AIMMS User's Guide - Identifier Declarations

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

Chapter 5

Identifier Declarations

This chapter shows you how to add new identifier declarations using the **Model Explorer** and how to modify existing identifier declarations. The chapter also explains how any changes you make to either the name or the domain of an identifier are propagated throughout the remainder of your model.

This chapter

5.1 Adding identifier declarations

Identifiers form the heart of your model. All data are stored in identifiers, and the bodies of all functions and procedures consist of statements which compute the values of one identifier based on the data associated with other identifiers.

Identifiers

Adding an identifier declaration to your model is as simple as adding a node of the desired type to a global declaration section (or to a declaration section local to a particular procedure or function), as explained in Section 4.3. Aimms will only allow you to add identifier declarations inside declaration sections.

Adding identifiers

There are many different types of identifiers. Each identifier type corresponds to a leaf node in the model tree and has its own icon, consisting of a white box containing one or more letters representing the identifier type. When you add an identifier to a declaration section of your model in the model tree, you must first select its identifier type from the dialog box as presented in Figure 5.1.

Identifier types

After you have selected the identifier type, Aimms adds a node of the specified type to the model tree. Initially, the node name is left empty, and you have to enter a unique identifier name. If you enter a name that is an Aimms keyword, an identifier predeclared by Aimms itself, or an existing identifier in your model, Aimms will warn you of this fact. By pressing the **Esc** key while you are entering the identifier name, the newly created node is removed from the tree.

Identifier name
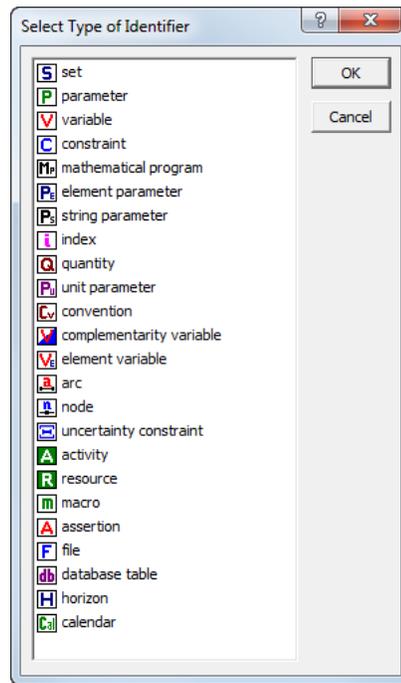
Figure 5.1: Choosing an identifier type

There is no strict limit to the length of an identifier name. Therefore, you are advised to use clear and meaningful names, and not to worry about either word length or the intermingling of small and capital letters. AIMMS offers special features for name completion such as **Ctrl-Spacebar** (see Section 5.2), which allow you to write subsequent statements without having to retype the complete identifier names. Name completion in AIMMS is also case consistent.

Meaningful names are preferable

In addition, when an identifier is multidimensional, you can immediately add the index domain to the identifier name as a parenthesized list of indices that have already been declared in the model tree. Alternatively, you can provide the index domain as a separate attribute of the identifier in its attribute form. Figure 5.2 illustrates the two ways in which you can enter the index domain of an identifier. In both cases the resulting list of indices will appear in the model tree as well as in the IndexDomain attribute of the attribute form of that identifier. In the IndexDomain attribute it is possible, however, to provide a further restriction to the domain of definition of the identifier by providing one or more domain conditions (as explained in full detail in the Language Reference). Such conditions will not appear in the model tree.

Index domain
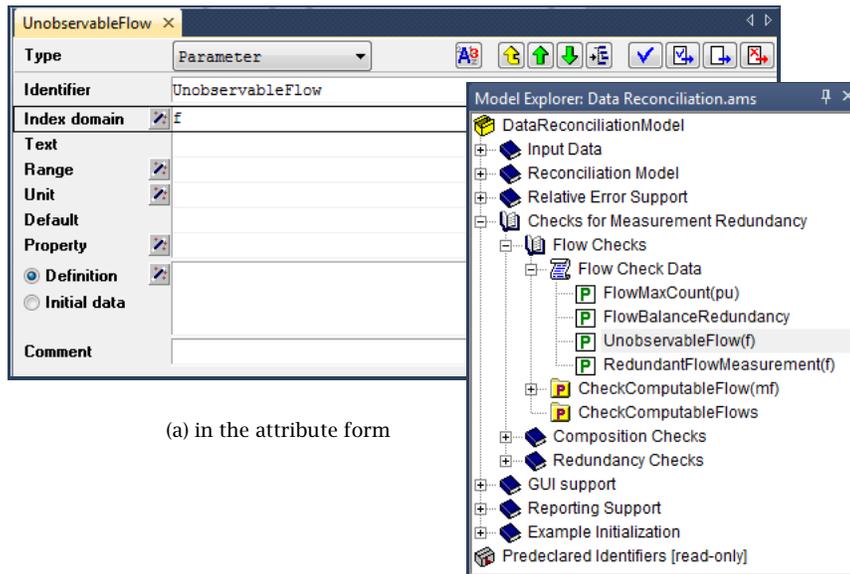
(a) in the attribute form

(b) in the model explorer

Figure 5.2: Specifying an index domain

The identifier declarations in the model tree can be used independently of the order in which they have been declared. This allows you to use an identifier anywhere in the tree. This order independence makes it possible to store identifiers where you think they should be stored logically. This is different to most other systems where the order of identifier declarations is dictated by the order in which they are used inside the model description.

Unrestricted order of declarations

In general, all identifiers in an AIMMS model are known globally, unless they have been declared inside a local declaration section of a procedure or function. Such identifiers are only known inside the procedure or function in which they have been declared. When you declare a local identifier with the same name as a global identifier, references to such identifiers in the procedure or function will evaluate using the local rather than the global identifier.

Identifier scope

Local identifiers declared in procedures and functions are restricted to particular types of identifier. For example, AIMMS does not allow you to declare constraints as local identifiers in a procedure or function, as these identifier types are always global. Therefore, when you try to add declarations to a declaration section somewhere in the model tree, AIMMS only lists those types of nodes that can be inserted at that position in the model tree.

Local declarations

As an alternative to explicitly adding identifier nodes to the model tree, it is sometimes possible that Aimms will implicitly define one or more identifiers on the basis of attribute values of other identifiers. The most notable examples are indices and (scalar) element parameters, which are most naturally declared along with the declaration of an index set. These identifiers can, therefore, be specified implicitly via the Index and Parameter attributes in the attribute form of a set. Implicitly declared identifiers do not appear as separate nodes in the model tree.

Declarations via attributes

5.2 Identifier attributes

The attributes of identifier declarations specify various aspects of the identifier which are relevant during the further execution of the model. Examples are the index domain over which the identifier is declared, its range, or a definition which expresses how the identifier can be uniquely computed from other identifiers. For the precise interpretation of particular attributes of each identifier type, you are referred to the Aimms Language Reference, which discusses all identifier types in detail.

Identifier attributes

The attributes of an identifier are presented in a standard form. This form lists all the relevant attributes together with the current values of these attributes. The attribute values are always presented in a textual representation, consisting of either a single line or multiple lines depending on the attribute. Figure 5.3 illustrates the attribute form of a variable ComponentFlow(f,c). The
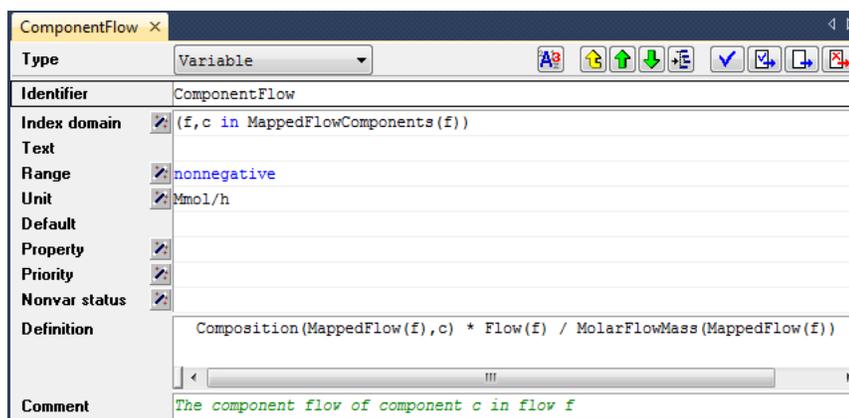
Attribute window

Figure 5.3: Identifier attributes

attributes specify, for instance, that the variable is measured in Mmol/h, and provide a definition in terms of other parameters and variables.

You do not need to enter values for all the attributes in an attribute window. In fact, most of the attributes are optional, or have a default value (which is not shown). You only have to enter an attribute value when you want to alter the behavior of the identifier, or when you want to provide a value that is different to the default.

Default values

You can freely edit the text of almost every attribute field, using the mechanisms common to any text editor. Of course, you will then need to know the syntax for each attribute. The precise syntax required for each attribute is described in the Aimms Language Reference book.

Entering attribute text ...

To help you when filling in attributes, Aimms offers specialized wizards for most of them. These wizards consists of (a sequence of) dialog boxes, which help you make specific choices, or pick identifier names relevant for specifying the attribute. An example of an attribute wizard is shown is Figure 5.4. In this

... or using attribute wizards
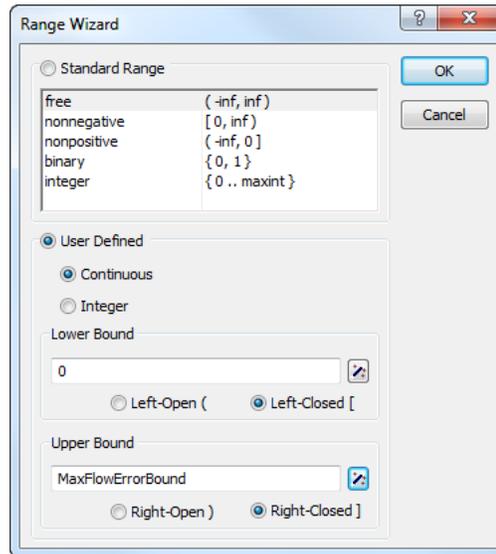
Figure 5.4: Example of an attribute wizard

wizard, the numerical range of a particular parameter or variable is specified as the user-defined interval [0,MaxFlowErrorBound]. After completing the dialog box, the result of filling in the wizard is copied to the attribute window with the correct syntax.

Some of the attribute fields are not editable by hand, but require you to always use the associated wizard. Aimms requires the use of wizards, whenever this is necessary to keep the model in a consistent state. Examples are (non-empty) Index and Parameter attributes of sets, as well ass the BaseUnit attribute of quantities.

Mandatory use of wizards

Even when you decide to enter an attribute into a field manually, Aimms still offers support to help you enter such a field quickly and easily. If your application contains a large number of identifiers and/or if the names of these identifiers are long, then it may be difficult to remember all the exact names. There are two ways to let Aimms help you in filling in the appropriate names in an attribute field:

Identifier reference support

- you can drag and drop the names from the model tree into the field, or
- with the name completion feature you can let Aimms fill in the remainder of the name based on only the first few characters typed.

When filling in an attribute field, you can drag any identifier node in the model tree to a particular location in the attribute field. As a result, Aimms will copy the identifier name, with its index domain, at the location where you dropped the identifier.

Dragging identifiers

When you use the **Ctrl-Spacebar** combination anywhere in an attribute field, Aimms will complete any incomplete identifier name at the current cursor position wherever possible. With the **Ctrl-Shift-Spacebar** combination Aimms will also complete keywords and predefined procedure and function names. When there are more than one possibilities, a menu of choices is presented as in Figure 5.5. In this menu the first possible extension will be selected and the

Name completion . . .
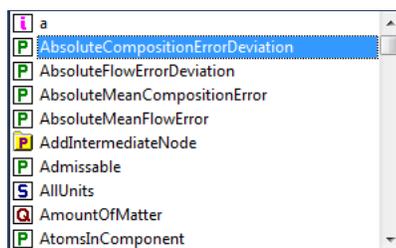
Figure 5.5: Name completion

selection will be updated as you type. When an identifier name is complete, applying name completion will cause Aimms to extend the identifier by its index domain as specified in its declaration.

By pressing **Ctrl-Spacebar** in a string that contains the :: or . characters, Aimms will restrict the list of possible choices as follows.

. . . applied to the :: and . characters

- If the name in front of the :: character is a module or library module prefix, Aimms will show all the identifiers contained in the module, or all identifiers contained in the interface of the library module, respectively.
- If the string to complete refers to a property in a PROPERTY statement, and the name in front of the . character is an identifier, Aimms will show all properties available for the identifier (based on its type).

- If the string to complete refers to an option in an OPTION statement, and the string in front of the . character refers to an element of the set All-Solvers, AIMMS will show all options available for that solver.
- In all other cases, if the name in front of the . character is an identifier, AIMMS will show all the suffices available for the identifier (based on its declaration).

5.2.1 Navigation features

From within an attribute window, there are several menus and buttons available to quickly access related information, such as the position in the model tree, identifier attributes and data, and context help on identifier types, attributes and keywords.

Navigation features

From within an attribute window you can navigate further through the model tree by using the navigation buttons displayed at the top of the window.

Browsing the model tree

- The **Parent** 🔼, **Previous** 🔼 and **Next Attribute Window** 🔽 buttons will close the current attribute window, and open the attribute window of the parent, previous or next node in the model, respectively.
- The **Location in Model Tree** 🔳 button will display the model tree and highlight the position of the node associated with the current attribute window.

When an identifier attribute contains a reference to a particular identifier in your model, you may want to review (or maybe even modify) the attributes or current data of that identifier. AIMMS provides various ways to help you find such identifier details:

Viewing identifier details

- by clicking on a particular identifier reference in an identifier attribute, you can open its attributes window through the **Attributes** item in the right-mouse pop-up menu,
- you can locate the identifier declaration in the model tree through the **Location in Model Tree** item in the right-mouse pop-up menu, and
- you can view (or modify) the identifier's data through the **Data** item in the right-mouse pop-up menu (see Section 5.4).

Through either the **Context Help** button 🔳 on the toolbar, or the **Help on** item in the right-mouse pop-up menu, you can get online help for the identifier type, its attributes and keywords used in the attribute fields. It will open the section in one of the AIMMS books or help files, which provides further explanation about the topic for which you requested help.

Context help sensitive help

5.3 Committing attribute changes

The modifications that you make to the attributes of a declaration are initially only stored locally within the form. Once you take further action, the changes in your model will be checked syntactically and committed to the model. There are three ways to do this.

Syntax checking

- **Check and commit** ✔. This command checks the current values of the attributes for syntax errors, and if there are no errors the new values are applied to the model.
- **Check, commit and close** . Same as check and commit, but if there are no errors it also closes the current form. Since this is the most frequently used action, you can also invoke it by pressing **Ctrl-Enter**.
- **Commit and close** . This command does *not* check the current values, but simply applies them to the model and then closes the form. The changes will be checked later, when the entire model is checked or when you re-open and check the form yourself.
- **Discard** . If you do not want to keep any of the changes you made in the attribute form, you can discard them using the **Discard** button.

In addition to committing the changes in a single attribute form manually as above, the changes that you have made in any attribute form are also committed when you save the model (through the **File-Save** menu), or recompile it in its entirety (through the **Run-Compile All** menu).

Saving the model

It is quite common to rename an existing identifier in a modeling application because you consider that a new name would better express its intention. In such cases, you should be aware of the possible consequences for your application. The following questions are relevant.

Renaming identifiers

- Are there references to the (old) identifier name in other parts of the model?
- Are there case files that contain data with respect to the (old) identifier name?
- Are there pages in the end-user interface that display data with respect to the (old) identifier name?

If the answer to any of these questions is yes, then changing the identifier name could create problems.

AIMMS helps you in dealing with the possible consequences of name changes by offering the following support:

Automatic name changes

- AIMMS updates all references to the identifier throughout the model text, and in addition,

■ AIMMS keeps a log of the name change (see also Section 2.5), so that when AIMMS encounters any reference to the old name in either a page or in a case file, the new name will be substituted.

Problems arise when you want to change the index domain of an identifier, or remove an identifier, while it is still referenced somewhere in your application. Such changes are called *structural*, and are likely to cause errors in pages and cases. In general, these errors cannot be recovered automatically. To help you locate possible problem areas, AIMMS will mark all pages and cases that contain references to changed or deleted identifiers. To check how a change really affects these pages and cases, you should open them, make any required adaptations to deal with the errors, and resave them.

Beware of structural changes

You can modify the type of a particular identifier in the model tree via the identifier type drop-down list `Element Parameter ▼` in the attribute window of the identifier. The drop-down list lets you select from all identifier types that are compatible with the current identifier type. Alternatively, you can change the identifier type via the **Edit-Change Type** menu.

Modifying identifier type

Before a change of identifier type is actually committed, AIMMS displays the dialog box illustrated in Figure 5.6, which lists all the attributes of the identifier that are not compatible with the newly selected type. If you do not want
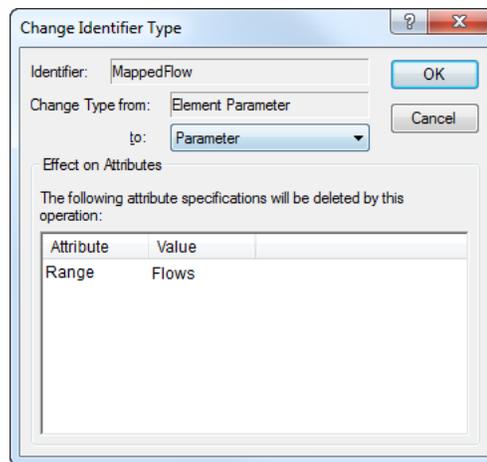
Incompatible attributes

Figure 5.6: The **Change Identifier Type** dialog box

such attributes to be deleted, you should cancel the operation at this point. When you allow AIMMS to actually perform the type change, the incompatible attributes will be deleted.

5.4 Viewing and modifying identifier data

When you are developing your model (or are reviewing certain aspects of it later on), Aimms offers facilities to directly view (and modify) the data associated with a particular identifier. This feature is very convenient when you want to enter data for an identifier during the development of your model, or when you are debugging your model (see also Section 8.1) and want to look at the results of executing a particular procedure or evaluating a particular identifier definition.

Viewing identifier data

Via the **Data** button available in the attribute window of every global identifier (see, for instance, Figure 5.3), Aimms will pop up one of the *data pages* as illustrated in Figure 5.7. Data pages provide a view of the *current contents* of
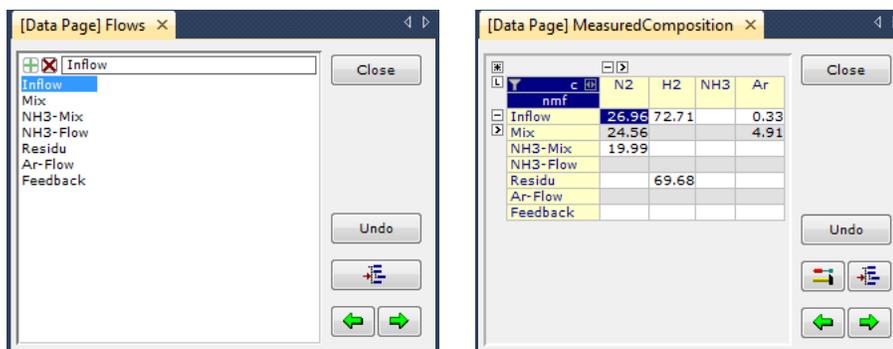
The Data button

Figure 5.7: Data pages of a set and a 2-dimensional parameter

the selected identifier. Which type of data page is shown by Aimms depends on the type of the identifier. The data page on the left is particular to one-dimensional root sets, while the data page on the right is appropriate for a two-dimensional parameter.

For variables (and similarly for constraints), Aimms will display a pivot table containing all the indices from the index domain of the variable plus one additional dimension containing all the suffixes of the variable that contain relevant information regarding the solution of the variable. Depending on the properties set for the variable, this dimension may contain a varying number of suffixes containing sensitivity data related to the variable.

Data pages for variables and constraints

Data pages can also be opened directly for a selected identifier node in the model tree using either the **Edit-Data** menu, or the **Data** command in the right-mouse pop-up menu. Additionally, you can open a data page for any identifier referenced in an attribute window by selecting the identifier in the text, and applying the **Data** command from the right-mouse pop-up menu.

Viewing data in the Model Explorer

For multidimensional identifiers, Aɪᴍᴍs displays data using a default view   *Multidimen-*
which depends on the identifier dimension. Using the [icon] button on the data   *sional identifiers*
page you can modify this default view. As a result, Aɪᴍᴍs will display the
dialog box illustrated in Figure 5.8. In this dialog box, you can select whether
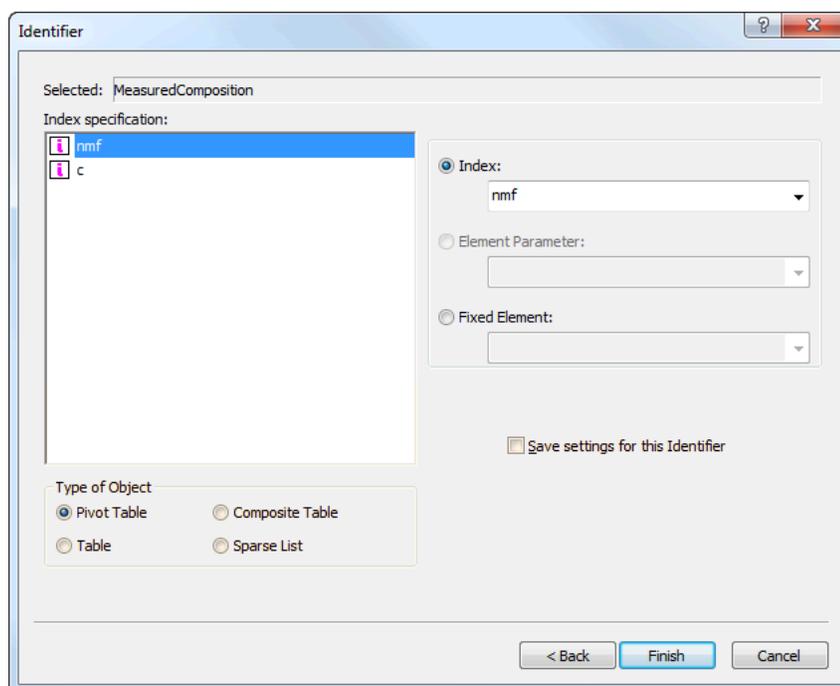

Figure 5.8: Selecting a data page type

you want to view the data in a sparse list object, a composite table object, a
pivot table object or in the form of a (rectangular) table. Additionally, you can
indicate that you want the view to be *sliced* (see also Section 10.4), by selecting
fixed elements for one or more dimensions. For every sliced dimension, Aɪᴍᴍs
will automatically add a floating index to the data page, allowing you to view
the data for every element in the sliced dimension.

If you want to always use the same data page settings for a particular identifier,   *Saving your*
you can save the choices you made in Figure 5.8. As a result, Aɪᴍᴍs will   *choice*
save the data page as an ordinary end-user page in the special **All Data Pages**
section of the **Page Manager** (see also Section 12.1). If you so desire, you can
further edit this page, and, for instance, add additional related identifiers to
it which will subsequently become visible when you view the identifier data in
the **Model Explorer**.

Whenever there is a page in the **All Data Pages** section of the page manager with the fixed name format [Data Page] followed by the name of an identifier of your model, AIMMS will use this page as the data page for that identifier. This enables you to copy a custom end-user page, that you want to use as a data page for one or more identifiers, to the **All Data Pages** section of the page manager, and rename it in the prescribed name format. When you remove a page from the **All Data Pages** section, AIMMS will again open a default data page for that identifier. If you hold down the **Shift** key while opening a data page, AIMMS will always use the default data page.

End-user page as data page

Normally, AIMMS will only allow you to open data pages of global identifiers of your model. However, within the AIMMS debugger (see also Section 8.1), AIMMS also supports data pages for local identifiers within a (debugged) procedure, enabling you to examine the contents of local identifiers during a debug session.

Global and local identifiers