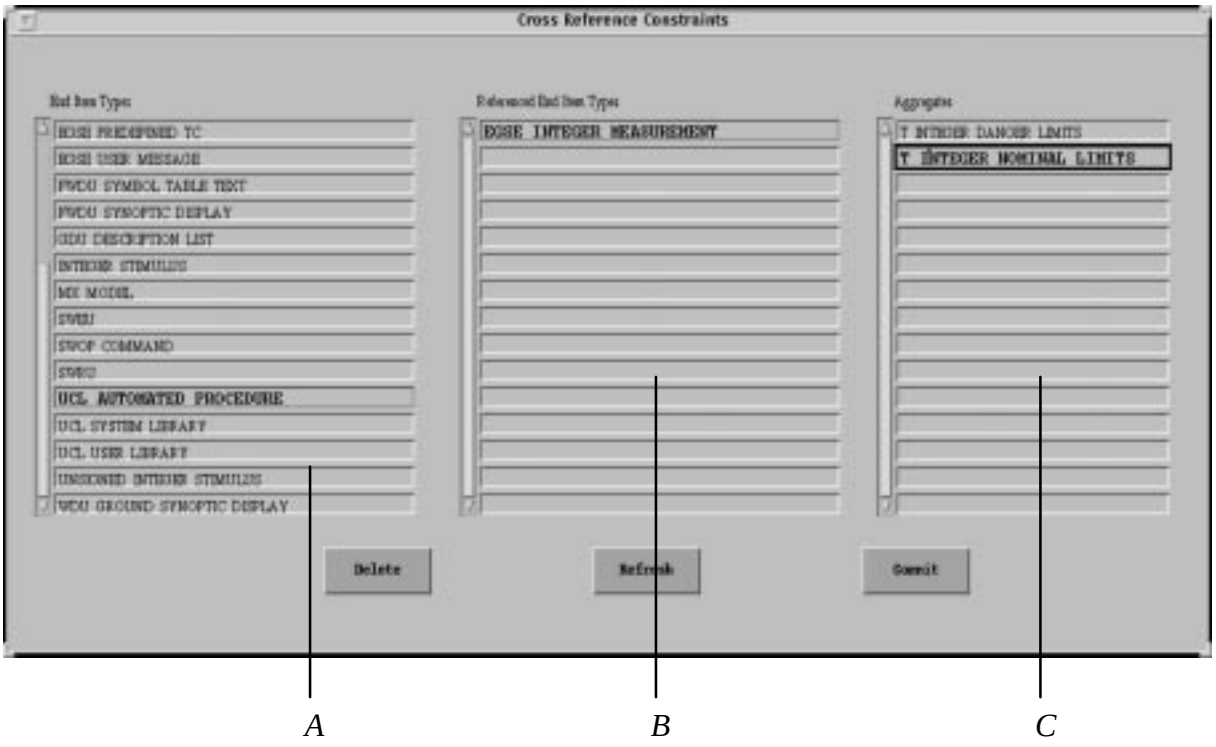

Figure 22. Cross Reference Constraints

A *End Item Types*

This column contains all end item types which contains a cross reference aggregate.
It is generated automatically and not by the user.

B *Referenced End Item Types*

Here are listed the refernced end item types of the selected end item type (Column A).

C *Aggregates*

Specifies the aggregates which belong to the selected referenced end item type (Column B).

To create Cross Reference Constraints

1. Select **Edit–>Cross Reference Constraints** from the main menu to open the Cross Reference Constraints Window.
2. Select an ***End Item Type*** field within the End Item Type scroll list.
3. Select an ***Referenced End Item*** field an type **Crtl+l** to pop up the end item type selection list.
4. Select the end item type by **double click**
   or
   select the end item type and press **OK**.

5. Select an ***Aggregate*** field within the Aggregate scroll list and type **Crtl+l**.
6. Select the aggregate by **double click**
   or
   select the aggregate and press the **OK** button.
7. Press the **Commit** button to make your changes persistent.

To delete Cross Reference Constraints

1. Select the record in the aggregate scroll list.
2. Press the **Delete** button.
3. Press the **Commit** button to make the changes persistent and the **Refresh** button to update the lists.

☞ *Please not, only entries in the aggregate scroll list can be deleted.*

5.6 Object Relations

Three relation types can be implemented:

- End–Item–Type to Domain (optional)
- Aggregate to End–Item–Type (optional)
- Attribute to Aggregate (mandatory)

The Attribute to Aggregate relation must be performed in any case. The other relations are optional, but in general it makes sense to create relations also for them.

5.6.1 Creation of End–Item–Type to Domain Relations

End–Item–Type to Domain relations are optional, but in general it makes sense to create the relation. A 1 to N relation per End–Item–Type can be created. This means, that one End–Item–Type can be included in different domains.

To create an End–Item–Type to Domain Relation

1. Select an ***End–Item–Type*** field (may be an empty one in case of a new relation) in the browser window and press **Edit** to pop up the End–Item–Type editor.
2. Select the ***Related Domains*** list on the right side of the type editor and pop up the domain selection list by typing **Ctrl+l**.
3. Select the to be related domain by a **double left** mouse click on the related domain
   or
   select the domain and press the **OK** button.
4. Select **File–>Commit** from the main menu
   or
   press the **Commit** button in the browser window to make your changes persistent.
5. Select the ***End–Item–Type scroll list*** in the browser area and press **Refresh** to update the End–Item–Type scroll list.

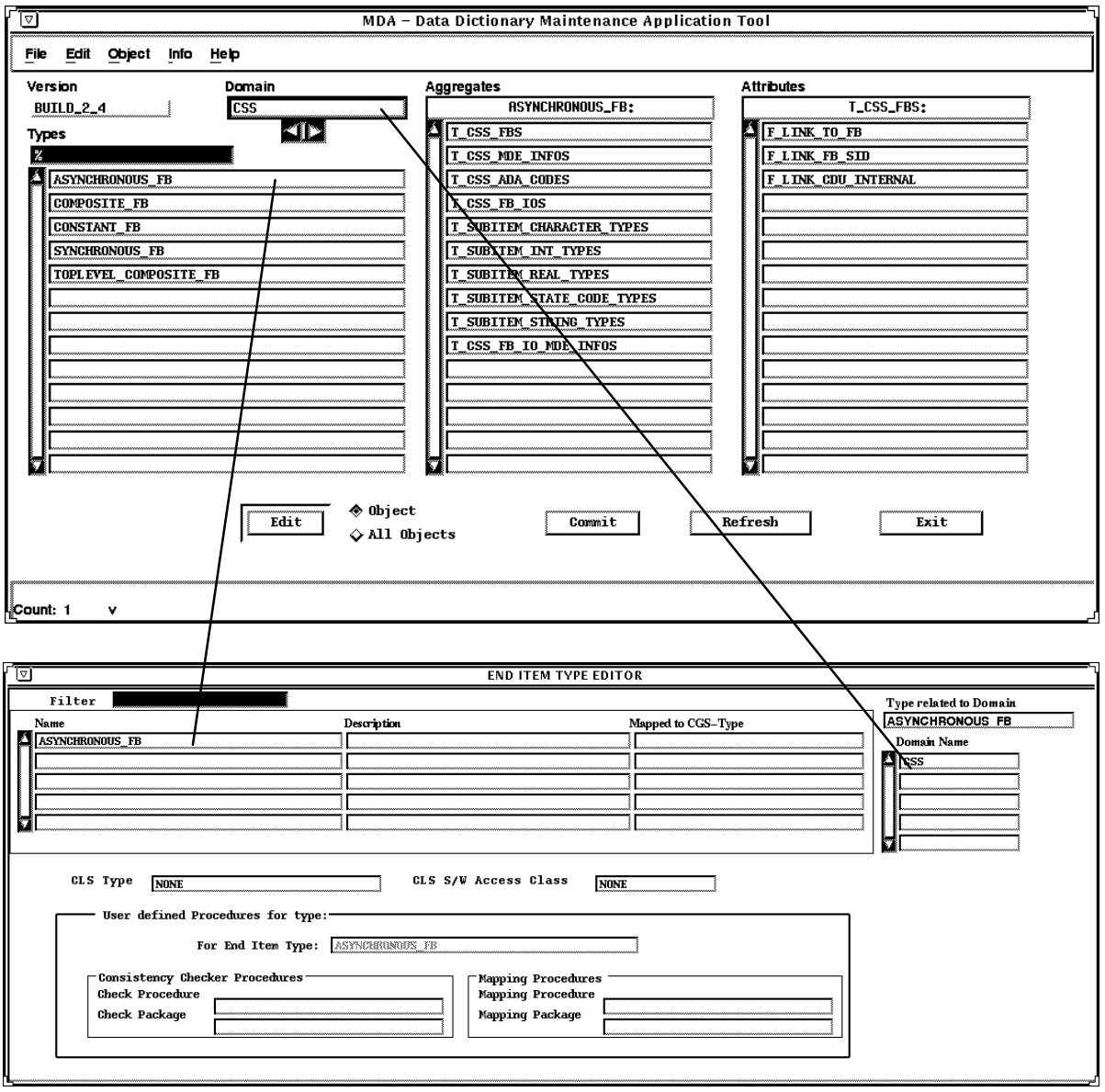

Figure 23. Relate an End–Item–Type to a Domain

☞ Please note that the related domain scroll list editor is an 1:n relation per End–Item–Type. This means that the shown list of related domains does always belong to a single End–Item–Type in the Edit Window. To experience the effect you should select one End–Item entry after the other and see that the related domain list changes with every selection of the End–Item–Type list.

5.6.2 Creation of Aggregate to End–Item–Type Relations

Aggregate to Domain relations are optional, but in general it makes sense to create the relation. A 1 to N relation per Aggregate can be created. This means, that one Aggregate can be included in different End–Item–Types.

To create an Aggregate to End–Item–Type Relation

1. Select an ***aggregate*** (may be empty in case of a new relation) in the browser and press **Edit** to pop up the aggregate editor.
2. Select the ***Related Types*** list on the right side of the aggregate editor and pop up the type selection list by typing **Ctrl+l**.
3. Select the to be related type by a **double left click** on type
   or
   select the type and press **Ok**.
4. Select the ***Agg. Seq. No.*** field and define the Aggregate Sequence Number.

The aggregate sequence number defines the order of the aggregates belonging to the End–Item–Type. The first sequence number has to have the value one.

5. Select the ***Agg. Flag*** field.

A selection list with the two items MANDATORY and OPTIONAL is popped up. If this flag is mandatory, this aggregate has to exist. Optional means, that the aggregate might exist or not. In any case, one of these two options has to be selected.

6. Select **MANDATORY** or **OPTIONAL** from the selection list.
7. Select **File –> Commit** from the main menu
   or
   click on the **Commit** button in the browser window to make your changes persistent.
8. Select the ***Aggregate scroll list*** in the browser area and press **Refresh** to update the Aggregate scroll list.

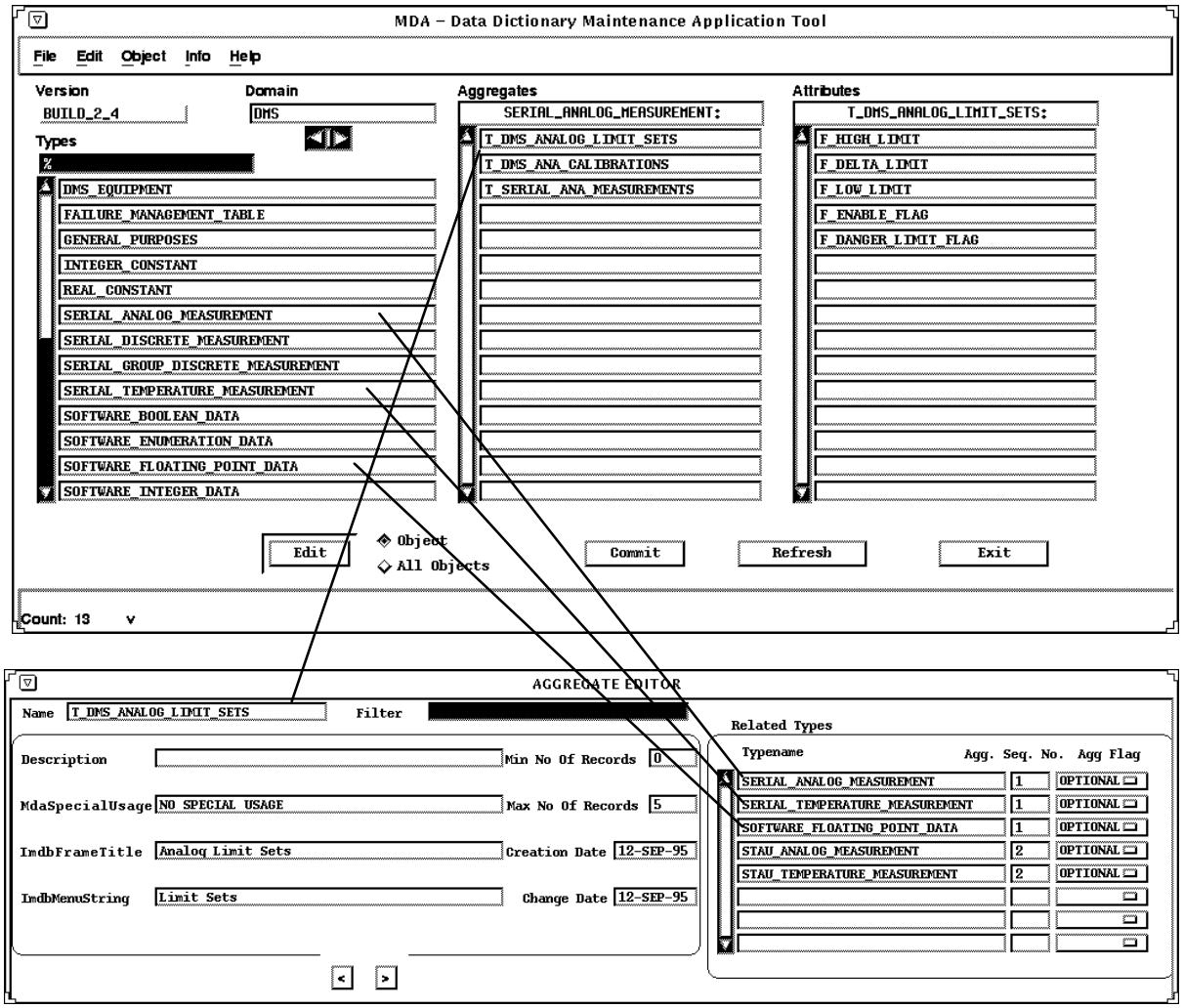

Figure 24. Relate an Aggregate to an End–Item–Type

☞ Please Note that the related End–Item–Type scroll list editor is an 1:n relation per Aggregate. This means that the shown list of related EI does always belong to a single Aggregate in the Edit Window. To experience the effect you should select one Aggregate after the other and see that the related EI list changes with every selection of the Aggregate list.

5.6.3 Creation of Attribute to Aggregate Relations

Attribute to Aggregate relations are mandatory, they have to be created in any case. It is a 1 to 1 relation per Attribute. This means, that one Attribute is related to one and only one Aggregate.

To create an Attribute to Aggregate Relation

1. Select an ***attribute*** (may be empty in case of a new relation) in the browser and press **Edit** to pop up the attribute editor.
2. Select the ***Aggregate Name*** field in the upper right corner of the attribute editor and pop up the aggregate selection list by typing **Ctrl+l**.
3. Select the to be related aggregate by double click on the related aggregate
   or
   select the aggregate and press the **OK** button.
4. Select the ***Att. Seq. No.*** field and define the Attribute Sequence Number. This value does define the order of the Attribute belonging to the Aggregate.
5. Select **File–>Commit** from the main menu
   or
   press the **Commit** button in the browser window to make your changes persistent.

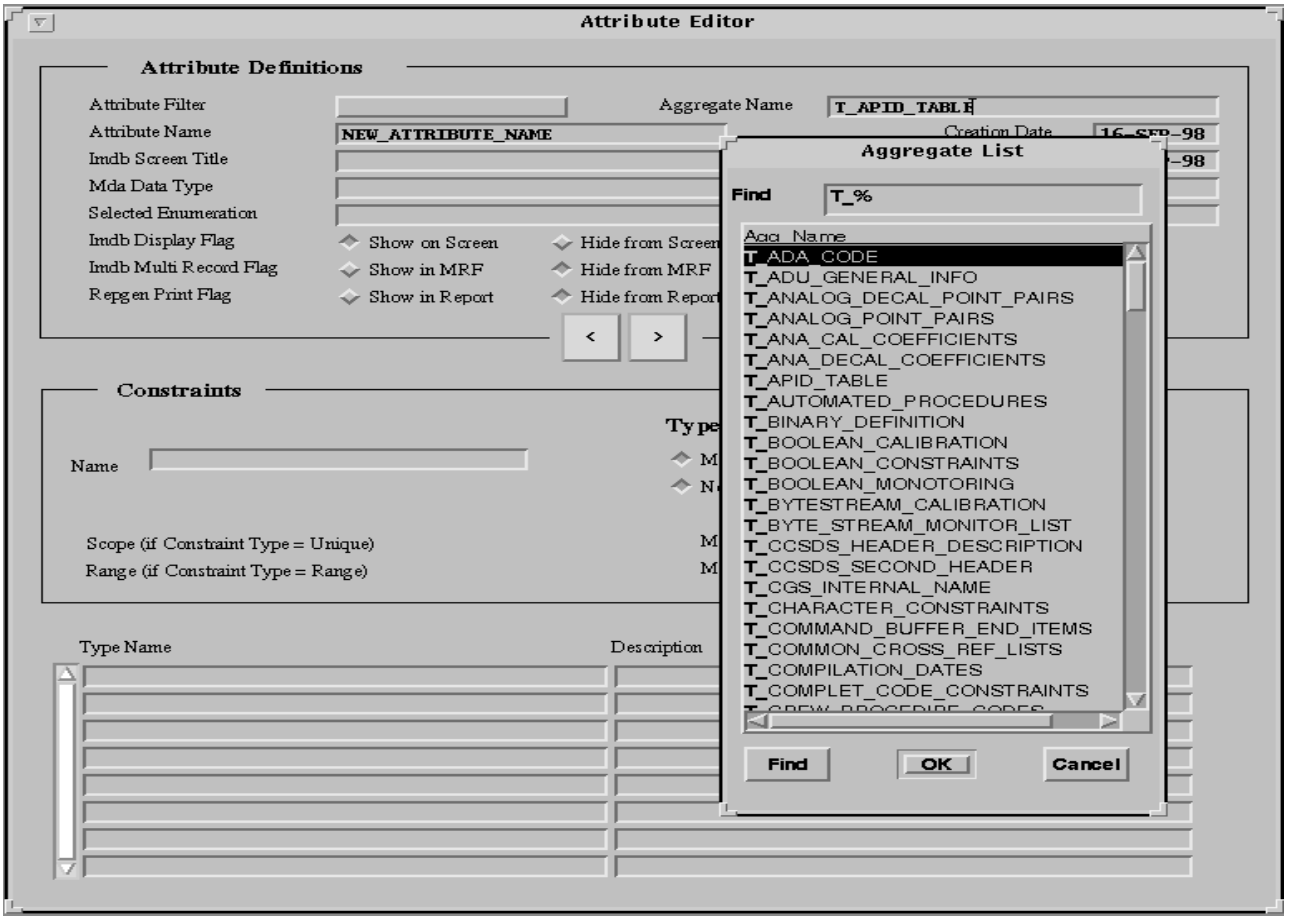

Figure 25. Relate an Attribute to an Aggregate

5.6.4 Deletion of Relations

Any existing relation can be deleted. The deletion operation is executed from the respective End–Item, Aggregate, or Attribute Editor window.

To delete Object Relations

1. Select the related ”item” (either end–item, aggregate or attribute) in the corresponding Browser Window scroll list.
2. Select **Object–>Delete** from the main menu to delete the relation.

☞ Please note, only the relation is deleted not the item itself.

3. Select **File–>Commit** from the main menu
   or
   press the **Commit** button in the browser window to make your changes persistent.
4. Select the related ***scroll list*** in the browser area and press **Refresh** to update the scroll list.

5.7 Print Reports

5.7.1 Default Printer

All print report are send to the specified default printer. The default printer can be changed.

To setup a default printer

1. Select **Edit–>Preferences** from the main menu to pop up the Preferences options.
   The preferences view shows up in the main browser window.
2. Select the field ***Default Printer*** and type the name of the printer.
3. Press **Apply** to activate the new preferences.
4. Select **Back to the Browser** to show up the main browser window.

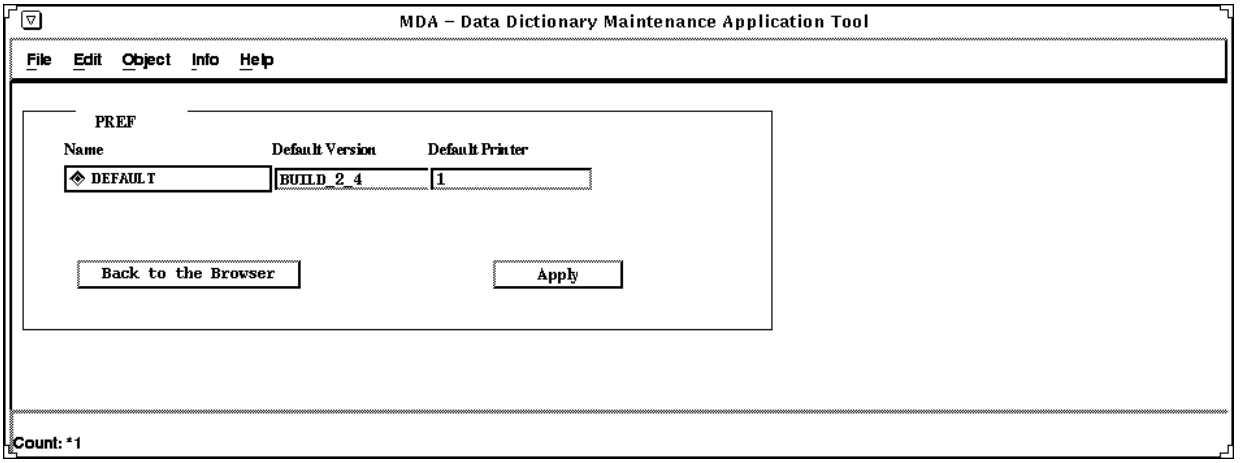

Figure 26. Define a default printer

5.7.2 Printing Reports

Different kinds of reports can be created and send to the default printer. There is the option to print a selection of tables, to print all existing tables or to print the type tree.

To print Reports

Within the report dialogue window the user can select predefined reports for every DADI table.

1. Select **File–>Print** from the main menu to pop up the print reports settings window.
2. Select the print option:
   - ***Print selected tables***: for selected information as presented in the ”Selected Reports” area
   - ***Print All***: to print all tables (could take some time)
   - ***Print Type Tree***: to print a top–down type tree.
3. Select the ***Reports*** you want to print within the ’Selected Reports’ frame using the radio buttons.
4. Select the print style
   - ***Interactive***:
     The report writer is started in interactive mode. You can select the destination for the report (Preview,Mail, Screen File or Printer), the printer name and a filter for the version using the report writer.
   - ***Batch to default printer***:
     The report writer is started in batch mode. All printouts are made silently to the default printer with the default entries.
5. Press **Execute Report** to start the reports.

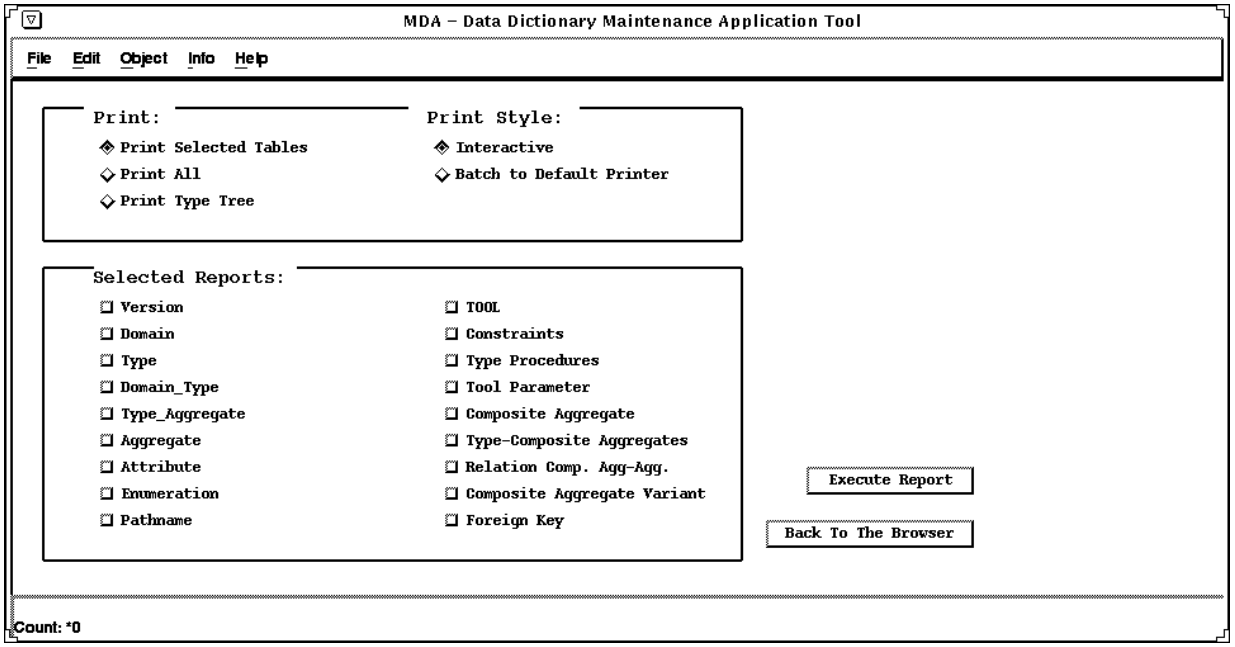

Figure 27. Define Report

6. **Wait** until the Report Parameter Form pops up after a while.
7. Define the parameters for the printing of the report.

☞ Please Note: that the standard Oracle report function is used here. For explanation about the handling, please refer to the Oracle User Manual.

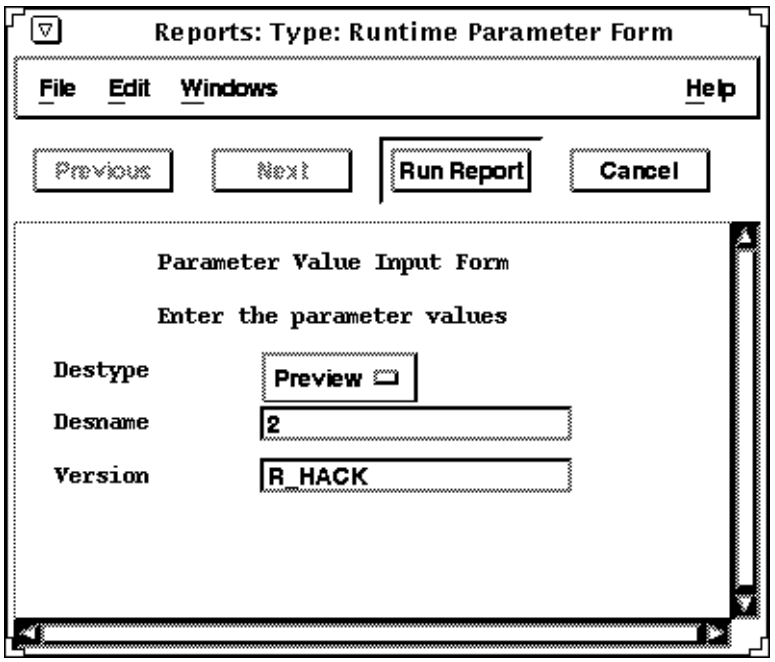

Figure 28. Print parameter definition

8. Press **Run Report** to start the report preview or print depending on the Destype setting.

9. **Wait** until you have got the print or until the previewer appears on the screen.
10. Press **Close** within the previewer if preview have been selected to return to the report selection window.
11. Press the button **Back to the Browser**.

A typical type report is shown in figure NO TAG

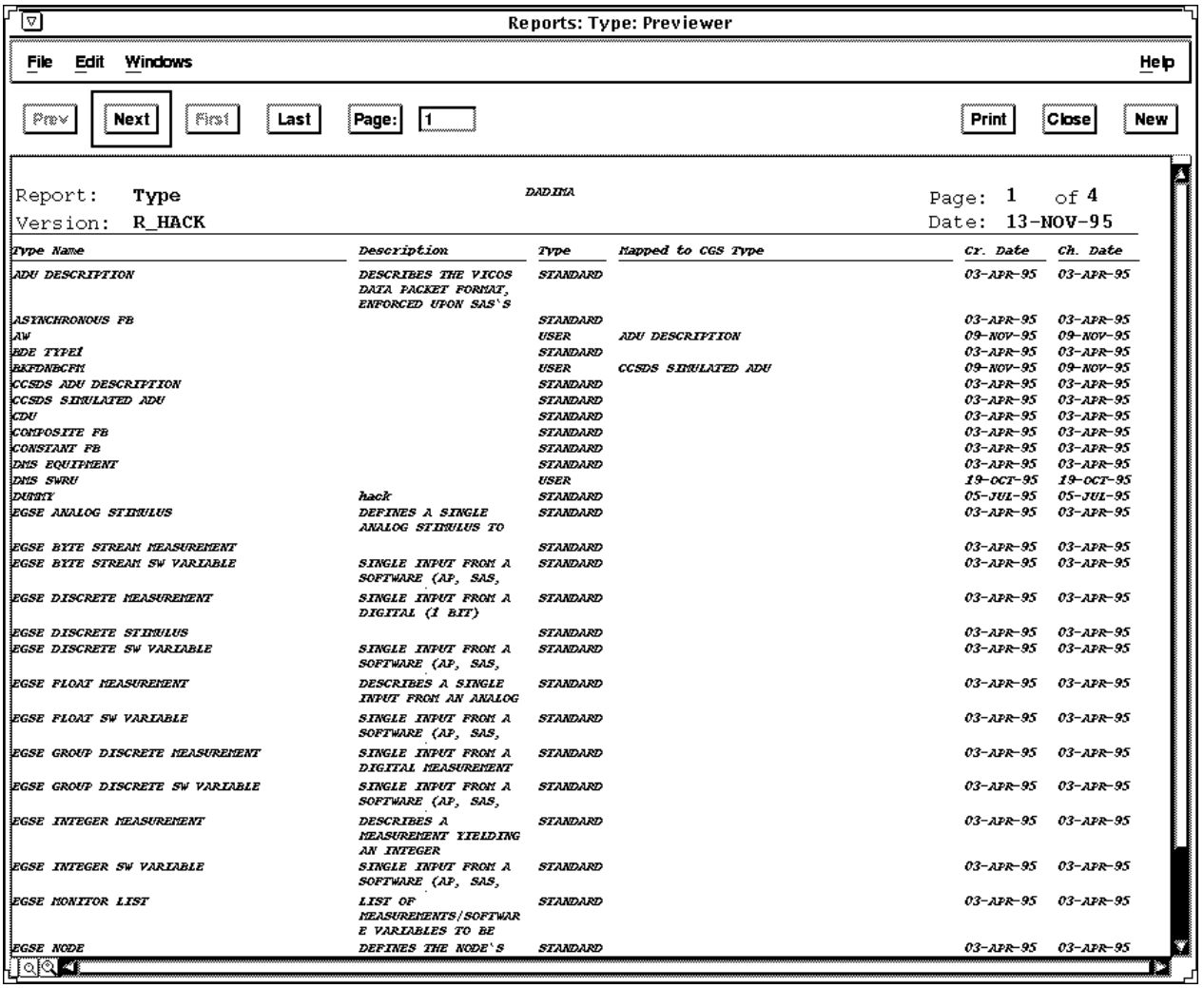

Figure 29. Type Report Example

5.8 Export to MDB

The following MDB files and tables can be generated from within the DADIMA tool:

- create MDB–Table scripts
- create API–View scripts
- create 'Create Aggregate Table' scripts
- create BDE Control Files

Export Data Dictionary Versions to the Mission Database (MDB) includes:

- **MDA Data Dictionary**
  - export of the Data Dictionary table contents
- **Data API End Item View Scripts**
  - creation of scripts that generate the End–Item views used by the Applications Programmer Interface (API).
- **MDA Create Table Statements, Aggegate Views and API–Write Procedures**
  - generation of Table Statements
  - generation of API Write Procedures
  - generation of Aggregate Views
- **Batch Data Entry (BDE) control files**
  - creation of Batch Data Entry used load data configuration description files

To export a Version to MDB

1. Select **File–>Export to MDB** from the main menu.
2. Select the ***Version*** field and pop up the version for export selection list by typing **Ctrl+l**.
3. Select the DD version which you want to export by double click on the related version
   or
   select and press the **OK** button.
4. Define which parts of the Data Dictionary should be generated using the check–boxes below the ***Create*** statement.
5. You don't have to define the field ***Difference Version*** because the Difference Report is not yet implemented.
6. Press the **Export to MDB** button to start the export process.

All export processes are started parallel to the DADIMA application.

7. The output is located in:

- Data Dictionary: $MDA_HOME/util/dadi/dadi_export/dadi_data/imp_mda_version.sql
- API: $MDA_HOME/util/dadi/dadi_export/api_views
- Create Tables: $MDA_HOME/util/dadi/dadi_export/aggregates/mdb and $MDA_HOME/util/dadi/dadi_export/aggregates/temp_mdb
- BDE Control Files: $MDA_HOME/util/dadi/dadi_export/bde_control_files

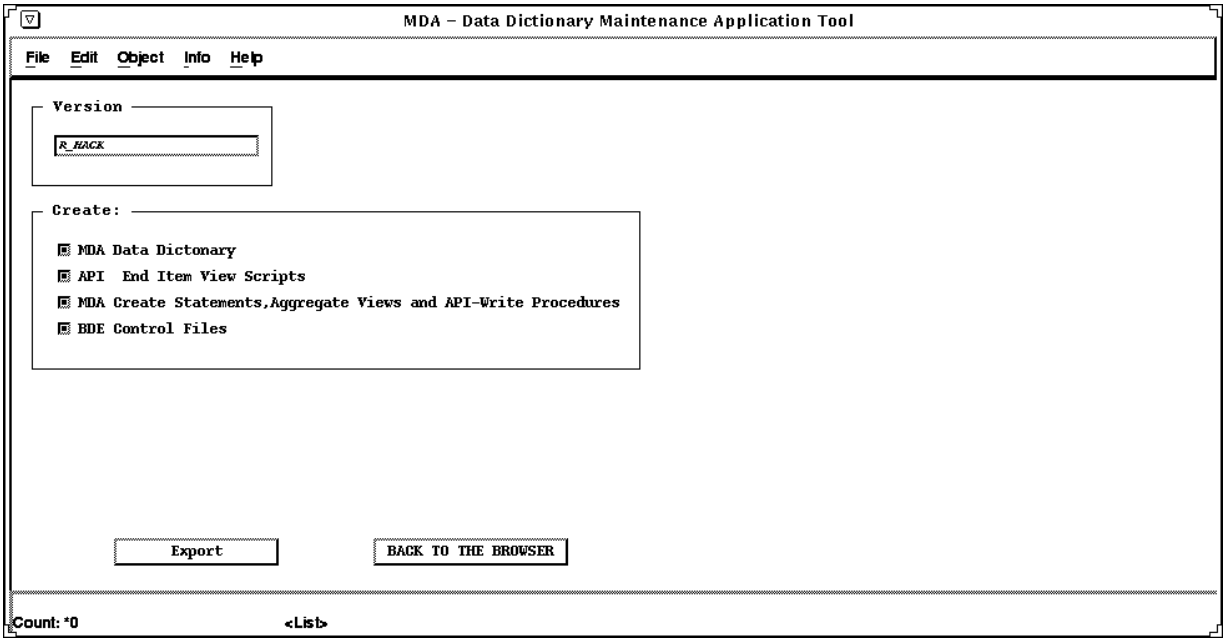

Figure 30. Export to MDB parameter definition

8. When there are no changes during the last export, a window with the message 'No changes to commit' will appear. Press **Ok** to continue.

Depending on the number of selections, different texteditor windows will appear. Each contains a log file for each performed export.

9. **Wait** until the last texteditor window is on the screen.

10. Make an inspection on the logs if necessary.

11. **Quit** the texteditor windows.

5.9 Tool Invocation Definition

The flexible tool invocation function of DADIMA allows to attach user application tools to CDU versions, CCU versions and End Items. It is a feature that allows the integration of tools to be started from IMDB. Interface parameter can be defined as well as the scope in which the tool shall be executable.

In the upper part of the window the tool itself will be specified by its name, command to start the application tool, etc. In the other parts the scopes can be defined in which the tool shall be available. It can be available for configuration units (CDU or CCU), so that whenever a configuration unit is selected in the configuration unit window, the tool appears in the menu of the <Command> menu button.
A tool can also be available for a specific End Item Type within a navigation scope (CCU and CDU) of the I_MDB navigator window. For **each** of the defined scopes customer defined parameters can be entered.

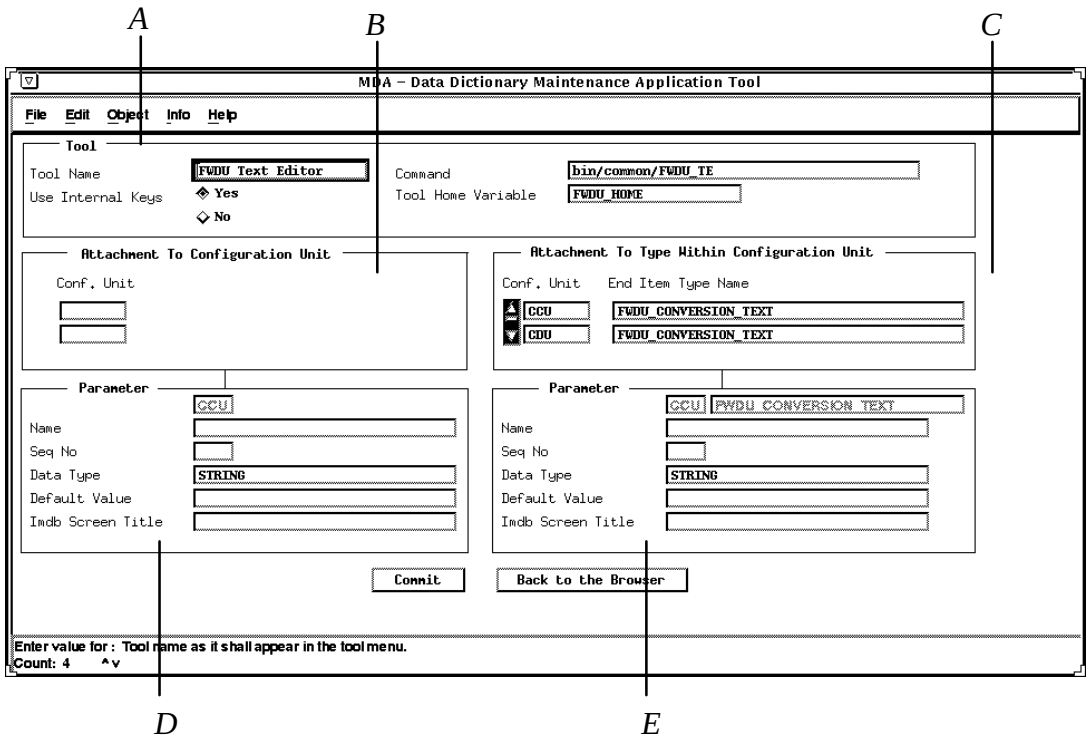

Figure 31. Tools Invocation Definition Editor window

A *Tool Definition*

Within this frame the tool invocation will be defined by the tool name for the I_MDB menu, the command string to start the tool and the tool home variable.

B *Attachment to Configuration Unit*

To attach the tool to a CCU version and/or to a CDU version.

C *Attachment to Type within Configuration Unit*

To attach the tool to an End–Item–Type.

D *Parameter*

Within this frame parameters may be defined, which shall be attached to the end of the standard parameter. This parameter frame is valid for the tool attachment to a CCU version and/or CCU version.

E Parameter

Within this frame parameters may be defined, which shall be attached to the end of the standard parameter. This parameter frame is valid for the tool attachment to an End–Item–Type.

To enter the Flexible Tool Invocation Definition window

1. Select **Edit–>Tools Invocation Definition** from the Main–Window–Menu to enter the editor.

To define the User Application Tool

Within the upper part of the Tool Invocation Definition window the tool attributes will be specified. All attributes are mandatory.

1. Select the ***Tool Name*** field and enter the name of the tool to be used.

This field will appear within the tool invocation menu of I_MDB as the menu title where the tool will be called for a CCU version, CDU version or an End Item.

2. Select the ***Command*** field and enter the command necessary to start the user application tool from the tool home directory.

This command is the same used to start your application from the Unix environment within the home directory of the tool.
You don't have to enter a slash in front of the tool name. If the tool is within a subdirectory you have to enter the name of the subdirectory in front of the tool name separated by a slash, e.g. *directory1/directory2/toolname*.

3. Select the ***Tool Home Variable*** field and enter the name of the environment variable holding the directory of the tool home directory.

The tool home variable normally will be defined in the unix *cshrc* file. You don't have to enter a slash at the end of the home variable. A definition for example looks like
setenv USER_TOOL_HOME /usr/application/user_tool.
The tool home variable connected with the command lead to an internal call like *$USER_TOOL_HOME/directory1/directory2/toolname.*

4. Select ***Use Internal Keys*** **Yes** or **No** depending wether the internal version keys are expected by the user tool or not.
   By selecting **Yes**, the internal version keys with the following format will be taken:

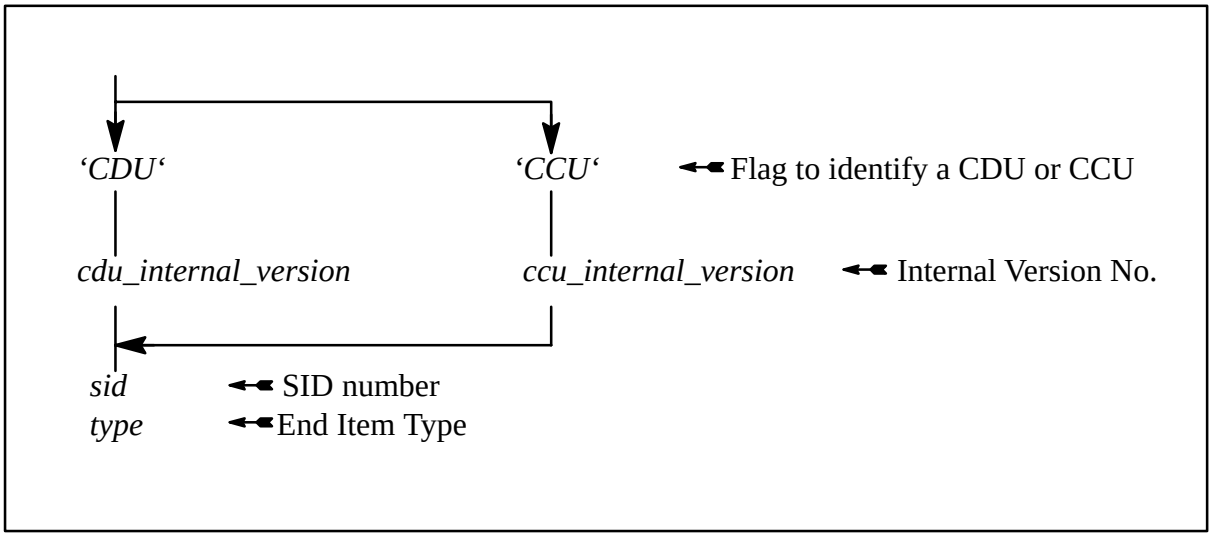

Figure 32. Usage of Internal Keys

An *example* of the result by selecting the internal keys version is:

'CCU'
'104711'
'11456'
'FLIGHT_DISPLAY'

By selecting **No**, the complete key sequence shown in I_MDB are taken with the following format will be taken:

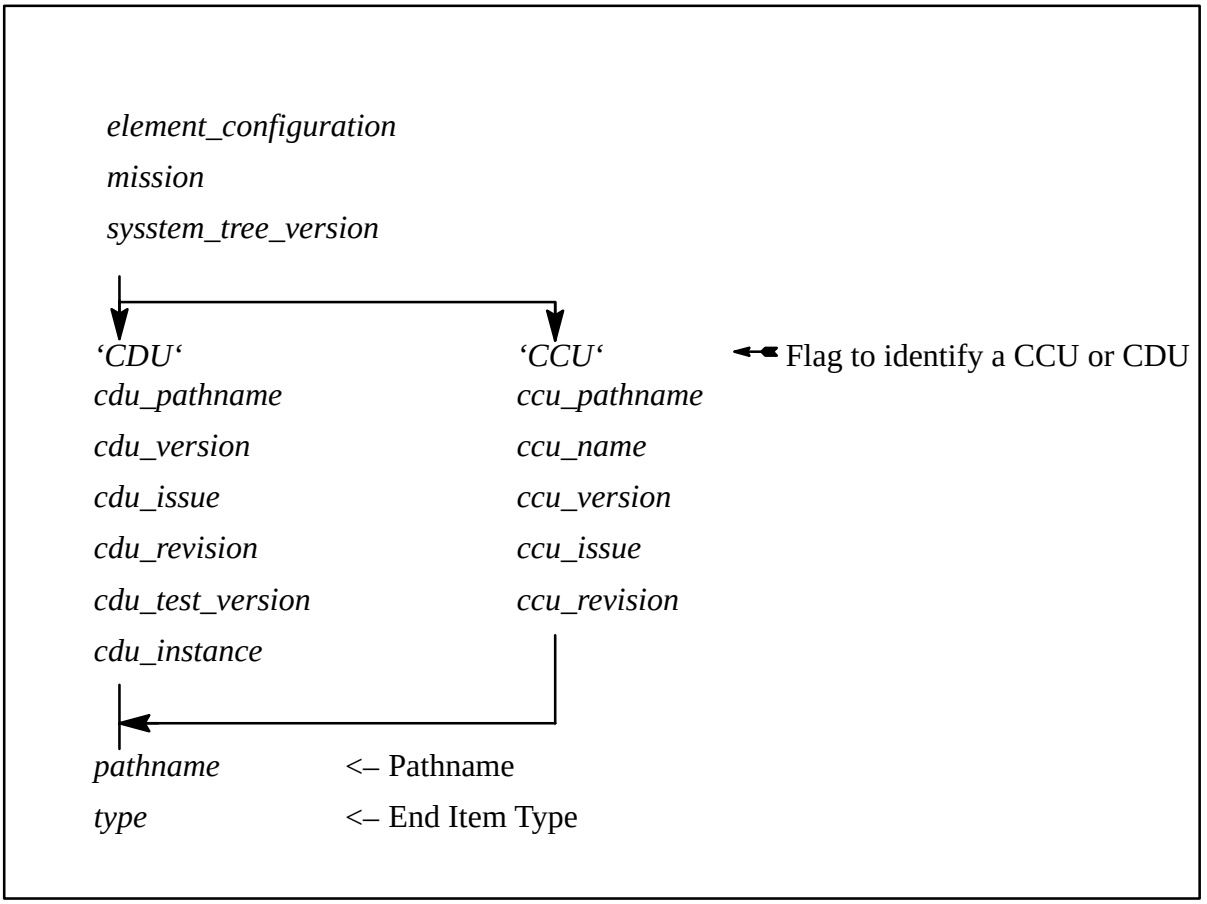

Figure 33. No Usage of Internal Keys

An *example* of the result by deselecting the internal keys version is:

for a CDU version where the End Item'ISSA' has been selected:

'DUMMY_MISSION'
'1'
'CDU'
'\ISSA\TEST'
'1'
'0'
'0'
'0'
'MDB_INSTANCE'
'\ISSA\TEST\DISPLAY1'

for a CCU version where the End Item'ISSA' has been selected:

'DUMMY_MISSION'
'1'
'CCU'
'\ISSA\ONBOARD'
'CCU_NAME'
'1'
'0'
'0'
'\ISSA\ONBOARD\DISPLAY1'
'FLIGHT_DISPLAY'

The end item is identified by its pathname or SID. If no end item has been selected, the pathname is empty and the SID is zero. Also the end item type is empty if no end item has been selected. All parameters are passed as strings.

To attach the User Tool to a Configuration Unit

The user tool may be attached to a CCU version **or / and** a CDU version. Attachments can be defined to a CCU version and a CDU version at the same time for one user application tool.
If there is an assignment to a CCU scope the tool menu appears in the CCU Versions window. If there is an assignment to a CDU scope the tool menu appears in the CDU Versions window.

On the middle and lower left part of the editor window, the parameters necessary to attach the user application tool to a CCU version and/or to a CDU version, will be specified.

1. Move the cursor to the first ***Conf.Unit*** field within the 'Attachment to Configuration Unit' pane and enter "CCU" or "CDU" depending on which kind of configuration unit the tool shall be executable
or
press **Ctrl+l** to pop up the 'Configuration Unit Codes' window. Then select the Configuration Unit Code and press the **Ok** button.

The definition of parameters is *optional*. If parameters will be defined, they will be **added to** the end of the standard parameters (version identification). When inserting parameter definitions, all fields are *mandatory* and have to be filled in.

The parameters have to be defined according the following steps.

2. Select the ***Name*** field and enter the name of this parameter.

The name will support you to remember to which the following definitions relate to. It will not transferred from I_MDB to the tool interface.

When you have selected the name field, the name of the related configuration unit will appear in the field just above the name field. At the same time the ***Data Type*** field will be filled with the value STRING.

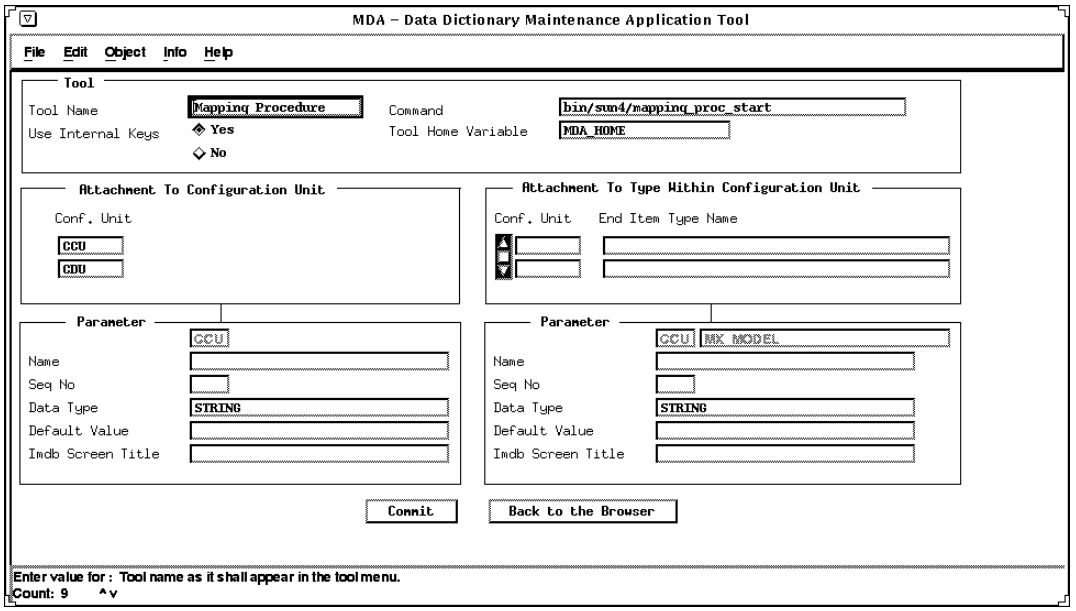

Figure 34. Attachment of a Mapping Procedure to a CCU and a CDU Version

3. Select the ***Seq No*** field and enter the position where the value of this parameter should appear at execution time.

The sequence number defines the ordering of the parameters. They are append at the end of the standard parameters. The numbers should be without gaps starting at 1.

4. Select the ***Data Type*** field to define the data type of the parameter. The default data type has been set to STRING.
5. If the data type shall be changed, press **Ctrl+l** to pop up the 'Tool Parameter Types' selection window.
   Then select one of the shown data types and press the **Ok** button.
6. Select the ***Default Value*** field and enter the default value for this parameter.

This default value will appear in I_MDB when selecting a tool from the menu. This default value may be changed in I_MDB.

7. Select the ***Imdb Screen Title*** field and enter the screen title of the parameter which will be shown in the Tool Invocation window of I_MDB.
8. Press the **arrow down key** if more than one parameter shall be transmitted. The cursor must be located in the parameter area. Then continue according the steps NO TAG to NO TAG

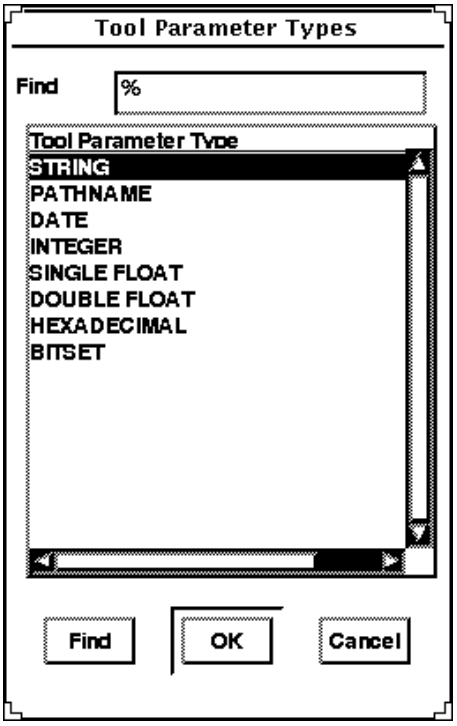

Figure 35. Parameter Data Type Selection window

To attach the User Tool to an End–Item–Type within a Configuration Unit

The user tool may be attached to End Item Types. For each End Item Type, the navigation scope has to be defined in which the tool shall be available. The configuration unit identifies the navigation scope (CCU and / or CDU) in I_MDB.

On the middle and lower right part of the editor window, the parameters necessary to attach the user application tool to an End Item Type within a CCU version and/or to a CDU version, will be specified.

1. Move the cursor to the first ***Conf.Unit*** field within the 'Attachment to Type Within Configuration Unit' pane and enter "CCU" or "CDU" depending in which navigation scope the tool shall be executable
   or
   press **Ctrl+l** to pop up the 'Configuration Unit Codes' window. Then select the Configuration Unit Code and the OK button.

2. Select the ***Type Name*** field and enter the name of the End Item Type for which the tool shall be executable
   or
   press **Ctrl+l** to pop up the 'End Item Type Selection' window. Then select the End Item Type and the press the **Ok** button.

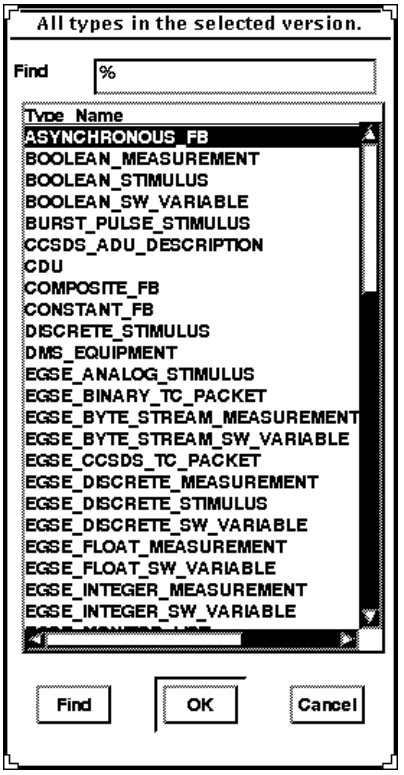

Figure 36. End Item Type Selection window

The parameter set is *optional*. If parameters will be defined, they will be added to the standard parameters (version identification). When using the parameter set, all fields are *mandatory* and have to be filled.

3. Perform the parameter definition according to the previous section step 2 to step 6.

4. If you want to define more attachments, move the cursor to the ***Conf.Unit*** field and press the arrow down key until you get an empty field.
Then continue and enter the End Item Type Name to be attached and enter the related parameters if needed.

To make the definitions persistent

1. Press **Commit** to make the changes persistent within the database.

When you have finished the definitions of the tool invocation in DADI–MA, the MDB version has to be exported and the data dictionary has to be loaded into the MDB.

5.10 Composite Aggregate Definition

The Data Dictionary consists of a set of aggregates. Each aggregate is assigned to *one* or *more* types. So far the aggregates belonging to a type are independent from each other, except that they all contain records of the same type.

A so called composite aggregate can be constructed by grouping aggregates. A composite aggregate is an aggregate on a higher level consisting of simple aggregates. All aggregates of a composite aggregate are displayed in the same window in the user defined sequence.

Composite aggregates are defined independently from end item types. A composite aggregate can be assigned to *one* or *more* end item types.

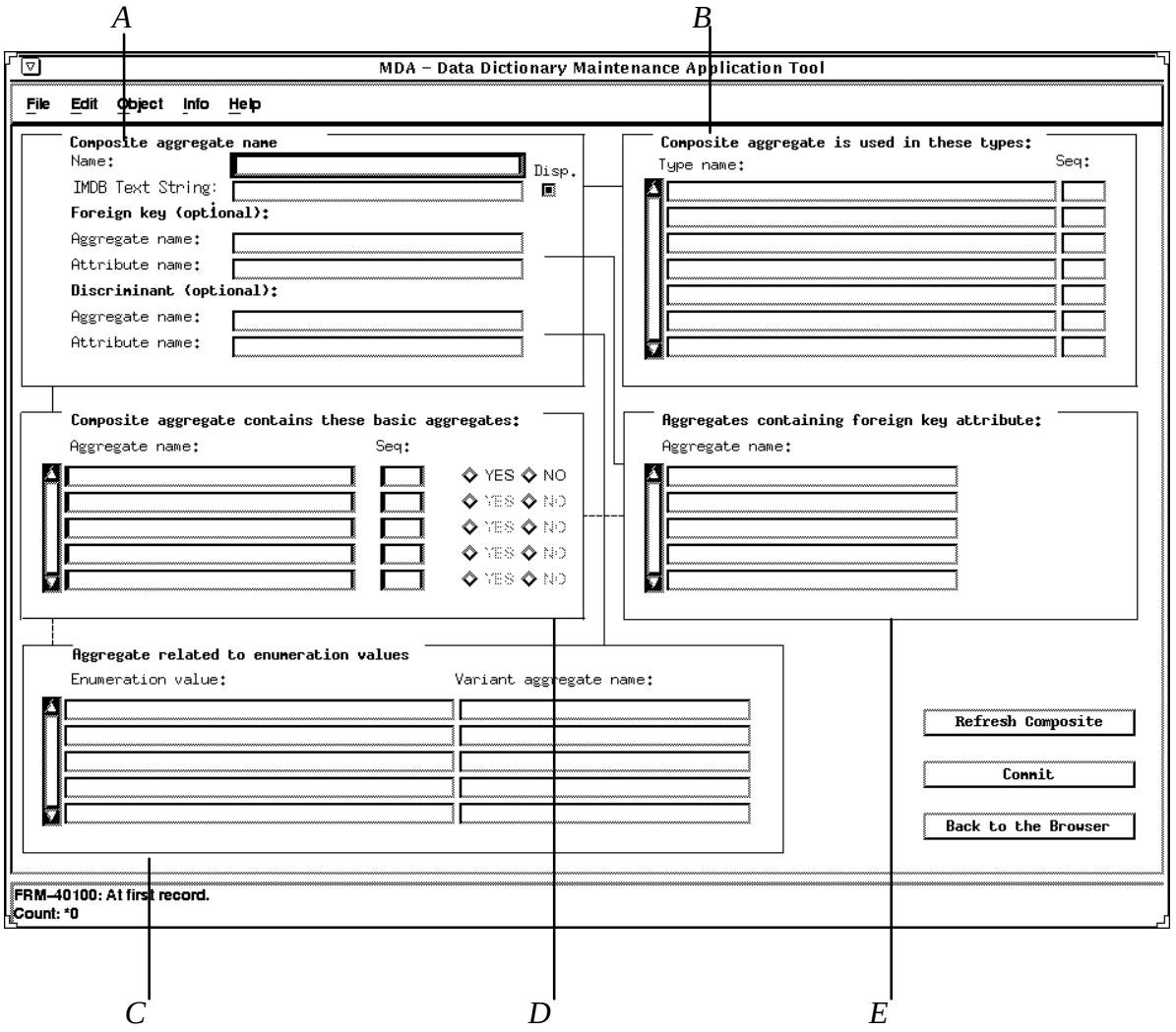

Figure 37. Composite Aggregate Definition window

A Composite Aggregate

Within this frame the composite aggregate will be defined by its name and the text string which shall be shown in I_MDB. A variant can be defined for one or more aggregates of the composite aggregate. Also foreign key references can be defined for one or more aggregates of the composite aggregate.

B *Composite Aggregate Usage*

Within this frame the composite aggregate, defined in frame A, can be assigned to one or more End–Item–Types.

C *Aggregate related to Enumeration Values*

Within this frame the aggregates of variant are defined including the enumeration value.

D *Aggregates contained in the Composite Aggregate*

Within this frame all 'simple' aggregates are defined which makes up the composite aggregate created in frame A.

E *Aggregates containing Foreign Key Attributes*

Within this frame the reference from the foreign key aggregate to other aggregates are defined.

To create Composite Aggregates

1. Select **Edit–>Composite Aggregate Def.** from the main menu bar to enter the aggregate definition window.

Within the upper left frame, the composite aggregate will be defined. The composite aggregate will be constructed by grouping aggregates. The relationships and dependencies are also created here.

1. Select the composite aggregate ***Name*** field.
2. Select **Object–>Insert** from the main menu bar to get an empty composite aggregate record.
3. Enter the composite aggregate name.
4. Select the ***IMDB Text String*** field and enter the string which shall be shown for the composite aggregate in I_MDB at run–time.
5. The ***Display*** flag is for the specification of the composite aggregate visibility in I_MDB. By default this flag is selected, which results to a displayed composite aggregate.

The next steps describe the basic aggregate definition which makes up the composite aggregate. A composite aggregate can contain one or more basic aggregates, but no composite aggregate. A composite aggregate is also unique.

6. Select within the ***Aggregate name*** scroll list the first empty field and press **Ctrl+l** to pop up the 'Aggregate List' window.
7. Select the aggregate which shall be included in the composite aggregate and press **Ok**.
8. Select the ***Seq.*** field and enter the sequence number for the aggregate.

The sequence number defines the ordering of the aggregates belonging to the composite aggregate within I_MDB during run–time. The sequence numbers start with one. The sequence number is mandatory.

9. The ***Separated*** flag is for the specification of separation lines in I_MDB during run–time. A structured visualization may be obtained in I_MDB. If the YES flag is set, a separation line will be displayed below the attributes of this aggregate. By default the NO flag is selected.

10. If more than one aggregate shall be defined for the composite aggregate, the steps NO TAG to NO TAG have to be repeated.

The last step is the definition which End–Item–Types shall use the composite aggregate. The composite aggregate may be assigned to one or more End–Item–Types.

11. Select the first empty field in the ***Type name*** field and and press **Ctrl+l** to pop up the 'Allowed Type Names' window.

12. Select the type which shall be related to the composite aggregate and press **Ok**.

13. Select the ***Seq.*** field and enter the sequence number for the type.

The sequence number defines the ordering of the types belonging to the composite aggregate within I_MDB during run–time. The sequence numbers start with one. The sequence number is mandatory.

14. If the composite aggregate shall be related to more than one type, the steps NO TAG to NO TAG have to be repeated.

15. Press **Commit** to make the changes persistent.

That are all things for the definition of composite aggregates, not containing variant parts or foreign key references. The definition of variant parts and foreign key references will be described next.

To create Variant Parts

For the aggregates of a composite aggregate, a variant part can be defined which is similar to the construct 'variant record' in ADA. A subset of the aggregates is defined as the variant part. *One* special attribute of an aggregate represents the discriminant and this has to be defined first. The variant part definition is optional.

16. It is prerequisite, that at least one composite aggregate has been defined according the steps NO TAG to NO TAG

17. Select the composite ***aggregate name*** field and press the **arrow down key** until the composite aggregate appears for which the variant part shall be created.

18. Select the *Diskriminant (optional)* ***Aggregate name*** field and press **Ctrl+l** to pop up the 'Aggregate List' window.

19. Select the aggregate which shall include the attribute representing the diskriminant and press **Ok**.

20. Select the *Diskriminant (optional)* ***Attribute name*** field and press **Ctrl+l** to pop up the 'Attribute List' window.
The window contains all actually defined attributes related to the 'diskriminant' aggregate. The attributes are all defined with an enumeration data type.

21. Select the attribute which shall represent the diskriminant and press **Ok**.

☞ Notice, only one diskriminant attribute and its aggregate can be defined within each composite aggregate.

The special attribute representing the discriminant is defined now. The variant aggregates have to be created next.

22. Select the first empty ***Enumeration Value*** field within the lower left frame and press **Ctrl+l** to enter the 'Enumeration Values' window.
The enumeration values of the attribute representing the discriminant are defined and listed here.

23. Select the enumeration which shall be assigned to the aggregate of variant and press **Ok**.

24. Select the enumeration value related ***Variant Aggregate name*** field and press **Ctrl+l** to pop up the 'Aggregate Names' window.

☞ Notice, the aggregate names list only contains aggregates following the aggregate with the diskriminant. The aggregates with the discriminant shall only be followed by aggregates of the variant. During the aggregate definition the complete variant shall appear at the end of the sequence of the aggregates of the composite aggregate.

25. Select the aggregate which shall be the variant aggregate and press **Ok**.

26. If additional variant aggregates shall be defined, the steps NO TAG to NO TAG have to be repeated.

27. Press **Commit** to make the changes persistent.

28. The **Refresh Composite** key may be pressed at any time to update all composite aggregate definition fields.

That are all things for the definition of variant parts. If the aggregate, containing the disriminant attribute, shall be changed, the aggregate has to be deleted by Ctrl+Del and selected new with Ctrl+l.

If foreign key references shall be defined additionally, the following steps have to be executed.

To create Foreign Key References

For the aggregates of a composite aggregate, foreign key references can be defined. *One* special attribute of *one* aggregate is the foreign key to one or more other aggregates, and this has to be defined first. The foreign key reference definition is optional.

29. It is prerequisite, that at least one composite aggregate has been defined according the steps NO TAG to NO TAG

30. Select the composite ***aggregate name*** field and press the **arrow down key** until the composite aggregate appears for which the variant part shall be created.

31. Select the *Foreign Key (optional)* ***Aggregate name*** field and press **Ctrl+l** to pop up the 'Aggregate List' window.

32. Select the aggregate which shall include the attribute representing the foreign key and press **Ok**.

33. Select the *Foreign Key (optional)* ***Attribute name*** field and press **Ctrl+l** to pop up the 'Attribute List' window.
The window contains all actually defined attributes related to the 'foreign key' aggregate.

34. Select the attribute which shall represent the foreign key and press **Ok**.

☞ Notice, only one foreign key attribute and its aggregate can be defined within each composite aggregate.

The special attribute representing the foreign key is defined now. The referenced aggregates have to be created next. The name of the foreign key attribute has to be the same in all referenced aggregates.

35. Select the first empty ***Aggregate Name*** field within the lower right frame and press **Ctrl+l** to enter the 'Aggregate Name' window.
The list contains all aggregates having included the foreign key attribute.

36. Select the aggregate which shall be references by the foreign key attribute and press **Ok**.

37. If additional referenced aggregates shall be defined, the steps NO TAG to NO TAG have to be repeated.

38. Press **Commit** to make the changes persistent.

39. Press **Back to the Browser** to return to the browser.

If the aggregate, containing the foreign key reference attribute, shall be changed, the aggregate has to be deleted by Ctrl+Del and selected new with Ctrl+l.

5.11 List Composite Aggregates by Type

DADI–MA provides the capability to display all types using composite aggregates. This is for user support during the data structure development. No definitions are executed from this window.

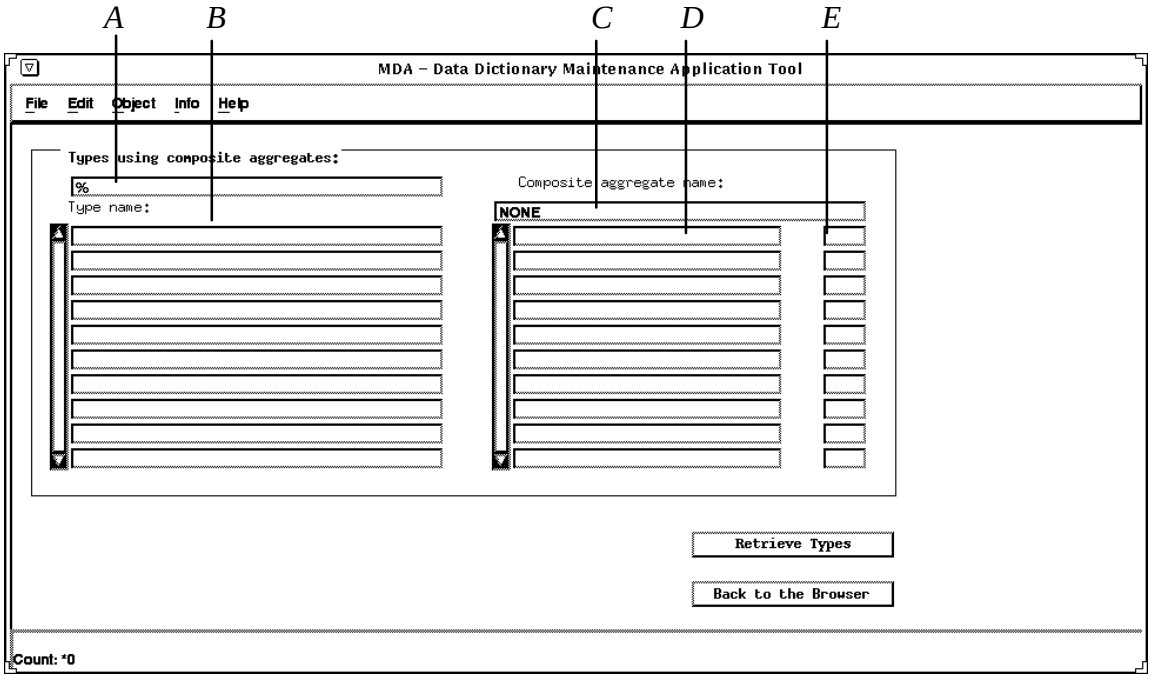

Figure 38. List Composite Aggregates by Type

A *Filter*

Specification of a filter for the type name list.

B *Type Name List*

Listing of all types using one or more composite aggregates.

C *Type Selection*

The field display the type actually selected in list B.

D *Composite Aggregate List*

Listing of all defined composite aggregates of the actually selected type.

E *Sequence Number*

The sequence number shows the ordering of the composite aggregate belonging to the type.

To list Composite Aggregates by Type

1. Select **Info–>Composite Aggregates by type** from the main menu bar to enter the 'list composite aggregates by type' window.
2. Press **Retrieve Types** to retrieve all types using composite aggregates. The type name list and the composite aggregate name list are updating.
3. Select the **Type name**, information about composite aggregates shall be supplied. The composite aggregate list will be updated accordingly.

If only a type name subset shall be shown, the next two steps have to be performed.

4. Select the **filter** field and specify the filter for retrieval, e.g. E% to display all types starting with an E.
5. Press **Retrieve Types** to retrieve all types using composite aggregates where the filter is valid. The type name list and the composite aggregate name list are updating.
6. Press **Back to the Browser** to return to the browser.

5.12 Category Report Definition

A category report provides a subset of an all detail report in a tabular form. A CCU version and a CDU version category report will be offered. The possible categories will be predefined with DADI–MA. The definition of a category report includes an ordered list of aggregate / attribute pairs to be printed in the report. A category report is invoked via I_MDB.

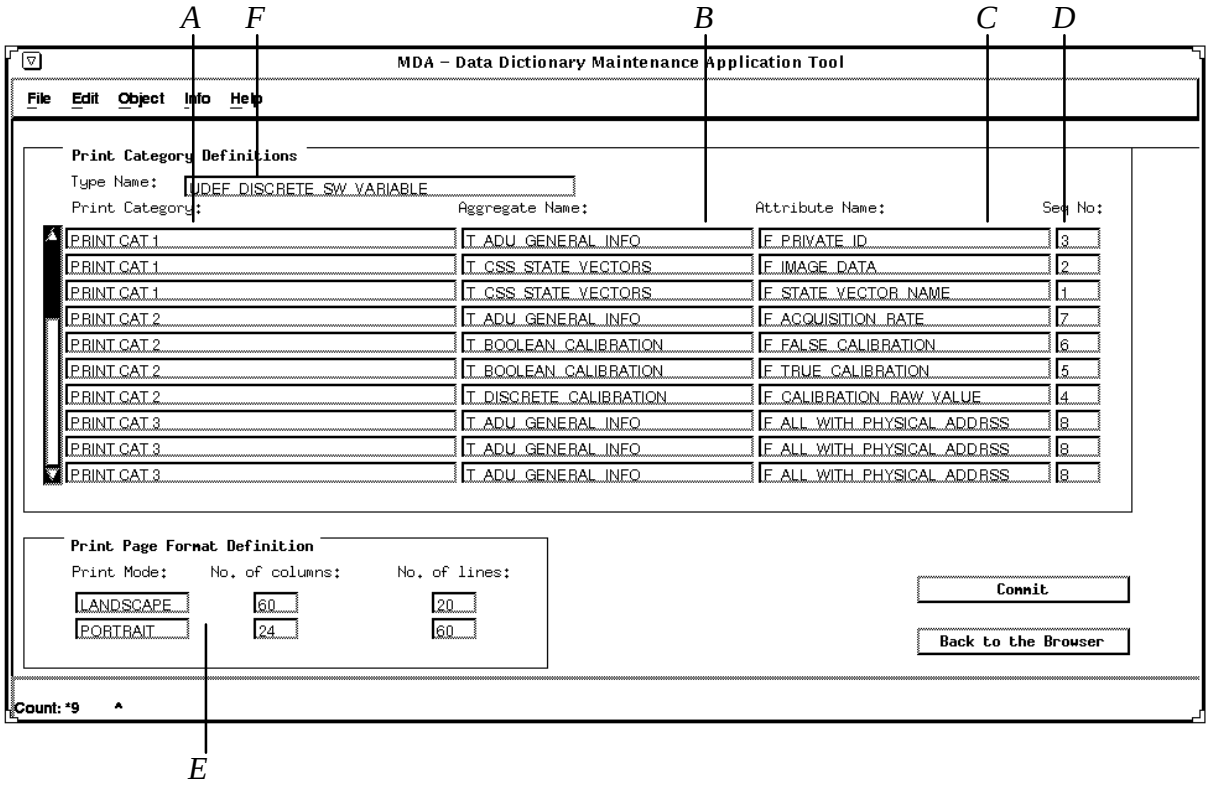

Figure 39. Print Category Report Definition

A *Print Category*

Definition of the print categories, e.g. calibration data, monitoring data, pin assignment,etc.

B *Aggregate Name*

Specification of the aggregates to be printed.

C *Attribute Name*

Specification of the attributes to be printed.

D *Seq. No.*

Definition of the attribute print ordering.

E *Print Page Format Definition*

Definition of the print mode, and the number of lines and columns to be printed.

F *Type Name*

Definition of the scope the selection list (B) aggregate, attribute pairs is working on

To define a Category Report

1. Select **Edit–>Category Report Definition** from the main menu bar to enter the 'Print Category Definitions' window.
2. Select the ***Type Name*** field and press **Ctrl+l** to pop up the 'Type Name selection list'. Select the appropriate type. The scope of the 'Select Aggregate and Attribute Name' will be limited to the selected End Item Type.
3. Select the first empty ***Print Category*** field and enter the print category name.
   The print category name is a string and can be defined on the user choice. Each print category may only have one multi record aggregate.
4. Select the ***Aggregate Name*** field and press **Ctrl+l** to pop up the 'Select Aggregate Name and Attribute Name' window.

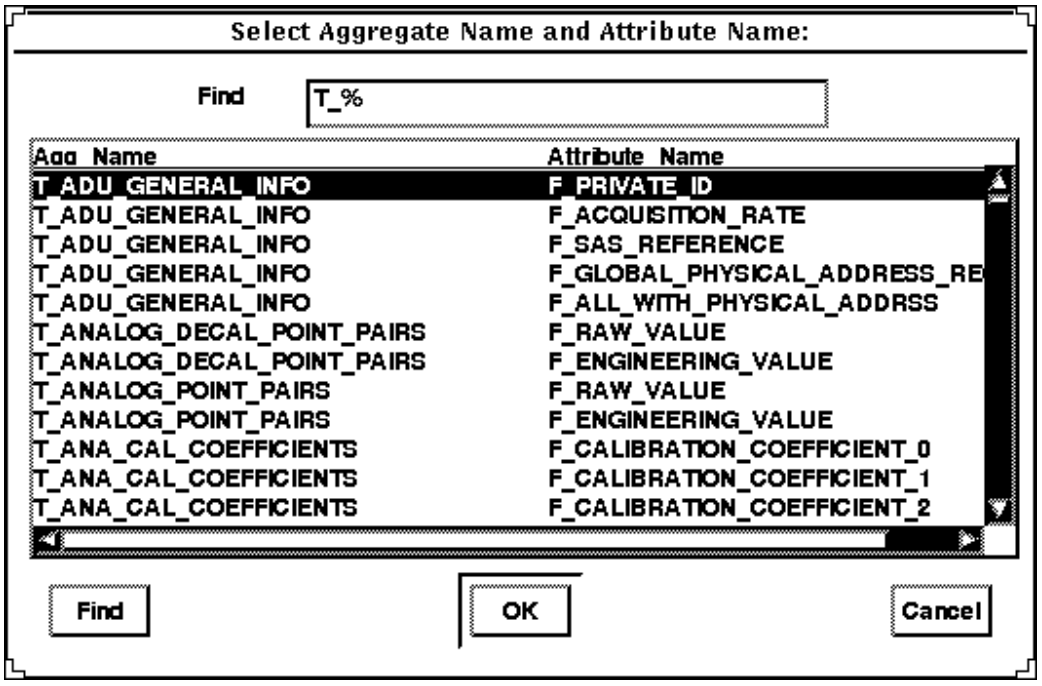

Figure 40. Aggregate and Attribute Selection for Category Report

The selection list contains all defined aggregates and attributes within the database version. On the left side there are the aggregates and their related attributes are displayed accordingly on the right side.

5. Select the ***Aggregate – Attribute*** which shall be included for the previously defined category.

Use the scroll bars for the selection if necessary. The find filter may be used in conjunction with the find key to display only a subsection of all available aggregates and views.

6. Press **OK** to return to the Category Definition window where the selection now appears.
7. Select the ***Seq. No.*** field and enter a sequence number.

The sequence number defines the attribute print ordering for the report. The attributes of an End–Item will be printed in a row where the position in that row is depending on the sequence number. Each print category has its own sequence numbers. The sequence number is an integer greater than zero and it must be unique within a print category. The field is mandatory.

8. The steps NO TAG to NO TAG have to be repeated if *additional* Aggregates–Attributes shall be defined for that print category, or if a *new* print category shall be defined. If additional Aggregates–Attributes shall be specified for an existing category, the print category name has to be entered once more.

The lower part of the Print Category Definition window contains the definiton of the print page format. Landscape and portrait prints can be defined. The default print mode is landscape.

9. Select the ***No. of columns*** field and enter the number of character columns to be printed.

10. Select the ***No. of lines*** field and enter the number of character lines to be printed.

If a portrait print mode shall be defined in addition to the landscape print mode, perform the steps NO TAG to NO TAG

11. Select the empty ***print mode*** field and enter **PORTRAIT**.

12. Select the ***No. of columns*** field and enter the number of columns to be printed.

13. Select the ***No. of lines*** field and enter the number of lines to be printed.

A print mode can be deleted by the Ctrl+Del key.

14. Press **Commit** to make the changes persistent.

15. Press **Back to the Browser** to return to the main menu.

When starting the report generator in I_MDB, a tabular report of all End–Item detail data which belong to the specified category and CCU or CDU version, will be prepared.

5.13 Data Type Definitions

This chapter describes in the first section for the MDA basis data types the data type definitions, their ranges and the allowed characters. The second section describes the types, the allowed inputs and ranges for the Data Dictionary entries.

5.13.1 MDA Basis Data Types

5.13.1.1 SINGLE_FLOAT

- 12 bytes long (including all characters)
  min = –3.40282E+38
  max = 3.40282E+38
- Number of digits allowed for the mantissa < 7
  One decimal point and one sign (optional)
  At least one digit before and one digit after the decimal point
- Characters allowed in the mantissa ’0’ .. ’9’, ’+’, ’–’, ’.’
- Number of digits allowed for the exponent <
  Leading ’e’ or ’E’ and one sign (optional)
- Characters allowed in the exponent ’0’ .. ’9’, ’+’, ’–’, ’e’, ’E’
- No enclosed blanks allowed

5.13.1.2 DOUBLE_FLOAT

- 22 bytes long (including all characters)
  min = –1.79769313486231E+308
  max = 1.79769313486231E+308
- Number of digits allowed for the mantissa < 16
  One decimal point and one sign (optional)
  At least one digit before and one digit after the decimal point
- Characters allowed in the mantissa ’0’ .. ’9’, ’+’, ’–’, ’.’
- Number of digits allowed for the exponent < 4
  Leading ’e’ or ’E’ and one sign (optional)
- Characters allowed in the exponent ’0’ .. ’9’, ’+’, ’–’, ’e’, ’E’
- No enclosed blanks allowed

5.13.1.3 INTEGER

- 11 bytes long (10 digits plus sign)
  max integer = 2**31 –1
  min integer = – (2**31)
- Characters allowed ’0’ .. ’9’, ’+’, ’–’

· No enclosed blanks allowed

5.13.1.4 BITSET

· 32 bytes long
· Characters allowed ’0’, ’1’
· No leading blanks (if entered they will be cut by I_MDB)
· No enclosed blanks
· No trailing blanks (if entered they will be cut by I_MDB)

5.13.1.5 HEXADECIMAL

· 255 bytes long
· Characters allowed ’0’ .. ’9’, ’A’ .. ’F’
· No leading blanks (if entered they will be cut by I_MDB)
· No enclosed blanks
· No trailing blanks (if entered they will be cut by I_MDB)

5.13.1.6 PATHNAME

· Syntax of an *MDB pathname and node name :*

pathname ::= \node_name {\node_name } | \

· node_name ::= letter_or_digit_or_underline { letter_or_digit_or_underline }
with length of node_name <= 16
· letter_or_digit_or_underline ::= letter | digit | underline
· In the MDB the node names are stored in capital letter.

5.13.1.7 ENUMERATION

· name : string 1..40 characters long
Characters allowed ’0’ .. ’9’, ’A’ .. ’F’
No leading blanks
No enclosed blanks
· value : string 1..40 characters long

5.13.1.8 STRING

· 1..MAX_STRING long, with MAX_STRING = 255

5.13.1.9 LONG_CHAR

- is used for MDA_VERY_LONG_RAW Aggregate to define a text
- ASCII data with unlimited length
- there will be no input window in I_MDB

5.13.1.10 LONG_RAW

- is used for MDA_VERY_LONG_RAW Aggregate to define an image
- binary data with unlimited length
- there will be no input window in I_MDB

5.13.1.11 DATE

- DD–MOM–YY HH24:MI:SS

5.13.2 Data Dictionary Entries

5.13.2.1 Version Name

- The Version Name is a string with a length between 1 and 16
- Lower case letters are not allowed
- Characters allowed ’A’..’Z’, ’_’
- No leading blanks
- No enclosed blanks

5.13.2.2 Domain Name

- The Domain Name is a string with a length between 1 and 16
- Characters allowed ’A’..’Z’, ’_’
- Lower case letters are not allowed.
- No leading blanks
- No enclosed blanks

5.13.2.3 End–Item–Type Name

- The End–Item–Type Name is a string with a length between 1 and 28
- Characters allowed ’A’..’Z’, ’_’
- Lower case letters are not allowed
- No leading blanks
- No enclosed blanks

5.13.2.4 End–Item–Type Description

- The End–Item–Type Description is a string with a length between 1 and 80
- The name may have lower case and upper case letters, or a mixture

5.13.2.5 Consistency Check Procedure Name

- The Consistency Check Procedure Name is a string with a length between 1 and 30
- Characters allowed ’A’..’Z’, ’_’
- No leading blanks
- No enclosed blanks

5.13.2.6 Consistency Check Package Name

- The Consistency Check Package Name is a string with a length between 1 and 30
- Characters allowed 'A'..'Z', '_'
- No leading blanks
- No enclosed blanks

5.13.2.7 Mapping Procedure Name

- The Mapping Procedure Name is a string with a length between 1 and 30
- Characters allowed 'A'..'Z', '_'
- No leading blanks
- No enclosed blanks

5.13.2.8 Mapping Package Name

- The Mapping Package Name is a string with a length between 1 and 30
- Characters allowed 'A'..'Z', '_'
- No leading blanks
- No enclosed blanks

5.13.2.9 Aggregate Name

- The Aggregate Name is a string with a length between 1 and 28
- Characters allowed 'A'..'Z', '_'
- Lower case letters are not allowed
- No leading blanks
- No enclosed blanks

5.13.2.10 Aggregate Description

- The Aggregate Description is a string with a length between 1 and 80
- All characters are allowed

5.13.2.11 IMDB Frame Title

- The IMDB Frame Title is a string with a length between 1 and 40
- All characters are allowed

5.13.2.12 IMDB Menu String

- The IMDB Menu String is a string with a length between 1 and 40
- All characters are allowed.

5.13.2.13 Minimum Number of Aggregate Records

- The minimum number of aggregate records is an integer between 0 and 2**31–1

5.13.2.14 Maximum Number of Aggregate Records

- The maximum number of aggregate records is an integer between 1 and 2**31–1
- A maximum number of zero is equal to 2**31–1

5.13.2.15 Aggregate Sequence Number

- The aggregate sequence number is an integer between 1 and 999

5.13.2.16 AttributeName

- The Attribute Name is a string with a length between 1 and 30
- Characters allowed ’A’..’Z’, ’_’
- Lower case letters are not allowed
- No leading blanks
- No enclosed blanks

5.13.2.17 IMDB Screen Title

- The IMDB Screen Title is a string with a length between 1 and 40
- All characters are allowed

5.13.2.18 Attribute Sequence Number

- The attribute sequence number is an integer between 1 and 999

5.13.2.19 MDA Data Size

- The MDA data size depends on the selected data type. The minimum and maximum data size ranges of the data types are specified in chapter NO TAG.

5.13.2.20 Constraint Name

- The Constraint Name is a string with a length between 1 and 30
- Characters allowed 'A'..'Z', '_'
- Lower case letters are not allowed
- No leading blanks
- No enclosed blanks

5.13.2.21 Constraint Range Boarders

- The minimum value for a constraint range is an integer
- The maximum value for a constraint range is an integer
- The maximum value must be greater than the minumum value
- It is not allowed to enter hexadecimal or octal range values

5.13.2.22 Enumeration Name

- The Enumeration Name is a string with a length between 1 and 40
- Characters allowed 'A'..'Z', '_'
- Lower case letters are not allowed
- No leading blanks
- No enclosed blanks

5.13.2.23 Enumeration Value

- The Enumeration Value is a string with a length between 1 and 40
- All characters are allowed

5.13.2.24 Enumeration Sequence Number

- The enumeration sequence number is an integer between 1 and 999

5.13.2.25 Print Category

- The Print Category name is a string with a length between 1 and 40
- All characters are allowed

5.13.2.26 Print Category Sequence Number

- The Print Category sequence number is an integer between 1 and 999

This page is intentionally left blank.

6 INSTALLATION OF EXPORTED MDB VERSION

The Data Dictionary Tool exports the data which shall be used for an MDB installation into the following directories:

1. $MDA_HOME/util/dadi/dadi_export/dadi_data
2. $MDA_HOME/util/dadi/dadi_export/aggregates
3. $MDA_HOME/util/dadi/dadi_export/api_views
4. $MDA_HOME/util/dadi/dadi_export/bde_control_files

The installation procedure to install an MDB expects the actual exported Data Dictionary Data under the directories:

1. $MDA_HOME/config/mdb/install/dadi_export/dadi_data
2. $MDA_HOME/config/mdb/install/dadi_export/aggregates
3. $MDA_HOME/config/mdb/install/dadi_export/api_views
4. $MDA_HOME/config/mdb/install/dadi_export/bde_control_files

To make the exported Data Dictionary Tool Data visible for the MDB installation procedure the following steps have to be performed:

rm –r $MDA_HOME/config/mdb/install//dadi_export
(to remove the old data from the MDB installation directories)

cp –r $MDA_HOME/util/dadi/dadi_export $MDA_HOME/MDA/config/mdb/install
(to copy the actual exported Data Dictionary Data into the MDB installation directories)

After the export Data Dictionary Data have been copied into the MDB installation directories, the installation procedures for installation of an MDB have to be executed. This is done by

1.) $MDA_HOME/config/mdb/install/admin_scripts/install_mdb
2.) $MDA_HOME/config/mdb/install/admin_scripts/initialize_mdb.

For details refer to MDA Administration Manual, Chapter Install MDB.

This page is intentionally left blank.

A ACRONYMS

AP	Automated Procedure
API	MDA – Application Programmers Interface
APM	Attached Pressurized Module
BDE	Batch Data Entry Tool
CCU	Configuration Control Unit
CDU	Configuration Data Unit
CLS	Columbus Language System
CM	Configuration Management
CU	Configuration Unit
DADI	Data Dictionary Tool
DADI–MA	Data Dictionary Tool (–MA Maintenance Application)
DD	Data Dictionary
DMS	Data Management Subsystem
EGSE	Electrical Ground Support Equipment
EI	End Item Type
ICD	Interface Control Document
IMDB	Interactive Mission Database
MDA	Mission Database Application
MDB	Mission DataBase
SID	Short Identifier
SSMB	Space Station Manned Base
TCS	Thermal Control Subsystem
UCL	User Control Language

B DEFINITIONS

A

Access rights define what access various users or applications have to objects or entities.

Action Actions are high–level commands which provide for a flight configuration control at higher levels than elementary commands and **Automated Procedures (AP)**. They are pre–planned, goal–oriented operations of either Payload Elements or Subsystems. An Action may be linked to lower–level actions reflecting the hierarchical decomposition of the on–board operation. On Action level, all necessary pre–checks are carried out to ensure a safe implementation of an automated operation consistent with the actual mission phase and flight element configuration.

Application Program or set of programs performing some specialized user–oriented function (as opposed to general–purpose programs like a **DBMS**, or an **operating system**).

Archive Refers to the process of relegating obsolete data to external backing storage. The reverse operation (copying archived data back to active storage) is known as **restore**.

Authorized User see **User**

Automated Procedure A program written in the **User Control Language (UCL)**.

B

C

CDU domain is a set of item types

Child in a hierarchical structure, denotes an immediate descendant of a given component. A child is thus located one hierarchical level below its **parent.**

Compilation Unit Smallest unit of code that is accepted by the compiler. In UCL, there are 3 types of Compilation Units: Automated Procedure (AP), Library Specification, and Library Implementation (or Library body).

Component Component is a generic term used to cover any item in the higher levels of the software architecture (i.e. product, assembly and subsystem).

Configuration Unit (CU) Collection of MDB items treated as a single unit for configuration management purposes.
CUs are of two kinds:
(a) Configuration **Data** Units (CDU), which contain the actual data
(b) Configuration **Control** Units (CCU), which contain reference information (CU name, version number, etc.) about other CUs, just like a directory in a file system.

Configuration Control Unit (CCU)
A Configuration Control Unit is a Configuration Unit used to define and control other Configuration Units. It identifies which specific combina-

tions of CDU instances make up a particular configuration. A Configuration Control Unit may, in turn, point to lower–level control units, thus leading to a hierarchical configuration tree whose topmost (root) component corresponds to an entire Columbus Flight Configuration.

Configuration Data Unit (CDU)
Configuration Data Units are composite entities containing the actual data items (grouped into individual units for configuration management purposes).

Consistency Consistency is the software characteristic that ensures uniform design and implementation techniques and notations.

Consistency state LOCAL VALID,
LOCAL INVALID
GLOBAL VALID
NONE

D

Database A common or integrated collection of interrelated data whose purpose is to serve one or more applications.

Database Management System (DBMS)
The software responsible for the actual definition, storage and manipulation of data in a **Database** at both the physical and logical level.

Database Administrator (DBA)
The person(s) responsible for the operation and maintenance of a DBMS.

Data Entry / Data Maintenance
Generally refers to the process of entering and/or updating data in the database.
In this context, the term ”maintain” refers to any operation which alters the state of the Database, i.e. add (insert) new data, modify existing data, or delete data.

Database integrity Refers to the state in which the database is considered to be undamaged (both physically and logically).

Database Server Refers to the processor (network node) physically hosting the Database and providing DB access services to local or remote applications (clients).

DBA see **Database Administrator**

DBMS see **Database Management System**

Default a value supplied by the system when a user does not specify a required parameter, qualifier, or attribute.

Distributed Database A collection of databases that can be operated and managed separately and also share information.

Distributivity Distributivity is the degree to which software functions are geographically or logically separated within the system.

E

End–Item	see **MDB item**
Export	In the MPS context, this term refers to the process of extracting data from a DB and preparing it for inclusion (**import**) into another DB.

F

G

GLOBAL VALID	All consistency rules are fulfilled. That implies that all internal references are existing and external references do not exist.
Ground Software	All software that executes in any COLUMBUS ground computer or in the flight configuration computers during pre–launch ground operations.

H

Hierarchical Name Tree	see **Name Tree**

I

Import	In the MPS context, this term refers to the process of receiving or including data from an external (possibly remote) DB into the local DB.
Issue	see **Version**
MDB–Item instance	an occurrence of a particular MDB item in a given CU version.

J

K

L

LOCAL INVALID	Internal references are not existing or other consistency rules as defined in the MPSICD are not fulfilled.
LOCAL VALID	All internal references and all other consistency rules are correct. External references are still existing and cannot be checked.

M

MDB instance	One installation of an MDB with a specific SID range.
MDB installation node	Server where one or more MDB's are installed.
MDB Item, MDB Object	In the context of this document, these two terms are used interchangeably to denote a uniquely identifiable entity that has been defined in the Mission Database (and corresponds to a real–world HW or SW entity). An MDB Object or Item may be decomposed into lower–level items according to the

hierarchical nametree conventions, see **Nametree** below.
An **End–Item** is an MDB item located at the lowest hierarchical level (leaf or terminal node), and hence cannot be further decomposed.

Mission — The performance of a coherent set of investigation or operations in space to achieve space programme goals. A single mission may require more than one flight, and more than one mission may be accomplished on a single flight.

Mission Database (MDB) — This is the central repository for all HW / SW configuration information about Columbus Flight Elements, Payloads and associated Ground Support Equipment. Access to the MDB is controlled and managed by MPS.

N

Nametree — Hierarchical (tree) structure within the MDB which portrays the hierarchical decomposition of Columbus Flight Configurations into systems, subsystems, equipment, etc. The topmost node of the nametree (called the root node) designates the Flight Configuration, whereas terminal nodes (leaf nodes) represent the items that cannot (or need not) be further decomposed, i.e. the so–called **end–items**.
Each MDB object is thus identifiable by a **pathname** indicating the succession of nodes to be traversed to reach that particular item in the Nametree.

Node — any component of a network or tree structure.
(e.g. LAN node, nametree node)

O

Operating System (OS) — The system software that controls the computer and its parts, performing the basic tasks such as allocating memory, and allowing computer components to communicate.

P

Parent — In a hierarchical structure, denotes an immediate ancestor of a given component.

Pathname — **see Nametree**

Q

R

Reconfiguration — A procedure which changes the status of used hardware and software items

Report — In the context of this document, a report may be defined as any human–readable description of one or more MDB items. It is an assorted collection of information usually presented to the user in form of a table or itemized list (tabular format).

A report's specification contains the instructions for generating the report, e.g. data selection criteria, formatting instructions, and sort order. This specification may be stored in the MDB.
On request, a report is generated, i.e. the predefined instructions are executed, and the resulting output routed either to the workstation's screen (on–screen report), to the printer or to a user–selected file.

Revision see **Version**

S

System Administrator A person responsible for the operation and maintenance of the **operating system** of a computer.

T

U

Unit Unit is a generic term used to cover any lower level item of breakdown in the software architecture e.g. module, object etc.

User Throughout this document the term **User** refers to any person using MDA–provided services. Users are grouped into different classes or categories and will be assigned different privileges based on the task they perform.

V

Version In the course of its life cycle, a Configuration Unit (CU) usually undergoes several modifications due to evolving user requirements, design changes, etc.
It will thus possibly exist within the MDB in many different forms or instances (CU occurrences) commonly referred to as *versions*, e.g. DMS Version 3.2.1.

In the Configuration Management (CM) context, however, the various CU occurrences are classified according to the types of changes that have been made. The terms **versions**, **issues**, and **revisions** are then used to differentiate between the following 3 cases:
– Modifications due to *requirements changes* which result in a new **version**
– Modifications due to *design changes* which result in a new **issue**.
– Modifications due to *bug fixes*, *repairs* or *other corrections* (affecting neither the design nor the requirements) which result in a new **revision**.

(In the above example, the CU Identifier "DMS Version 3.2.1", therefore, refers to Version 3, Issue 2, Revision 1 of the DMS)

W

X

Y

Z