AIMMS

The Excel Add-In User's Guide

AIMMS 4

February 22, 2018

AIMMS

The Excel Add-In User's Guide

AIMMS

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
Shanghai Representative Office
Middle Huaihai Road 333
Shuion Plaza, Room 1206
Shanghai
China
Tel.: +86 21 51160733

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

Contents

Part I

AIMMS Excel Add-In Tutorial

Chapter 1

Introduction to the AIMMS Excel Add-In

The AIMMS *modeling system*

AIMMS is a very advanced optimization modeling tool, with which it's possible to implement optimization-based, but also non-optimization based, applications at a very high level. Because of the clear symbolic representation of a model in AIMMS, which can be structured in a tree, AIMMS is very well suited to

- specify large-scale or complex optimization models, and
- implement possibly complex algorithms, including pre- and post-processing of data, around such optimization models.

In addition, because all implementation details are hidden from the modeler, AIMMS can be used to build an optimization-based application directly by application engineers, rather than having to hire programmers to build the actual application based on a specification on paper of the optimization model.

Combining AIMMS *with other applications*

Despite these powerful capabilities of AIMMS, there are situations imaginable where it would be useful to be able to easily make use of the optimization power of AIMMS within other applications. For example, end-users might insist on using the common spreadsheet interface of Microsoft Excel for data entry, while the underlying optimization model could be more easily specified in AIMMS.

The Excel add-in

The AIMMS Excel add-in provides an easy-to-use interface for using the power of AIMMS within Microsoft Excel spreadsheets. Although Microsoft Excel does offer some optimization functionality itself through its Solver add-in, there are several reasons why using the AIMMS Excel add-in would be a much better choice for integrating optimization in a spreadsheet:

- Optimization models can be specified much easier and more intuitively in AIMMS.
- Optimization models (and especially large-scale models) in Excel cannot be maintained very easily because the formulation is hidden in the cell formulas.
- Excel optimization models suffer from "dimensional arthritis", i.e. the dimension of a multidimensional collection of variables can hardly be modified once the model is formulated.
- Optimization models in AIMMS solve much faster in general.

When to use

You should seriously consider using the AIMMS Excel add-in to implement optimization tasks in Excel, if

- the optimization model to be implemented is large-scale or very complex, and
- you are worried about the maintainability of the optimization model in Excel.

Separation of data and algorithms

When an AIMMS model is combined with a Microsoft Excel spreadsheet, only the data needs to be represented in Excel, together with a specification of how Excel must interact with AIMMS. This means that the data and the algorithms for optimization can be separated, which allows the Excel user to focus on the part that is important to him, namely the data. The algorithms for optimization should be contained in the AIMMS model.

Choice of data exposure

The fact that AIMMS and Excel are being combined also has the advantage that the developer of the AIMMS model and the Excel spreadsheet can decide which data is accessible to the end user: this data can be put in the spreadsheet, while data that is not considered to be of use to the end user can remain in the AIMMS model. This offers the advantage that end users are only confronted with data that's important and meaningful to them.

Improving existing applications

The fact that Microsoft Excel is very widespread makes it plausible that certain Excel applications already exist that make use of the solving capabilities of Excel. Those applications can now be improved by using the optimization power of AIMMS. The AIMMS Excel add-in makes it possible.

The interaction

The interaction between AIMMS and Excel offers the possibility to exchange data between the two applications and to execute AIMMS-procedures from within Excel. You can control those processes through the provided interface of the Excel add-in by defining so-called *execution sequences* (see section 3.3).

Chapter 2

Installation and Example

This chapter

This chapter describes in detail how to install the AIMMS Excel add-in. In addition it provides an example spreadsheet illustrating most of the features of the add-in.

2.1 Installation

Automatic installation

In order to be able to use the AIMMS Excel add-in, the add-in has to be installed in your copy of Excel. This task will be automatically performed for you if you open the example that is described in section 2.2. The Excel add-in is integrated with the AIMMS Launcher and available after running the AIMMS Launcher for the first time.

How?

After opening the Excel file of the example, you might be asked whether you want to enable macros. Please click on **Enable Macros**, since the automatic installation is done with an Excel macro. After clicking on this button, you should receive the message shown in figure 2.1.

Figure 2.1: The message after a successful installation of the add-in

Changed Excel interface

When the installation is successful, you should notice the appearance of a new **AIMMS** menu item, an **Execute** toolbar item and a toolbar item with the AIMMS logo (see figure 2.2). They form the starting point for using the AIMMS Excel add-in.

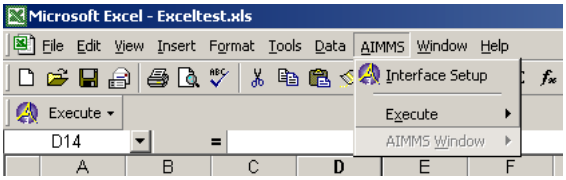


Figure 2.2: The **AIMMS** menu and the **Execute** toolbar item

Uninstall

When you close the example, you're being asked whether you want to keep the AIMMS Excel add-in installed or not. If you choose to' uninstall, you can always have the add-in automatically reinstalled by opening the example again.

Manual installation

In addition to installing the add-in by running the included example, you can of course also install and uninstall the AIMMS Excel add-in manually through the **Tools-Add Ins** menu in Excel versions previous to Excel 2007. In Excel 2007, you should:

- click the **Microsoft Office Button**
- click **Excel Options**
- click the **Add-Ins** category
- in the **Manage** box, click **Excel Add-ins**
- click **Go** to navigate to the AIMMS Excel Add-In and select it.

For further information about installing and uninstalling add-ins in this manner, you are referred to the Excel documentation.

2.2 An example spreadsheet

Transport model example

Your AIMMS system includes an Excel spreadsheet `ExcelAddInExample.xls` that makes calls to the well-known transport model in AIMMS using the AIMMS Excel add-in. You can find it by opening the `Excel Link.aimmspack` file that can be found in the `Examples` directory of AIMMS or in the Index of all Examples (see link on the startpage of AIMMS). Opening the example will unpack the `.aimmspack` file after which you can access `ExcelAddInExample.xls` via the Windows explorer. As illustrated in Figure 2.3, the spreadsheet contains all relevant input- and output data of the transport model, such as the unit transport cost, the supply and demand on the input side, as well as the actual transport on the output side.

How to solve?

To solve the model for given demand and supply, you can run the **Execute-Main** command from the AIMMS toolbar illustrated in Figure 2.4

	A	B	C	D	E	F	G	H	I	J
1		**Customers**	cust-1	cust-2	cust-3	cust-4	cust-5			
2										
3	**Depots**	**UnitCost**								
4	dep-1		0.63	0.45	0.67	0.63	0.85			
5	dep-2		0.10	0.00	0.04	0.65	0.84			
6	dep-3		0.67	0.66	0.21	0.14	0.07			
7	dep-4		0.82	0.86	0.11	0.67	0.76			
8	dep-5		0.15	0.13	0.17	0.85	0.31			
9										
10		**Demand**	25	25	25	25	25			
11									**Supply**	**Supply**
12	**Depots**	**Transport**							**Available**	**Left**
13	dep-1					21			27	6
14	dep-2		25		2				27	
15	dep-3					4	23		27	
16	dep-4				23				27	4
17	dep-5			25			2		27	
18										
19		**Total Cost**	24.38							

Figure 2.3: The transport example spreadsheet

Figure 2.4: The **AIMMS-Execute** toolbar

When you run this command, Excel will randomize the data in the unit transport cost area of the spreadsheet, and successively call AIMMS to solve the optimal transport for the given demand, supply and unit transport cost, and store these values in the transport area of the sheet.

Interface setup

To discover how this interface with AIMMS is accomplished, push the AIMMS icon on the **AIMMS** toolbar of Figure 2.4. This will pop up the **AIMMS Interface Setup** dialog box illustrated in Figure 2.5. Through three tabs on this dialog box, you can specify

- which AIMMS project the Excel sheet should connect to,
- which execution sequences are defined, and
- whether the setup dialog box should be password-protected.

Execution sequences

Through the **Execution Sequences** tab of the **AIMMS Interface Setup** dialog box, you can define one or more *execution sequences*, which each specify a number of *actions* with respect to the selected AIMMS project. Because execution sequences can be made visible under the **Execute** menu of the **AIMMS** toolbar, they form the main interface for end-users of your spreadsheet to perform AIMMS-related tasks.

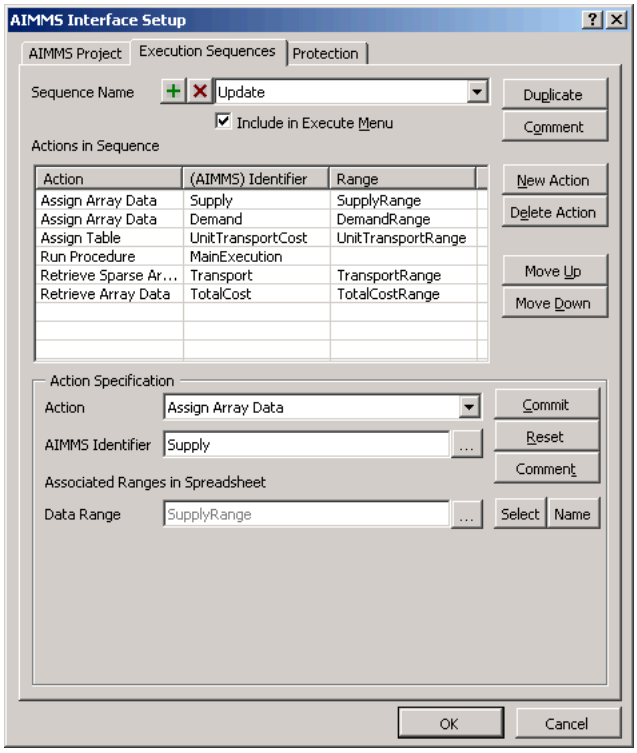


Figure 2.5: The **AIMMS Interface Setup** dialog box

Sequence actions

In the **Execution Sequences** tab you can add new actions to the list of actions associated with an execution sequence, modify or delete existing actions, supply comments on the execution sequences and actions, and change the order of execution of the specified actions. When adding an action, you choose from a number of predefined actions to perform various tasks that are relevant when interfacing with an AIMMS project, such as:

- transfer data from Excel to AIMMS and vice versa, in various formats,
- run procedures in the AIMMS model, or
- run macros in the spreadsheet.

`Update` sequence explained

The **AIMMS Interface Setup** dialog box of Figure 2.5 illustrates the definition of an execution sequence called `Update`. When this execution sequence is executed, the AIMMS Excel add-in executes a number of actions to:

- transfer the data in the named Excel range `SupplyRange` to the AIMMS identifier `Supply`,
- transfer the data in the named Excel range `DemandRange` to the AIMMS identifier `Demand`,
- transfer the data in the named Excel range `UnitTransportRange` to the AIMMS identifier `UnitTransportCost`,

- run the AIMMS procedure `MainExecution`,
- retrieve the data of the AIMMS identifier `Transport` and store it in the named Excel range `TransportRange`, and
- retrieve the data of the scalar AIMMS identifier `TransportCost` (the objective value) and store it in the named Excel range `TransportCostRange`.

Thus, the `Update` sequence performs the complete exchange of data with the transport model in AIMMS necessary to retrieve a new solution after an update of the input data.

Other sequences

In addition to the `Update` sequence, the **AIMMS Interface Setup** dialog box defines two additional (predefined) sequences `Initialization` and `Main` (see also Chapter 4). Of all defined sequences, the `Initialization` sequence is hidden in the **Execute** menu in the **AIMMS** toolbar.

`Main` and `Initialization` sequence

The predefined sequence `Initialization` is called once during the start-up of every session of the associated AIMMS project. In this example it is used to initialize the sets `Depots` and `Customers` in the AIMMS model, as illustrated in Figure 2.6.

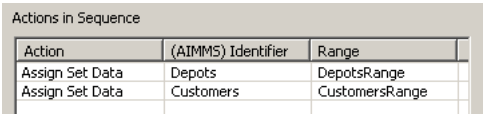


Figure 2.6: The **Initialization** execution sequence of the transport example

Main

The `Main` execution sequence of the transport model example contains two actions (see figure 2.7). The first action is a **Run Macro** action that runs an Excel macro which assigns random numbers to the **Unit Transport Range** of the sheet (initially range C4:G8). The second action is a call to the execution sequence `Update` described above.

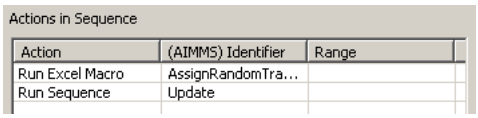


Figure 2.7: The **Main** execution sequence of the transport example

Separation of data and algorithms

The example uses a strict separation of data and algorithms. All data for the model is entered (or randomly assigned, in the case of the `Unit Cost` data) in the spreadsheet. The whole model needed to solve the problem is defined in AIMMS. The AIMMS Excel add-in is used to first transfer the data from Excel to AIMMS, solve the model and finally retrieve the AIMMS variables that have been calculated. Such a strict division of data and algorithms is highly advisable, as already pointed out in the introduction to this chapter.

Part II

AIMMS Excel Add-In Reference

Chapter 3

The AIMMS Excel Add-In Interface

Showing the 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.

Three tabs

The **AIMMS Interface Setup** dialog box consists of 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

Function of the tab

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

Project file

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. The Excel Add-In will automatically choose the AIMMS version to run with by inspecting this project file.

Config directory

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.

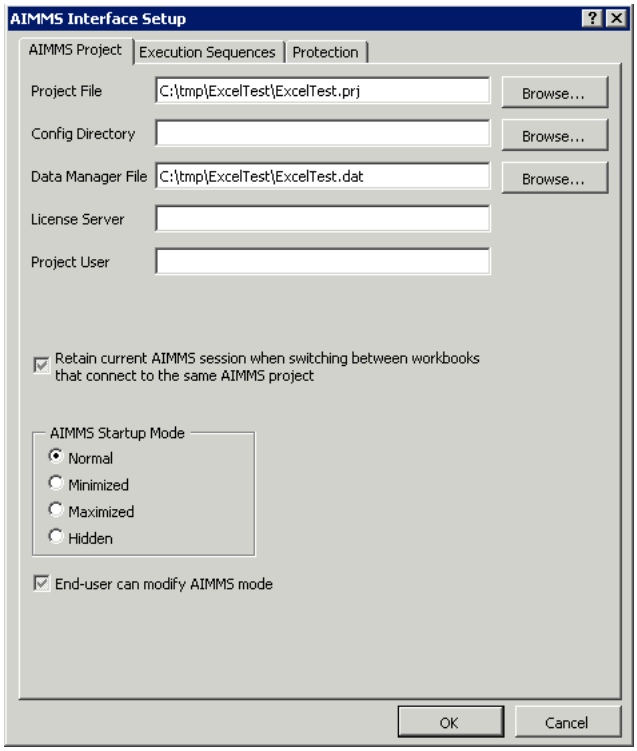


Figure 3.1: The AIMMS Project tab

Data manager file

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.

License server

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*.

Project user

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.

Retain current session

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.

Startup mode

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

- Normal
- Minimized
- Maximized
- Hidden

Modify AIMMS mode

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.

3.2 The Protection tab

Function of the 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.

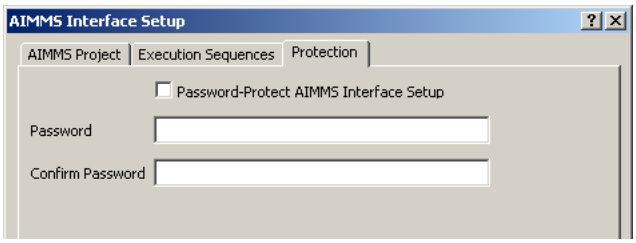


Figure 3.2: The **Protection** tab

Password-protect

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

Password

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.

Confirm

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

Password prompted

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.

3.3 The Execution Sequences tab

Function of the 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.

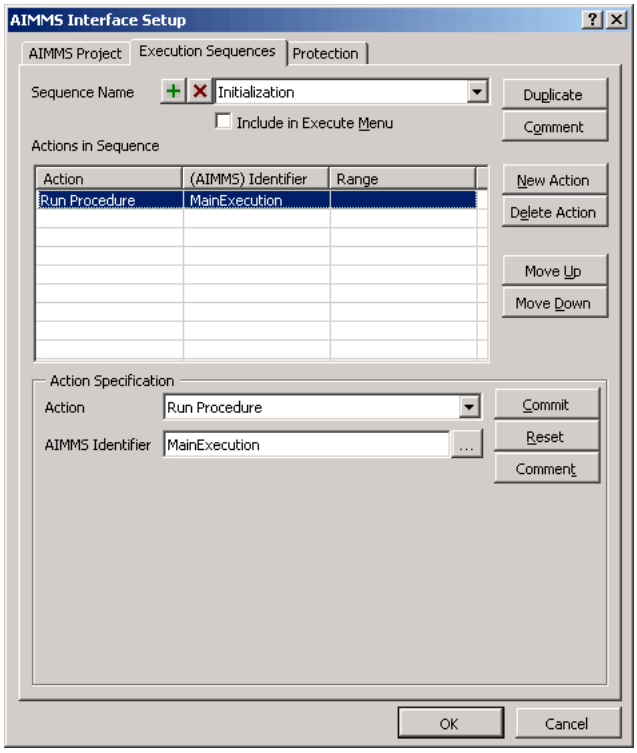


Figure 3.3: The **Execution Sequences** tab

What is an action?

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:

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.

Easily manageable

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.

What is an execution sequence?

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.

Sequence name

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.

Execute menu

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

Actions in sequence

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.

New and delete

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

Action specification

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.

Comments

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.

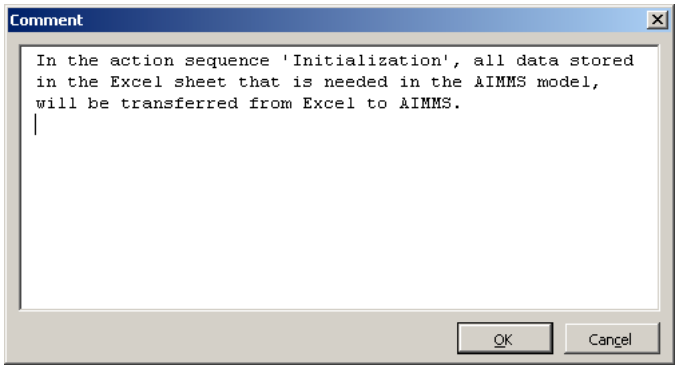


Figure 3.4: The **Comment** dialog

Action parameters

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).

All possible action parameters

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

- **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**.

Action parameter dialogs

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.

Required fields only

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.

Commit or reset

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.

Actions explained

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.

3.3.1 The action parameter dialogs

Introduction

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 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.

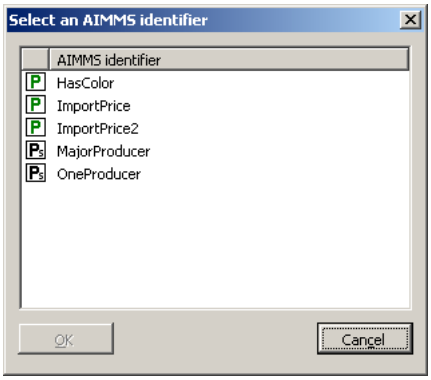


Figure 3.5: The **AIMMS identifier** dialog

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.

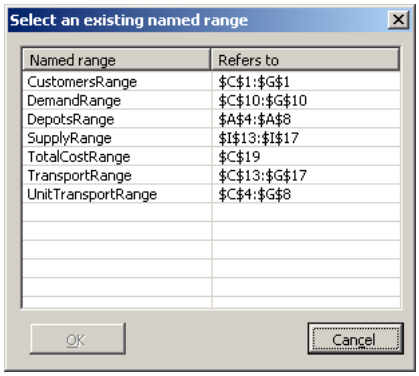


Figure 3.6: The named range dialog

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.

Odd selecting behavior with multiply named ranges

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.

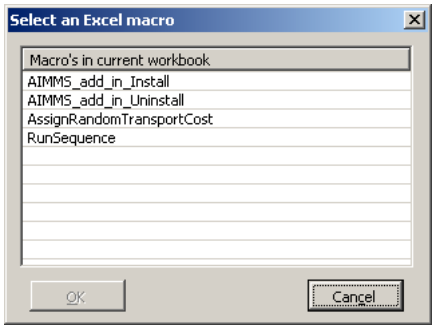


Figure 3.7: The Excel macro dialog

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

```
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

The action

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:

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

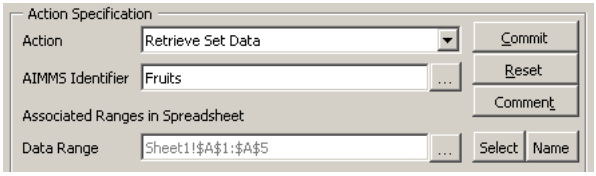


Figure 3.8: The action **Retrieve Set Data**

Example

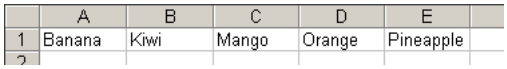
By specifying the action as in figure 3.8, all set elements are retrieved in the cells A1:A5 of sheet 1 of the Excel spreadsheet. Figure 3.9 shows the result after executing the action.

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

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

Remarks

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.



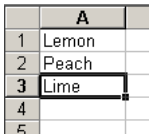
	A	B	C	D	E
1	Banana	Kiwi	Mango	Orange	Pineapple

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

3.3.3 Assign Set Data

The action

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**.



	A
1	Lemon
2	Peach
3	Lime
4	

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

Example

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

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

in AIMMS.

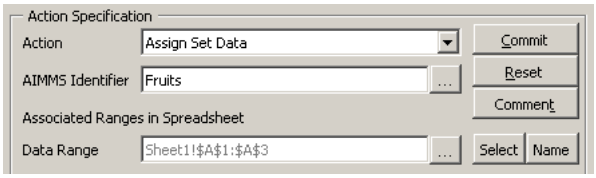


Figure 3.12: The action **Assign Set Data**

Remark

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.

3.3.4 Retrieve Array Data

The action

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**.

Consider an AIMMS project containing the following sets: *Example*

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

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

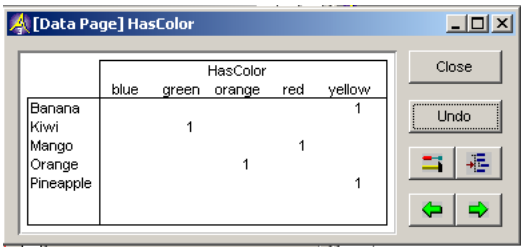


Figure 3.13: The data of the AIMMS parameter `HasColor`

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.

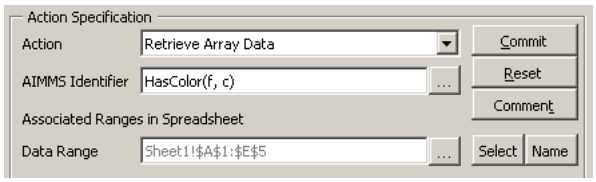


Figure 3.14: The action **Retrieve Array Data**

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**

No indices required

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')`.

3.3.5 Assign Array Data

The action

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.

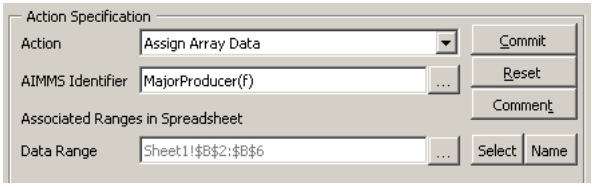


Figure 3.16: The action **Assign Array Data**

Example

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.

	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

The action

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.

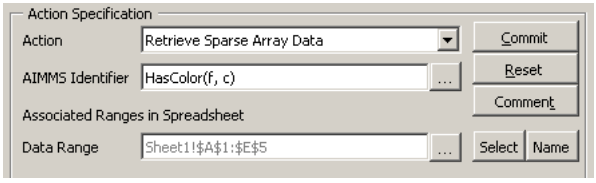


Figure 3.18: The action **Retrieve Sparse Array Data**

Example

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.

	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

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.

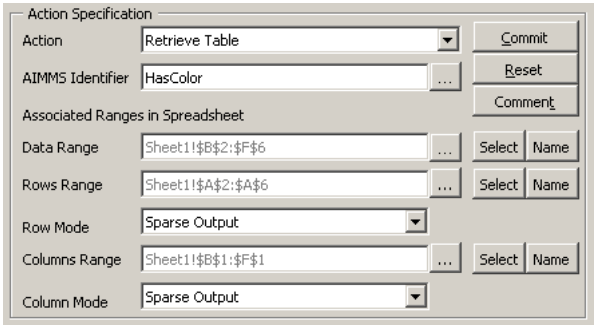


Figure 3.20: An example of the action **Retrieve Table**

Tabular data

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 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

Example

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.

A complete table

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 more advanced example

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

```
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.

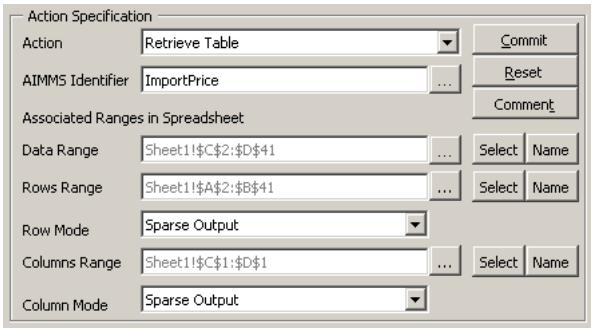


Figure 3.22: An advanced example of the action **Retrieve Table**

Remarks

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).

	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

Figure 3.23: The result of the action of figure

Extra zeroes

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.

Displaying of row- and column values

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 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 $r + c$ 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.

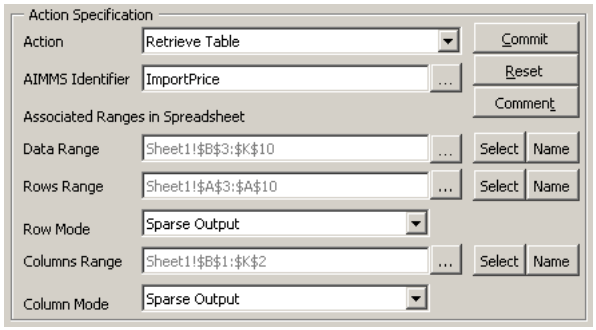


Figure 3.24: Action to achieve an alternative row- and column value display

More column values

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.

	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

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**.

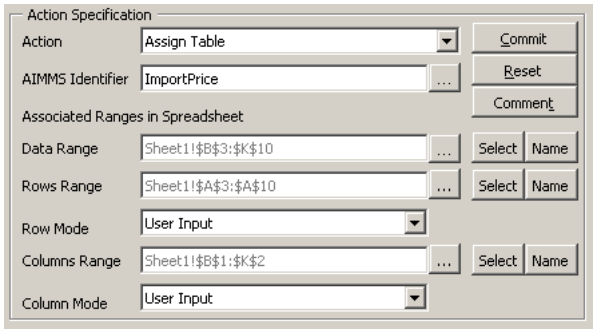


Figure 3.28: An **Assign Table** action

Example

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.

Remark

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*.

3.3.10 Empty Identifier

The action

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**.

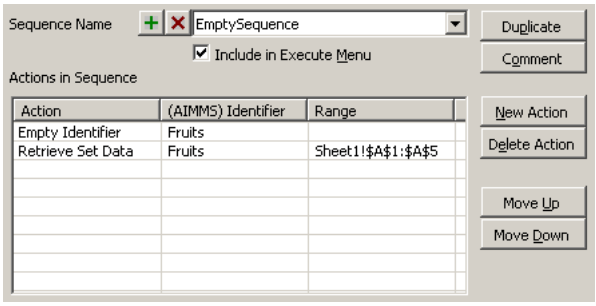


Figure 3.29: A sequence containing an **Empty** action

Example

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

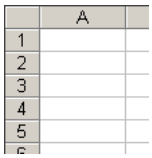


Figure 3.30: The result of the action sequence in figure

3.3.11 Update Identifier

The action

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.

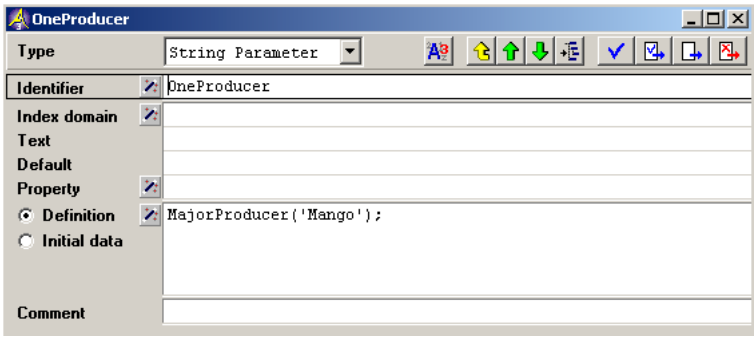


Figure 3.31: The definition of the `OneProducer` identifier

Example

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

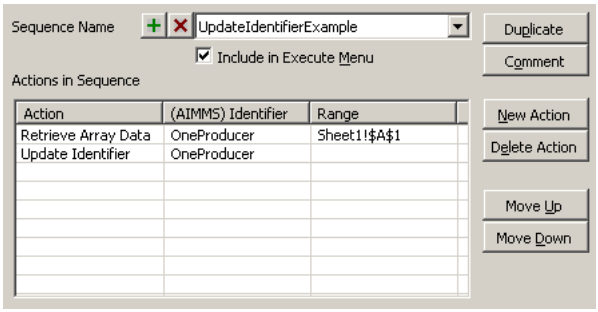


Figure 3.32: An action sequence containing an **Update** action

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

The action

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**.

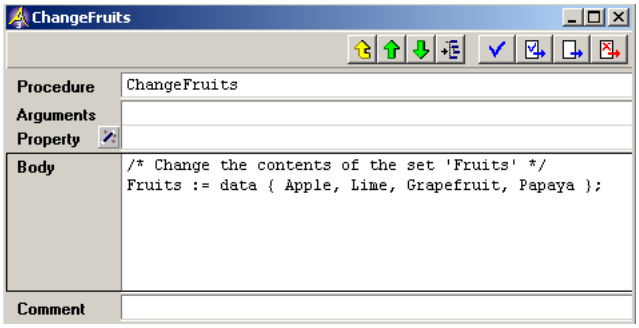


Figure 3.33: The definition of procedure `ChangeFruits`

Example

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.

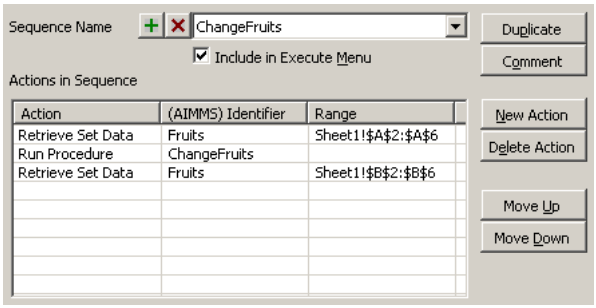


Figure 3.34: A sequence including the **Run Procedure** action

Remark

Notice the headers above the two columns in the output. Those headers were 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

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.

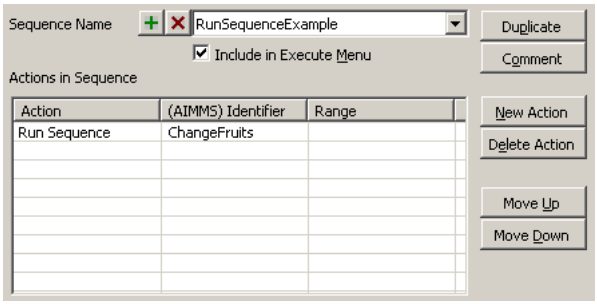


Figure 3.36: A sequence that executes another sequence

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*

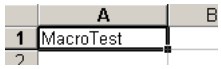


Figure 3.39: The result of **Run Excel Macro**

Chapter 4

Using the AIMMS Excel Add-In

This chapter

This chapter explains in more detail how the AIMMS Excel add-in can be used. You will find the details about the default sequences and the storage of the interface setup data, as well as the methods available to run one or more execution sequences.

4.1 Interface setup defaults and storage

Default sequences

When you start using the AIMMS Excel add-in with a new spreadsheet, you will be supplied with two default execution sequences, called

- `Initialization`, and
- `Main`.

The **AIMMS Interface Setup** dialog box will never allow you to delete these two sequences.

The `Initialization` sequence

The `Initialization` sequence is automatically called by the AIMMS Excel add-in directly after the AIMMS project has started. You can use this sequence, for instance, to initialize data in the AIMMS model that does not change throughout the session.

The `Main` sequence

The `Main` sequence is not called automatically by AIMMS, but can serve as a main execution entry in the **Execute** menu of the **AIMMS** menu or toolbar. To indicate this status, the **Main** sequence will always have an associated icon in the **Execute** menu, as illustrated in Figure 4.1.

Figure 4.1: The **AIMMS-Execute** toolbar

Not mandatory

Although you cannot delete the default sequences, you have the freedom to ignore them and remove them from the **Execute** menu, by unchecking the **Include in Execute Menu** checkbox on the **Execution Sequences** tab of the **AIMMS Interface Setup** dialog box.

Setup data storage

The AIMMS Excel add-in will store all interface data that you entered in the **AIMMS Interface Setup** dialog box in a hidden sheet in your workbook, called __AIMMS_SETUP__. In order to prevent the user from entering erroneous data on this sheet, which could cause the interface to stop functioning properly, the sheet cannot be made visible.

4.2 Running execution sequences

The execute menu

When you have defined some execution sequences using the **AIMMS Interface Setup** dialog and have checked the **Include in Execute Menu** checkbox for them, those execution sequences appear in the add-in provided **Execute** menu. Selecting an execution sequence from this menu will run it.

From Visual Basic

Another way to run an execution sequence is through the subroutine

```
ExecuteAimmsSequence(ByVal SequenceName As String)
```

which is exported by the AIMMS Excel add-in. You can call it from any Excel Macro that you've written. You only have to pass the name of the sequence defined in the **AIMMS Interface Setup** dialog box as an argument to this subroutine, as can be seen from the declaration. This way of running sequences offers an even more flexible way to integrate AIMMS projects into your Excel spreadsheets.

Include reference

To call the ExecuteAimmsSequence subroutine directly from within your Visual Basic code, you must include a reference to the AIMMS Excel add-in Aimms.xla to your spreadsheet through the **Tools-References** menu in the Visual Basic Editor. However, this is not strictly necessary, as you can also use the Run method to call ExecuteAimmsSequence directly from Aimms.xla, as illustrated below.

```
Run "Aimms.xla!ExecuteAimmsSequence", "Main"
```