

---

## **AIMMS Language Reference - Reading and Writing Spreadsheet Data**

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](http://www.aimms.com) or order your hard-copy at [www.lulu.com/aimms](http://www.lulu.com/aimms).

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](mailto:info@aimms.com)  
WWW: [www.aimms.com](http://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.  $\TeX$ ,  $\LaTeX$ , and  $\AMS-\LaTeX$  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  $\LaTeX$  and the LUCIDA font family.

## Chapter 27

# Reading and Writing Spreadsheet Data

Many users of AIMMS optimization models have input data for their models available in Excel workbooks, or want to export the results of the model back to an Excel workbook for further processing. While it is possible to exchange data with Excel through the READ and WRITE statements using the ODBC or OLE DB database connectivity interfaces (see Chapter 25), the Excel ODBC and OLE DB drivers have many limitations. Essential internal SQL statements like UPDATE and DELETE are not supported by these interfaces, effectively making the READ statement the only reliable AIMMS statement which might be used in this setup. For this reason, we strongly recommend you not to use this setup, but to consider to use the spreadsheet functions, described in this chapter, instead. Please note that there are no ODBC/OLE DB drivers available for OpenOffice Calc, so the remarks above apply to the Excel case only.

*Treating Excel as a database*

On the other hand, *from within Excel*, it is possible to exchange data with an AIMMS model in a variety of list and tabular formats using the Excel add-in provided with AIMMS. The Excel add-in is described in full detail in the *Excel Add-In User's Guide*.

*The Excel Add-In*

From AIMMS version 3.12 FR1 on, it is also possible to communicate with OpenOffice Calc workbooks from within the AIMMS model (there is no equivalent of the Excel add-in for OpenOffice Calc). The function library is the same as used for the communication with Excel from within the AIMMS model. To use the functions with OpenOffice Calc workbooks instead of Excel workbooks, simply use the extension `.ods` in the `WorkbookName` argument of the functions.

*The OpenOffice Calc function library*

This chapter provides a brief description of an AIMMS function library that allows you programmatic access, *from within your model*, to the extensive data exchange capabilities provided by the Excel add-in.

*This chapter*

---

### 27.1 An example

To illustrate the functionality of the Excel add-in, the AIMMS distribution contains an example, which provides a simple Excel workbook that illustrates the use of the Excel add-in. In this example workbook, all the input and output

*The Excel transport example...*

data of a transport model in AIMMS is retained in the workbook and exchanged with AIMMS using the data exchange functionality provided by the Excel add-in. You can find the example in the *Examples* directory of your AIMMS installation.

In this section, you will learn how the same data exchange could be accomplished from within your model using the spreadsheet function library of AIMMS. The source code illustrated in this section is contained in the AIMMS model accompanying the Excel Link example workbook. Thus, if you run this model in a stand-alone way from within AIMMS, the Excel Link example also serves as an example of the spreadsheet function library.

*... started from within AIMMS*

The input data of the transport model consists of:

- a set Depots with index *d*,
- a set Customers with index *c*,
- a parameter Supply(*d*),
- a parameter Demand(*c*), and
- a parameter UnitTransportCost(*d*,*c*).

*Retrieving the input data*

Using the spreadsheet function library, the following function calls retrieve all input data from the Excel workbook whose name is stored in the string parameter `WorkbookName`.

```
Spreadsheet::SetActiveSheet( WorkbookName, "Transport Model" );
Spreadsheet::RetrieveSet( WorkbookName, Depots, "DepotsRange" );
Spreadsheet::RetrieveSet( WorkbookName, Customers, "CustomersRange" );
Spreadsheet::RetrieveParameter( WorkbookName, Supply, "SupplyRange" );
Spreadsheet::RetrieveParameter( WorkbookName, Demand, "DemandRange" );
Spreadsheet::RetrieveTable( WorkbookName, UnitTransportCost,
    "UnitTransportRange", "DepotsRange", "CustomersRange" );
```

This sequence of function calls, with the exception of the first call, is the direct counterpart of the sequence of actions in the Excel workbook example used to pass the model data to the AIMMS model.

By calling the function `Spreadsheet::SetActiveSheet`, you indicate to AIMMS that all following calls operate on a single sheet, allowing you to omit the sheet name as an optional argument in subsequent calls. Through the functions

*Explained*

- `Spreadsheet::RetrieveSet`,
- `Spreadsheet::RetrieveParameter`, and
- `Spreadsheet::RetrieveTable`,

you indicate to AIMMS that the corresponding set and parameter data must be obtained from the specified named Excel ranges. The functionality of these functions is exactly the same as the functionality of the corresponding actions in the Excel add-in. Note that ranges can also be described using the standard A1 and R1C1 styles of Excel.

The input data of the transport model consists of:

- a variable `Transport(d,c)`, and
- a variable `TotalCost` containing the objective value of the optimization model.

These values can be stored in the given workbook using the following function calls.

```
Spreadsheet::SetActiveSheet( WorkbookName, "Transport Model" );
Spreadsheet::AssignParameter( WorkbookName, Transport, "TransportRange", sparse: 1 );
Spreadsheet::AssignValue( WorkbookName, TotalCost, "TotalCostRange" );
```

*Writing back the solution*

Again, these functions provide exactly the same functionality as the corresponding actions in the Excel add-in, and the sequence of function calls corresponds in a one-to-one fashion to the sequence of actions in the Excel workbook example to retrieve the solution back from AIMMS. Through the optional `sparse` argument of `Spreadsheet::AssignParameter` you can indicate whether zero values should be passed as 0.0 or as a blank.

*Explained*

The following function call illustrates how a macro contained in a workbook can be run from within your AIMMS model.

*Running a macro*

```
Spreadsheet::RunMacro( WorkbookName, "AssignRandomTransportCost" );
```

In the Excel Link example this macro is used to randomize the values of the range holding the values of `UnitTransportCost`. After re-retrieving the input data again and solving the model, this may result in a different optimal solution to the transport model.

---

## 27.2 Function overview

In this section you will find an overview of all the functions provided by the spreadsheet function library. The function library contains both

*Function overview*

- control functions, and
- data exchange functions.

All functions are described in full detail in the Function Reference.

From AIMMS 3.12 Feature Release 1 on, the first part of the function names has changed from `Excel...` to the more general `Spreadsheet::...`, to reflect the fact that the functions are not exclusively used to communicate with Excel anymore. When you want to work with an OpenOffice Calc workbook, the `WorkbookName` argument of the functions should end in `.ods` (which is the extension of Calc workbooks). Any other ending of this argument will result in AIMMS operating on an Excel workbook.

*Function naming*

The control functions listed in Table 27.1 allow you to perform actions such as opening and closing workbooks and worksheets, copying and printing ranges, and running macros contained in the workbook.

*Control  
functions*

The control functions listed in Table 27.1 do not have a direct counterpart in the AIMMS Excel add-in. They represent a subset of common spreadsheet commands, which may be convenient when reading and writing data to an Excel or OpenOffice Calc workbook.

*Not in Excel  
add-in*

Procedure	Description
Spreadsheet::CreateWorkbook	Creates a workbook
Spreadsheet::SaveWorkbook	Saves an opened workbook
Spreadsheet::CloseWorkbook	Closes an opened workbook
Spreadsheet::AddNewSheet	Adds a new sheet to a workbook
Spreadsheet::DeleteSheet	Delete a sheet from a workbook
Spreadsheet::SetActiveSheet	Sets the currently active sheet
Spreadsheet::Print	Prints a range from a workbook
Spreadsheet::ClearRange	Clears the specified range
Spreadsheet::CopyRange	Copies a source into a destination range
Spreadsheet::SetVisibility	Changes the visibility of a workbook
Spreadsheet::SetUpdateLinksBehavior	Sets the behavior w.r.t. linked workbooks
Spreadsheet::ColumnName	Returns the name of a numbered column
Spreadsheet::ColumnNumber	Returns the number of a named column
Spreadsheet::RunMacro	Runs the specified macro

Figure 27.1: Spreadsheet control functions

The functions listed in table 27.2 can be used to exchange set data, scalar values, one- and two-dimensional identifiers, and general multi-dimensional identifiers with tabular ranges in an Excel or Calc sheet. Each of these functions corresponds to an associated action in the Excel add-in.

*Data exchange  
functions*

<b>Function</b>	<b>Description</b>
Spreadsheet::AssignSet Spreadsheet::RetrieveSet	Assigns set elements to specified range Fills set with elements from specified range
Spreadsheet::AssignValue Spreadsheet::RetrieveValue	Assigns scalar value to specified range Fills scalar parameter from specified range
Spreadsheet::AssignParameter Spreadsheet::RetrieveParameter	Assigns 1- or 2-dimensional parameter to range Fills 1- or 2-dimensional parameter from range
Spreadsheet::AssignTable Spreadsheet::RetrieveTable	Assigns multi-dimensional parameter to range Fills multi-dimensional parameter from range

Figure 27.2: Spreadsheet data exchange functions