
AIMMS Excel Add-In User's Guide - AIMMS Excel Add-In Interface

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

Copyright © 1993–2011 by Paragon Decision Technology B.V. All rights reserved.

Paragon Decision Technology B.V.	Paragon Decision Technology Inc.	Paragon Decision Technology Pte.
Schipholweg 1	500 108th Avenue NE	Ltd.
2034 LS Haarlem	Ste. # 1085	80 Raffles Place
The Netherlands	Bellevue, WA 98004	UOB Plaza 1, Level 36-01
Tel.: +31 23 5511512	USA	Singapore 048624
Fax: +31 23 5511517	Tel.: +1 425 458 4024	Tel.: +65 9640 4182
	Fax: +1 425 458 4025	

Email: info@aimms.com
WWW: www.aimms.com

AIMMS is a registered trademark of Paragon Decision Technology B.V. IBM ILOG CPLEX and sc CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Ziena Optimization, Inc. XPRESS-MP is a registered trademark of FICO Fair Isaac Corporation. MOSEK is a registered trademark of Mosek ApS. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. T_EX, L^AT_EX, and A_MS- \LaTeX are trademarks of the American Mathematical Society. LUCIDA is a registered trademark of Bigelow & Holmes Inc. ACROBAT is a registered trademark of Adobe Systems Inc. Other brands and their products are trademarks of their respective holders.

Information in this document is subject to change without notice and does not represent a commitment on the part of Paragon Decision Technology B.V. The software described in this document is furnished under a license agreement and may only be used and copied in accordance with the terms of the agreement. The documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paragon Decision Technology B.V.

Paragon Decision Technology B.V. makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Paragon Decision Technology B.V., its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claims for lost profits, fees or expenses of any nature or kind.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

This documentation was typeset by Paragon Decision Technology B.V. using L^AT_EX and the LUCIDA font family.

Part II

AIMMS Excel Add-In Reference

Chapter 3

The AIMMS Excel Add-In Interface

After you've successfully installed the AIMMS Excel add-in, you can show the interface window of the add-in by selecting the menu item **Interface Setup** from the **AIMMS**-menu or by clicking on the AIMMS logo in the Excel toolbar.

Showing the interface

The **AIMMS Interface Setup** dialog box consists of three tabs:

Three tabs

1. The **AIMMS Project** tab
2. The **Protection** tab
3. The **Execution Sequences** tab

Using these three tabs allows you to specify which interactions you want between your Excel spreadsheet and a certain AIMMS project. In the next three sections, the tabs are explained in detail.

3.1 The AIMMS Project tab

The **AIMMS Project** tab allows you to specify with which AIMMS project you want Excel to interact. Figure 3.1 shows this tab.

Function of the tab

In the **Project File** field you must enter the name of the AIMMS project you want to use in Excel. Make sure you also specify the right path (you don't have to specify a path if the project is located in the same directory as your Excel spreadsheet). Of course you can click on the **Browse** button to look for the project file on your system.

Project file

You can specify the directory containing your AIMMS configuration settings in the **Config Directory** field. When you leave this field empty, the default AIMMS configuration directory is used. Entering an invalid configuration directory raises an error. You can use the **Browse** button here as well to assist you.

Config directory

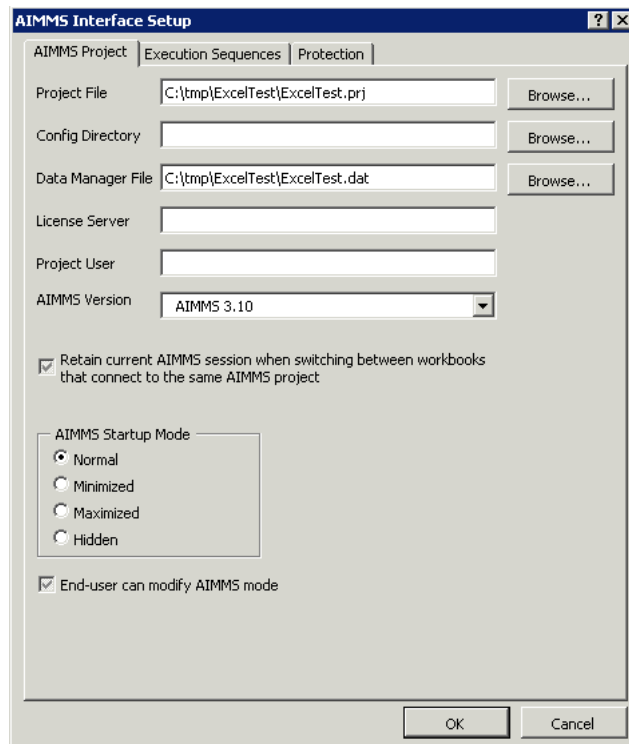


Figure 3.1: The AIMMS Project tab

To specify an alternative data manager file to open AIMMS with, you can enter the .dat-file you want in the **Data Manager File** field. Again, the **Browse** button can help you locating the right file.

Data manager file

You can use the **License Server** field to specify the host and port number of the server where AIMMS can obtain an AIMMS license and/or the VAR license(s) required to run the project. If you leave this field empty, AIMMS searches for the license file in the configuration directory and tries to find the required VAR licenses in either the configuration or the project directory. The format in which to enter this field is *host:port*.

License server

In the **Project User** field you can type the username and optionally the password of the person that uses the AIMMS project. When you want to specify a password, use the format *username:password* to enter the field.

Project user

In the **AIMMS Version** field, you can specify which version of AIMMS should be used to open the AIMMS project. The dropdown box is filled with all installed AIMMS versions on your system, from version 3.6 on. Selecting a different version in this field, only becomes effective when you have restarted Excel (you

AIMMS Version

will be notified by a pop-up dialog in this case). In case you, as the designer of an AIMMS Excel Add-In project, select a version here which is not installed on the machine of the user, the user will get an error message.

The checkbox just below the aforementioned five fields does essentially what its description promises. Checking the box will have the effect that when you change to another Excel workbook that uses the same AIMMS project as the current workbook does, the AIMMS project will not be closed and reopened when switching, but instead just stay opened. If you don't want this behavior, just uncheck the box.

Retain current session

The section **AIMMS Startup Mode** lets you select your preferred startup mode for the AIMMS project window. The four possible choices are:

Startup mode

- Normal
- Minimized
- Maximized
- Hidden

When you check the checkbox **End-user can modify AIMMS mode**, an extra **AIMMS** menu will appear in the Excel toolbar area. This menu allows the end-user of the Excel spreadsheet to pick his or her preferred AIMMS mode (the same modes that can be selected in the **AIMMS Startup Mode** section). Notice that the selected *startup* mode still applies; the new menu merely lets the end user switch to another mode *after* AIMMS has been started.

Modify AIMMS mode

3.2 The Protection tab

The **Protection** tab allows you to protect the interface setup with a password. Doing so prevents the end user of the Excel spreadsheet from making undesired changes to the interface that you carefully set up. Figure 3.2 shows the main part of this tab.

Function of the tab

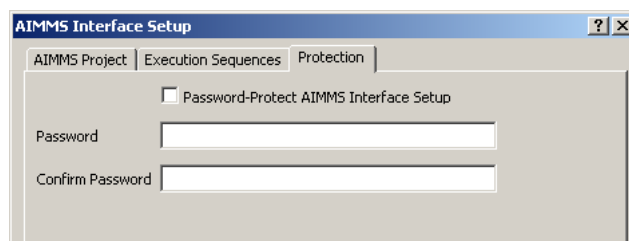


Figure 3.2: The **Protection** tab

In order to use the protection facility, you must check the checkbox **Password-Protect AIMMS Interface Setup**. *Password-protect*

In the **Password** field, you must type the password with which you want to protect the interface setup. As you type, the password is not literally shown on the screen, but is, as usual, replaced by a series of asterisks. *Password*

In the **Confirm Password** field you have to reenter the password, to guard you against typing errors. Click **OK** to apply the protection. *Confirm*

After having applied the password protection, trying to select the **Interface Setup** item from the **AIMMS** menu results in the displaying of a dialog that prompts you for your password. If you type in the wrong password, you're being notified and you're not allowed access to the interface setup window. *Password prompted*

3.3 The Execution Sequences tab

The **Execution Sequences** tab is the most important part of the interface setup window. With this tab, you can specify all the interactions between Excel and AIMMS that you want to execute. The interactions are ordered by means of *actions* and *execution sequences*. Figure 3.3 shows the tab. *Function of the tab*

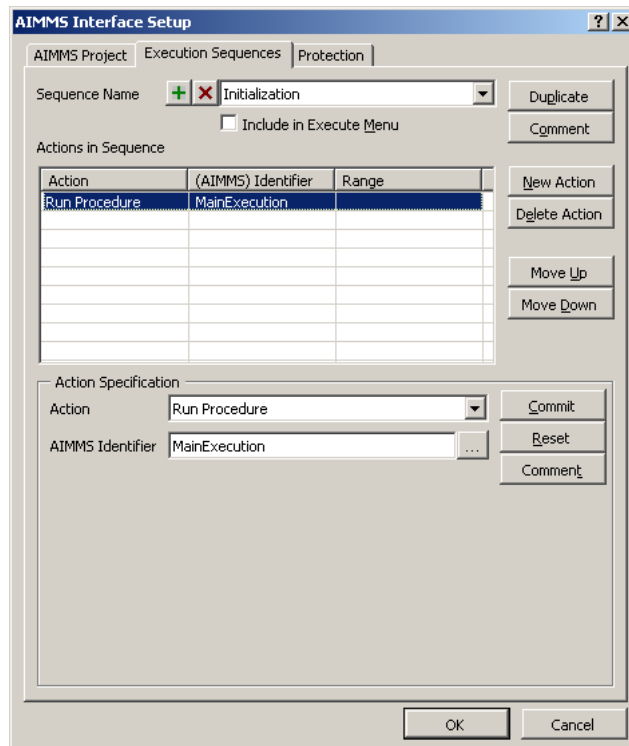


Figure 3.3: The Execution Sequences tab

An action is some kind of interaction between the Excel spreadsheet and the specified AIMMS project. Basically, there are six kinds of actions. You can:

What is an action?

1. Transfer data between AIMMS and Excel;
2. Empty an AIMMS identifier;
3. Update an AIMMS identifier;
4. Execute an AIMMS procedure;
5. Run an execution sequence (see section 3.3);
6. Run an Excel macro

The first kind of action, transferring data between AIMMS and Excel, comes in various forms.

You can easily manage all actions by selecting the one you need from a list and specifying additional information by filling in the required fields for the action, for example the cell area in the spreadsheet that you want to link with a certain AIMMS identifier.

Easily manageable

An execution sequence is nothing more than a list of actions to be executed in the specified order. Given the action **Run Sequence**, this implies that an execution sequence can call other execution sequences. Be careful not to create recursive execution sequences (execution sequences that—directly or indirectly—call themselves), as a recursively called sequence will lead to a run-time error.

What is an execution sequence?

In the **Sequence Name** field of the **Execution Sequences** tab you can type a name for the sequence that you want to create or you can select a yet existing sequence to alter. It's also possible to delete a complete sequence by clicking on the button with the red cross to the left of the edit field. To add a new sequence, just click on the button with the green +. If you have a sequence that you want to duplicate, click on the **Duplicate** button to do so. This allows you to create sequences that differ slightly or are based on each other: just create the first sequence, fill in all the actions (see below for details), duplicate the sequence and alter this newly created second sequence to your likings.

Sequence name

When you want a sequence to appear in the **Execute** submenu of the **AIMMS** menu, check the **Include in Execute Menu** checkbox.

Execute menu

With the part of the tab labeled **Actions in Sequence**, you can define the actions that together form an execution sequence. The order in which the actions are listed determines the order in which they will be executed if the execution sequence containing them is eventually run. This is the reason that the **Move Up** and **Move Down** buttons are provided. Using those buttons you can change the order in which the actions have to be executed.

Actions in sequence

With the **New Action** and **Delete Action** buttons you can create new (initially empty) actions and delete already existing actions, respectively.

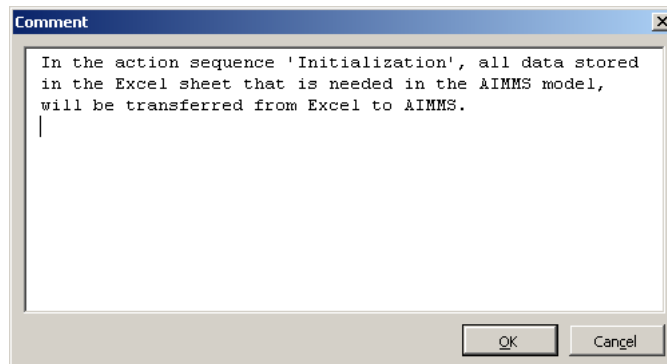
New and delete

To assign a specific interaction between Excel and AIMMS to an action in the action list, select the action by clicking on it and pick an interaction from the list to the right of the **Action** label. The exact interpretation of all action types is discussed later on in this section.

Action specification

In order to increase the maintainability of your execution sequences and actions, you can supply comments on them, using the two **Comment** buttons in the execution sequences tab. The button located in the upper right corner of the tab allows you to enter a comment on the selected execution sequence, while you can use the button in the lower right corner to comment on the selected action. When you click on either button, the dialog shown in figure 3.4 pops up. You can enter your comment and click on **OK** to accept it.

Comments

Figure 3.4: The **Comment** dialog

Not all possible actions have the same set of parameters that has to be specified in order to configure the action to your needs. Some actions only need one parameter (for example: the *Empty Identifier* action only needs the name of the AIMMS identifier that you want to empty), others need more (for example: the *Retrieve Set Data* action, which needs the name of the AIMMS set and the Excel cell range into which you want the set data to be copied).

Action parameters

This paragraph introduces all possible action parameters together with a description:

All possible action parameters

- **AIMMS Identifier**

Because an action almost always represents an interaction with an AIMMS project, it is logical that many actions require an AIMMS identifier as a parameter. This action parameter simply specifies to which AIMMS identifier the action applies.

- **Data Range**

In order to know which cells in your Excel spreadsheet are used to interact with AIMMS, many actions require the **Data Range** parameter to be specified. In the edit field for this parameter, you must type in a valid Excel cell range. Notice that it's possible to select a valid range using the mouse by first clicking on the **Select** button. It's also possible to assign a name to the specified range, to make the action more readable and the range better reusable. To assign a name to the range, use the **Name** button. Using named ranges rather than explicit ranges also has the advantage that the range will automatically be updated when you insert or delete rows or columns that are part of the range.

- **Row Range, Row Mode, Column Range and Column Mode**

All actions involving the transfer of tabular data (such as the **Retrieve Table** action) require the action parameters **Row Range**, **Row Mode**, **Column Range** and **Column Mode** to be specified. For the meaning

of these action parameters, see the explanation of the method `Identifier.RetrieveTable` in the AIMMS COM Object Reference. Just as with the **Data Range** action parameter, you can also assign a name to the **Row Range** and the **Column Range** ranges.

- **Sequence**

Obviously, when you select the **Run Sequence** action, the name of the sequence that you want to run is needed. Using this action parameter you can provide it.

- **Excel Macro**

The same can be said about the **Excel Macro** action parameter. This parameter is required in order to be able to run an Excel macro with the action **Run Excel Macro**.

You don't have to enter the action parameters by hand. Instead, a number of dialogs and combo boxes are supplied to help you enter the correct data for the action parameters. The various dialogs are explained in more detail in subsection 3.3.1.

*Action
parameter
dialogs*

Which action needs which action parameter(s) is automatically made clear, because on selecting an action only the required parameter fields for the action are displayed.

*Required fields
only*

After you've entered an action and its action parameters, you can add it to the current execution sequence by clicking on the **Commit** button. If you decide to discard your changes to the action, you can do so by clicking on the **Reset** button.

Commit or reset

In the subsequent sections all possible actions are described in detail, accompanied by short and clear examples. Almost all actions and the results of executing them are displayed in figures. The general structure of the explanation of an action consists of a description, one or more examples and some remarks, if any.

*Actions
explained*

3.3.1 The action parameter dialogs

As already stated in the previous section, there is a number of dialogs which you can use to enter the data for the various action parameters. In some cases, you can only enter the data by using a dialog (for example, while supplying the **Data Range** action parameter), while in other cases you can choose to enter the data by hand or to modify the dialog-provided data afterwards. This way, you can first select an AIMMS identifier using the dialog and then modify the selected identifier to create a slice of it. You are strongly advised to use the

Introduction

dialogs as much as possible, in order to prevent typing errors which can lead to annoying error messages. The next paragraphs explain the various dialogs in detail.

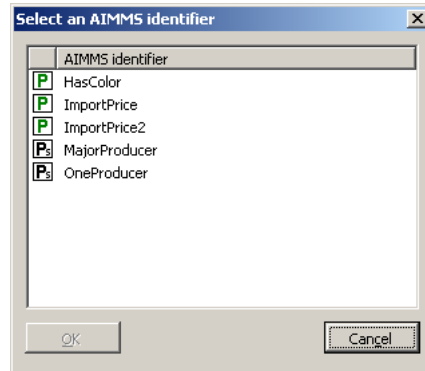


Figure 3.5: The AIMMS identifier dialog

You can activate the AIMMS identifier dialog, as shown in figure 3.5, by clicking on the button labelled ..., next to the AIMMS identifier field. The dialog allows you to select an AIMMS identifier for the current action. Only the relevant AIMMS identifiers are displayed, e.g. only sets are displayed when you must select an AIMMS identifier for the action **Retrieve Set Data**. The identifiers are initially sorted alphabetically, but you can also sort the list on identifier type by clicking on the column header above the identifier type icons. To have the list sorted alphabetically again, click on the column header above the identifier names.

The AIMMS identifier dialog

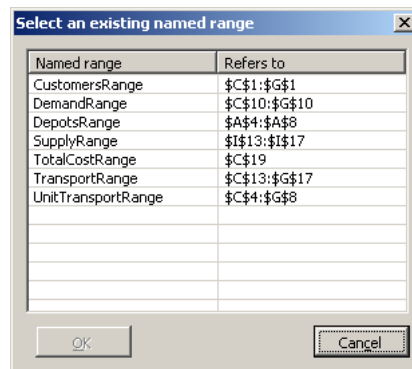


Figure 3.6: The named range dialog

When you've already assigned names to certain ranges of your Excel sheet, you can reselect them easily by using the named range dialog (see figure 3.6). You can activate this dialog by clicking on the buttons labelled ..., next to either the **Data Range**, the **Columns Range** or the **Rows Range** field. The existing named ranges are sorted alphabetically and the corresponding sheet ranges in the usual Excel notation are displayed as well.

The named range dialog

Note that when you've created more than one named range that correspond to the same physical range in your spreadsheet, the name you created first will be transferred to the range field, even if you've selected another name! This rather odd behavior is imposed on the add-in by the Visual Basic language built in Excel. It doesn't have any implication though on the correct functioning of the add-in.

Odd selecting behavior with multiply named ranges

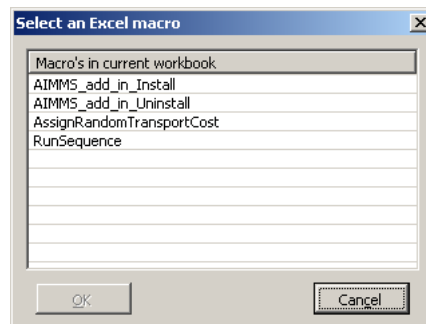


Figure 3.7: The Excel macro dialog

When you've selected the action **Run Excel Macro**, you can click on the button labelled ... next to the **Excel Macro** field to activate the Excel macro dialog (see figure 3.7). This dialog lists all available Excel macros in the current workbook, ordered alphabetically. In case you get the error

Excel macro dialog

```
Run-time error '1004'  
Programmatic access to Visual basic Project is not trusted.
```

you need to tell Excel first that programmatic access to Visual Basic projects is trusted. Please do so by going to *Tools/Macro/Security...* and going to the *Trusted Sources* tab. There you have to check the check box labeled *Trust Access to Visual Basic Project*.

3.3.2 Retrieve Set Data

With the action **Retrieve Set Data** you can transfer the data of an AIMMS set to Excel. The action parameters you have to supply are **AIMMS Identifier** and **Data Range**. Consider an AIMMS project containing just one set:

The action

```
Fruits := data { Pineapple, Mango, Kiwi, Banana, Orange };
```

The screenshot shows a dialog box titled "Action Specification". It contains the following fields and buttons:

- Action:** A dropdown menu with "Retrieve Set Data" selected.
- AIMMS Identifier:** A text field containing "Fruits".
- Associated Ranges in Spreadsheet:** A section containing a **Data Range** text field with "Sheet1!\$A\$1:\$A\$5" and "Select" and "Name" buttons.
- Buttons:** "Commit", "Reset", and "Comment" buttons are located on the right side of the dialog.

Figure 3.8: The action **Retrieve Set Data**

By specifying the action as in figure 3.8, all set elements are retrieved in the cells A1:A5 of sheet 1 of the Excel spreadsheet. Figure 3.9 shows the result after executing the action.

Example

	A
1	Pineapple
2	Mango
3	Kiwi
4	Banana
5	Orange
6	

Figure 3.9: The result of **Retrieve Set Data**

Specifying a horizontal cell range would result in the corresponding cells being filled with the set elements. Specifying a range that contains more cells than the number of elements of the set to be retrieved is allowed, whereas specifying a range that contains less cells results in an error. Any ordering in the set specified by the **ORDER BY** attribute in AIMMS is pertained by Excel. If the example set would be ordered alphabetically in AIMMS and cell range A1:E1 would be supplied, the result would be as in figure 3.10.

Remarks

	A	B	C	D	E
1	Banana	Kiwi	Mango	Orange	Pineapple
2					

Figure 3.10: The action **Retrieve Set Data** with ordering

3.3.3 Assign Set Data

The action **Assign Set Data** is the opposite action of **Retrieve Set Data**. With this action you can transfer cell data from Excel into an AIMMS set. The action parameters required are **AIMMS Identifier** and **Data Range**.

The action

	A
1	Lemon
2	Peach
3	Lime
4	
5	

Figure 3.11: Input for the action **Assign Set Data**

Suppose you have cell range A1:A3 in Excel filled as in figure 3.11. When the same set as in the previous subsection exists in AIMMS, the result of the action specified in 3.12 would be the set

Example

```
Fruits := data { Lemon, Peach, Lime };
```

in AIMMS.

Figure 3.12: The action **Assign Set Data**

Make sure that you don't assign data to a set in AIMMS that has a definition, as this results in an error. This is true for all actions of the AIMMS Excel add-in that assign data to AIMMS identifiers with a non-empty definition attribute.

Remark

3.3.4 Retrieve Array Data

The action **Retrieve Array Data** allows you to transfer data from AIMMS identifiers, like parameters and variables, into an Excel spreadsheet. You have to specify the action parameters **AIMMS Identifier** and **Data Range**.

The action

Consider an AIMMS project containing the following sets:

Example

```
Fruits := data { Pineapple, Mango, Kiwi, Banana, Orange };
```

```
Colors := data { blue, green, orange, red, yellow };
```

	HasColor				
	blue	green	orange	red	yellow
Banana		1		1	1
Kiwi		1			
Mango			1		
Orange			1		
Pineapple					1

Figure 3.13: The data of the AIMMS parameter HasColor

A binary parameter HasColor(f , c) is also added to the model. The indices f and c are indices into the sets Fruits and Colors, respectively. The parameter is defined as in figure 3.13. The result of the action specified in figure 3.14 is shown in figure 3.15.

Action Specification

Action: Retrieve Array Data

AIMMS Identifier: HasColor(f, c)

Associated Ranges in Spreadsheet

Data Range: Sheet1!\$A\$1:\$E\$5

Figure 3.14: The action Retrieve Array Data

Make sure that the data range you supply is of the same dimension as the AIMMS identifier you want to read in (so, in practice, you won't specify dimensions greater than two).

Remark

	A	B	C	D	E
1	0	0	0	0	1
2	0	1	0	0	0
3	0	0	0	1	0
4	0	0	1	0	0
5	0	0	0	0	1

Figure 3.15: The result of Retrieve Array Data

Please note the notation of the AIMMS identifier HasColor in figure 3.14. As you can see, both the indices f and c are explicitly written. They are not required, however, but they do improve the readability of an action, since they show the dimension of the identifier and clarify the relation of the identifier with the sets in your model. Of course, when you work with identifier *slices*, the indices *are* required. For example, to only retrieve the data of yellow fruits, you would have to specify HasColor(f , 'yellow').

No indices required

3.3.5 Assign Array Data

The action **Assign Array Data** is the opposite action of **Retrieve Array Data**. With this action you can transfer zero-, one- or two-dimensional data from an Excel spreadsheet to an AIMMS parameter of the same dimension. The action parameters **AIMMS Identifier** and **Data Range** are required.

The action

Figure 3.16: The action **Assign Array Data**

Consider the fruit example once again, extended with a string parameter `MajorProducer(f)`, that holds a major producing country of the given fruit. You can fill this string parameter using Excel by executing the action from figure 3.16 on the data that is shown in figure 3.17.

Example

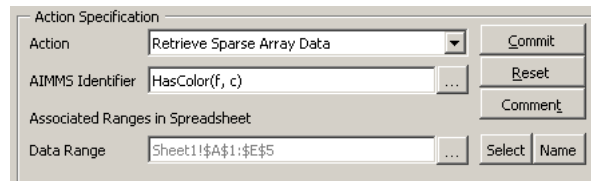
	A	B
1	Fruit	Main Producer
2	Banana	Ecuador
3	Kiwi	New Zealand
4	Mango	Mali
5	Orange	Brazil
6	Pineapple	Thailand
7		

Figure 3.17: The data for **Assign Array Data**

3.3.6 Retrieve Sparse Array Data

If you use the action **Retrieve Array Data** and the AIMMS identifier that you retrieve contains default values, those default values are displayed in your Excel spreadsheet as zeroes (see for example figure 3.15). You might want to display those values as empty cells in Excel. In order to do that, you can use the action **Retrieve Sparse Array Data**. This action requires the action parameters **AIMMS Identifier** and **Data Range** to be supplied.

The action

Figure 3.18: The action **Retrieve Sparse Array Data**

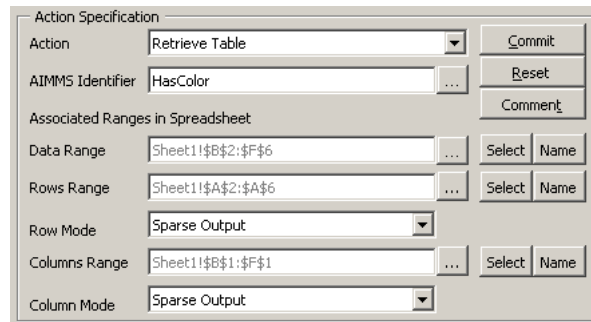
To get the output from figure 3.19 instead of the output of figure 3.15, all you need to do is to execute the action from figure 3.18. *Example*

	A	B	C	D	E
1					1
2		1			
3				1	
4			1		
5					1

Figure 3.19: The result of **Retrieve Sparse Array Data**

3.3.7 Retrieve Table

The action **Retrieve Table** is one of the more complex actions. Though it may require some effort to understand how it works, it's a very powerful action. For additional details on the action, you are advised to read the corresponding section in the AimmsCOM Function Reference. *The action*

Figure 3.20: An example of the action **Retrieve Table**

The action **Retrieve Table** is one of the actions that transfer *tabular data* between AIMMS and Excel. This means that not only a multidimensional AIMMS identifier is transferred between Excel and AIMMS, but also the possible values of the indices involved with the transferred identifier. When, for example, you take a look at the result of **Retrieve Sparse Array Data** from figure 3.19, it is obvious that the presentation of this data would be a lot more meaningful if *Tabular data*

the fruits and colors would be listed as row- and column headers. The *tabular* actions can do that for you. This action requires the action parameters **AIMMS Identifier**, **Data Range**, **Rows Range**, **Row Mode**, **Columns Range** and **Column Mode**.

	A	B	C	D	E	F	G
1		green	orange	red	yellow		
2	Pineapple	0	0	0	1	0	
3	Mango	0	0	1	0	0	
4	Kiwi	1	0	0	0	0	
5	Banana	0	0	0	1	0	
6	Orange	0	1	0	0	0	
7							

Figure 3.21: The result of the action of figure 3.20

Some examples will probably help to understand the **Retrieve Table** action. Consider once again the HasColor identifier from figure 3.13. When you execute the action from figure 3.20, you get the output of figure 3.21 as a result. Notice that the row- and column headers are put into the cells of the ranges supplied by the action parameters **Rows Range** and **Columns Range** respectively.

Example

As you can see, a complete *table* (e.g. not only the data itself, but also the row- and column values) is indeed retrieved from AIMMS. Notice the absence of the column value blue. The reason for this is that the corresponding column (column F) contains only zeroes. The specification of Sparse Output as column mode for the action, has this effect. Specifying Dense Output instead would have shown the value blue in cell F1.

A complete table

As an example of the more advanced possibilities of the **Retrieve Table** action, reconsider the fruit example. Two additional sets are added:

A more advanced example

```
Countries := data { Cuba, Ecuador, France, 'Ivory Coast',
                  Mali, 'New Zealand', Spain, USA };
```

```
TransportType := data { Plane, Ship };
```

The three-dimensional identifier

```
ImportPrice(f,c,t)
```

is also added. It holds the import price of a particular type of fruit, from the given country using the given means of transport, in guilders per kilogram. In the set ImportPrice, f is an index into the set Fruits, c is an index into the set Countries and t is an index into the set TransportType. Executing the action from figure 3.23 will result in the output of figure 3.23.

Figure 3.22: An advanced example of the action **Retrieve Table**

The result displayed in figure 3.23 emphasizes some characteristics of the **Retrieve Table** action (the prices displayed are prices that have been put into the model before executing the action).

Remarks

	A	B	C	D
1			Plane	Ship
2	Pineapple	Ivory Coast	6,00	4,30
3	Pineapple	USA	5,80	5,25
4	Mango	Mali	4,00	2,80
5	Mango	USA	3,90	3,05
6	Kiwi	New Zealand	3,80	3,10
7	Kiwi	USA	3,40	2,80
8	Banana	Cuba	3,20	2,50
9	Banana	Ecuador	3,10	2,50
10	Banana	Spain	2,95	2,45
11	Orange	Cuba	4,00	3,25
12	Orange	France	3,50	2,90
13	Orange	Spain	3,70	2,65
14			0	0
15			0	0
16			0	0
17			0	0
18			0	0

Figure 3.23: The result of the action of figure

You've probably noticed the extra zeroes below the 'active' part of the table. This behavior is due to the fact that the column- and row values are retrieved in a sparse manner, while the actual data is not. You can eliminate the extra zeroes by using the similar action **Retrieve Sparse Table** instead, using exactly the same action parameters.

Extra zeroes

The result displayed in figure 3.23 is in fact just one of many possible representations of the table. For example, you might prefer to have the fruit types listed as column values and the combination of country and transport types as row values. The display that the AIMMS Excel add-in (more precisely: the AIMMS COM object) comes up with, depends on the order of the indices of the corresponding AIMMS identifier. The first r rows are filled with the first r indices. The columns are filled with the remaining c indices. Note that

Displaying of row- and column values

$r + c$ must always equal the dimension of the AIMMS identifier displayed. To actually have the fruit types listed as column values, the AIMMS identifier `ImportPrice(f,c,t)` should be redefined as either `ImportPrice(c,t,f)` or `ImportPrice(t,c,f)`, specifying `f` as the last index. Of course, the action parameters **Data Range**, **Columns Range** and **Rows Range** would also have to be changed to achieve the alternative display format.

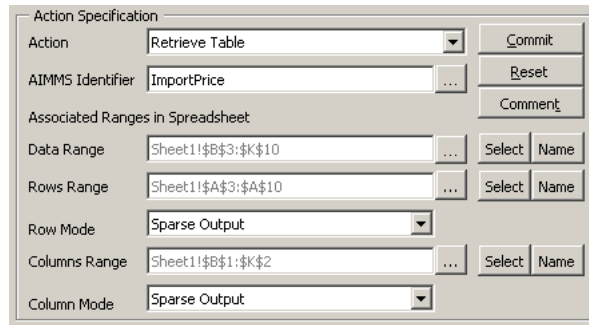


Figure 3.24: Action to achieve an alternative row- and column value display

You're not restricted to one-dimensional column values, like in the last two examples. You can, for example, just as well display the fruit- and transport type as column values, leaving the countries as the only remaining row values. In order to achieve this output (shown in figure 3.25), you first have to redefine the AIMMS identifier `ImportPrice(f,c,t)` to `ImportPrice(c,f,t)` (or `ImportPrice(c,t,f)`, depending on how you want the column values to appear precisely). Then, you need to modify the various action parameters of the action into those shown in figure 3.24.

More column values

	A	B	C	D	E	F	G	H	I	J	K
1		Pineapple	Pineapple	Mango	Mango	Kiwi	Kiwi	Banana	Banana	Orange	Orange
2		Plane	Ship	Plane	Ship	Plane	Ship	Plane	Ship	Plane	Ship
3	Cuba	0,00	0,00	0,00	0,00	0,00	0,00	3,20	2,50	4,00	3,25
4	Ecuador	0,00	0,00	0,00	0,00	0,00	0,00	3,10	2,50	0,00	0,00
5	France	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3,50	2,90
6	Ivory Coast	6,00	4,30	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
7	Mali	0,00	0,00	4,00	2,80	0,00	0,00	0,00	0,00	0,00	0,00
8	New Zealand	0,00	0,00	0,00	0,00	3,80	3,10	0,00	0,00	0,00	0,00
9	Spain	0,00	0,00	0,00	0,00	0,00	0,00	2,95	2,45	3,70	2,65
10	USA	5,80	5,25	3,90	3,05	3,40	2,80	0,00	0,00	0,00	0,00
11											
12											

Figure 3.25: The altered row- and column display

As you can see, there are four possible choices for the action parameters **Rows Mode** and **Columns Mode**. For details on those choices, please refer to the section 'Identifier.RetrieveTable' in the AIMMS COM Object Reference.

Rows- and columns mode

3.3.8 Retrieve Sparse Table

As already noticed in the previous subsection, the action **Retrieve Sparse Table** is similar to **Retrieve Table**. The only difference between the two has already been pointed out: cells containing a zero value are eliminated from the resulting display of the action.

The action

	A	B	C	D	E
1			Plane	Ship	
2	Pineapple	Ivory Coast	6,00	4,30	
3	Pineapple	USA	5,80	5,25	
4	Mango	Mali	4,00	2,80	
5	Mango	USA	3,90	3,05	
6	Kiwi	New Zealand	3,80	3,10	
7	Kiwi	USA	3,40	2,80	
8	Banana	Cuba	3,20	2,50	
9	Banana	Ecuador	3,10	2,50	
10	Banana	Spain	2,95	2,45	
11	Orange	Cuba	4,00	3,25	
12	Orange	France	3,50	2,90	
13	Orange	Spain	3,70	2,65	
14					
15					
16					

Figure 3.26: The output of **Retrieve Sparse Table**

	A	B	C	D	E	F	G	H	I	J	K
1		Pineapple	Pineapple	Mango	Mango	Kiwi	Kiwi	Banana	Banana	Orange	Orange
2		Plane	Ship	Plane	Ship	Plane	Ship	Plane	Ship	Plane	Ship
3	Cuba							3,20	2,50	4,00	3,25
4	Ecuador							3,10	2,50		
5	France									3,50	2,90
6	Ivory Coast	6,00	4,30								
7	Mali			4,00	2,80						
8	New Zealand					3,80	3,10				
9	Spain							2,95	2,45	3,70	2,65
10	USA	5,80	5,25	3,90	3,05	3,40	2,80				
11											

Figure 3.27: Another example of the action **Retrieve Sparse Table**

For example, a **Retrieve Sparse Table** action with exactly the same action parameters as the **Retrieve Table** action of figure 3.23 would result in the output of figure 3.26.

Example

As another example, a **Retrieve Sparse Table** action with the same action parameters as the **Retrieve Table** action of figure 3.24 results in the output of figure 3.27.

Another example

3.3.9 Assign Table

The action **Assign Table** is the opposite action of **Retrieve Table**. This means that you can transfer data, represented in a tabular format, from Excel to AIMMS. This action requires the action parameters **AIMMS Identifier**, **Data Range**, **Rows Range**, **Row Mode**, **Columns Range** and **Column Mode**.

The action

The screenshot shows a dialog box titled "Action Specification". It contains several fields and buttons. The "Action" dropdown is set to "Assign Table". The "AIMMS Identifier" text box contains "ImportPrice". Under the "Associated Ranges in Spreadsheet" section, there are three rows: "Data Range" with "Sheet1!\$B\$3:\$K\$10", "Rows Range" with "Sheet1!\$A\$3:\$A\$10", and "Columns Range" with "Sheet1!\$B\$1:\$K\$2". Each of these three rows has "Select" and "Name" buttons to its right. The "Row Mode" dropdown is set to "User Input", and the "Column Mode" dropdown is also set to "User Input". On the right side of the dialog, there are three buttons: "Commit", "Reset", and "Comment".

Figure 3.28: An **Assign Table** action

Consider the output displayed in figure 3.27. To use this Excel sheet as input rather than output, you have to create an **Assign Table** action with almost the same action parameters as the **Retrieve Sparse Table** action that displayed the output first. This action is shown in figure 3.28. When, for example, you enter the value 4.50 in cell C3 (saying that importing 1 kg of Cuban pineapples by ship costs 4.50 guilders) and then execute the action in figure 3.28, the complete table is transferred to AIMMS. You can check this by executing the **Retrieve Sparse Table** action that displayed the table in the first place.

Example

Notice the two action parameters **Rows Range** and **Columns Range**. Both are set to *User Input* in the example. This means that the data in the table is considered to be input by the user. If you don't specify a rows range or a columns range (for example, in case of a one-dimensional table), set the corresponding rows or columns mode to *Non Existing*. For further details, please refer to the AimmsCOM Function Reference.

Remark

3.3.10 Empty Identifier

With the action **Empty Identifier** you can delete the contents of an AIMMS identifier (slice). This action is equivalent with the AIMMS Empty statement. You only have to specify the action parameter **AIMMS Identifier**.

The action

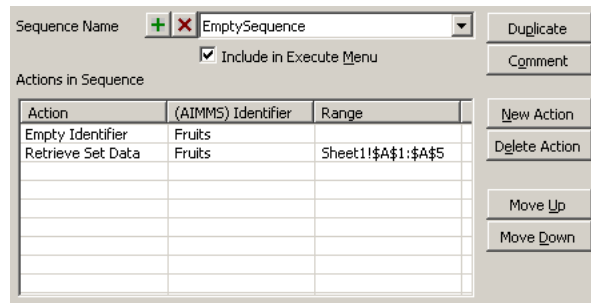


Figure 3.29: A sequence containing an **Empty** action

Consider the original Fruits set once again. Executing the action sequence *Example* from figure 3.30 results in the empty spreadsheet cells from figure 3.30.

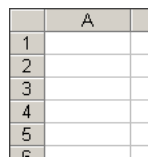


Figure 3.30: The result of the action sequence in figure

3.3.11 Update Identifier

It is common practice in AIMMS models to define identifiers in terms of other identifiers. To ensure that an identifier that is defined in such a way is up-to-date after adjusting the data of the identifier it depends on, you can explicitly update the dependent identifier. *The action*

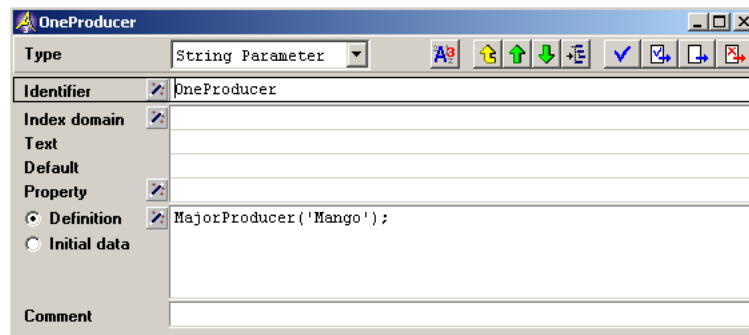


Figure 3.31: The definition of the OneProducer identifier

In the context of the fruit example, consider an AIMMS string parameter One-Producer, that has a definition in terms of the AIMMS identifier MajorProducer. Figure 3.31 shows this identifier.

Example

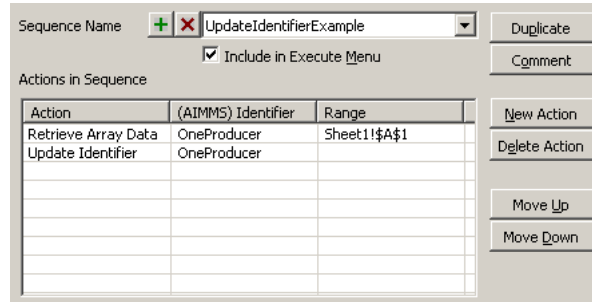


Figure 3.32: An action sequence containing an **Update** action

The first time you execute the sequence from figure 3.32, cell A1 of Sheet1 remains empty. After the second time, this cell will be filled with Mali.

3.3.12 Run Procedure

With the action **Run Procedure** you can run AIMMS procedures. Defining this action is straightforward: just specify the AIMMS procedure that you want to execute as action parameter **AIMMS Identifier**.

The action

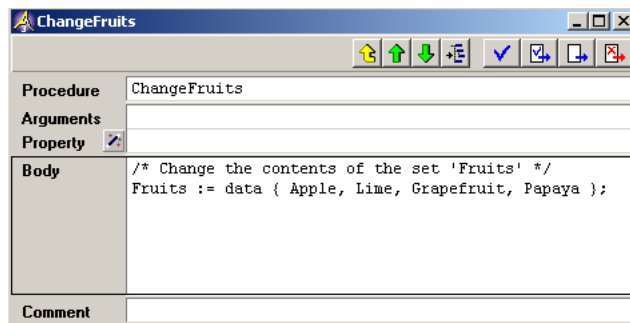


Figure 3.33: The definition of procedure ChangeFruits

Consider an AIMMS model containing the original Fruits set. This model also contains a procedure ChangeFruits, which is defined as shown in figure 3.33. You can execute this procedure from within Excel using the **Run Procedure** action, like in the sequence defined in figure 3.35. Running this sequence results in the data in figure 3.35.

Example

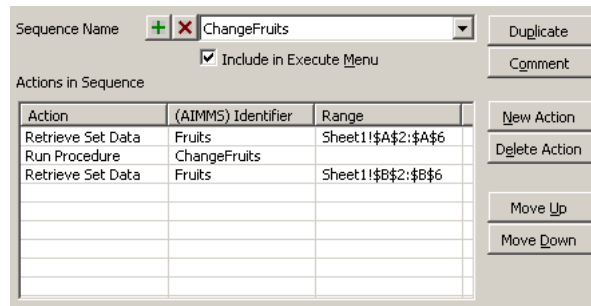


Figure 3.34: A sequence including the **Run Procedure** action

Notice the headers above the two columns in the output. Those headers were *Remark* not produced by the AIMMS Excel add-in, but put there by hand.

	A	B
1	Old fruits	New fruits
2	Banana	Apple
3	Kiwi	Grapefruit
4	Mango	Lime
5	Orange	Papaya
6	Pineapple	
7		

Figure 3.35: The result of executing the sequence from figure

3.3.13 Run Sequence

The action **Run Sequence** allows you to run an execution sequence from another execution sequence. This offers the possibility of dividing complex sequences in smaller parts, which improves the readability and maintainability of your interface with AIMMS. The only action parameter you have to specify is the **Sequence**. To facilitate the selecting of a sequence, a combo box (listing all existing execution sequences) is provided for this action, from which you can choose. *The action*

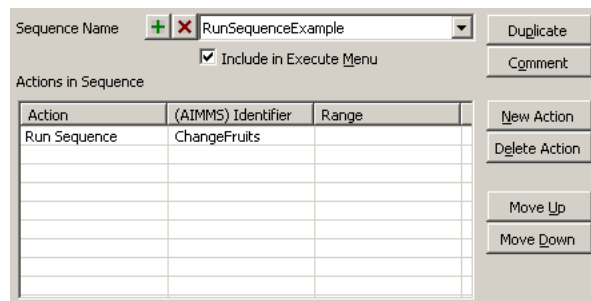


Figure 3.36: A sequence that executes another sequence

Consider the execution sequence executed in subsection 3.3.12. As an example of the use of the action **Run Sequence**, figure 3.36 shows a sequence containing this action. More precisely, the sequence from subsection 3.3.12 is started from the action sequence. Of course, the result is the same as in figure 3.35.

Example

Notice that the header **AIMMS Identifier(s)** in figure 3.36 is not correct in the case of a **Run Sequence** action.

Remark

3.3.14 Run Excel Macro

```
Sub TestMacro()
    'Place some text in cell A1 of Sheet1.
    Sheet1.Cells(1, 1) = "MacroTest"
End Sub
```

Figure 3.37: A small Excel macro

To allow for a better integration of the AIMMS Excel add-in with the Visual Basic environment of Excel itself, the action **Run Excel Macro** is defined. With this action, you can run Excel macro's from within action sequences.

The action



Figure 3.38: The action **Run Excel Macro**

As an example, consider the Excel macro as shown in figure 3.37. The action in figure 3.38 will run this macro. After running the sequence that contains the action, the result shown in figure 3.39 is produced.

Example

	A	B
1	MacroTest	
2		

Figure 3.39: The result of **Run Excel Macro**