
**AIMMS COM Object User's Guide and Reference - Introduction to the
AIMMS COM Object**

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. $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, and $\text{A}_{\text{M}}\text{S}_{\text{L}}\text{A}_{\text{T}}\text{E}_{\text{X}}$ 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 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and the LUCIDA font family.

Part II

The AIMMS COM Object Reference

Chapter 2

Introduction to the AIMMS COM Object

2.1 The AIMMS COM object

The AIMMS COM object defines a number of COM interfaces, which allow you to easily integrate AIMMS projects into languages such as Visual Basic, Visual Basic .NET, or any of the other .NET languages, and scripting languages such as VBScript.

*The AIMMS
COM object*

The AIMMS COM object defines four COM interfaces:

*AIMMS COM
interface*

- `Aimms.Project`,
- `Aimms.Identifier`,
- `Aimms.Set`, and
- `Aimms.Procedure`

Of these four interfaces, only the `Aimms.Project` interface is registered as a real COM object, in the sense that you can create it via a call to a standard object creator function (for example through the `New` operator, or the `CreateObject` procedure in Visual Basic). The other three AIMMS COM interfaces can only be created via methods in the `Aimms.Project` interface. This may seem odd at first, but these objects correspond directly to model identifiers, and thus can only exist within the context of an `Aimms.Project` object.

The AIMMS COM object has been made version dependent as of AIMMS 3.6. The interpretation of this remark is as follows.

*Version
dependent
interfaces*

- If you are using a so-called ProgID to creating an `Aimms.Project` interface of the AIMMS COM object, the following rules apply:
 - the ProgID “`Aimms.Project.3.6`” will open the AIMMS 3.6 COM object,
 - the ProgID “`Aimms.Project.1`” will open an AIMMS COM object corresponding to AIMMS 3.5 or earlier. Which AIMMS COM version depends on which AIMMS version ≤ 3.5 has been installed latest.
 - the ProgID “`Aimms.Project`” will open the AIMMS COM object associated with the latest installed AIMMS version.

- If you are creating an `Aimms.Project` interface of the AIMMS COM object via its associated CLSID (i.e. a GUID associated with the `Aimms.Project` object) or as through a reference in a VB or .NET application, the following rules apply:
 - if you are referencing the CLSID of AIMMS 3.6 (directly or indirectly via a reference to the AIMMS 3.6 COM object), this will always lead to the use of the AIMMS 3.6 COM object
 - if you are referencing the CLSID of AIMMS 3.5 or lower, the COM object of the latest installed AIMMS version ≤ 3.5 will be used

The benefits for you as a developer are clear. If you want to enforce the use of the AIMMS 3.6 COM object, you can use the “`Aimms.Project.3.6`” ProgID or the associated CLSID. If the AIMMS COM of any AIMMS version suffices, just use the “`Aimms.Project`” ProgID to create an `Aimms.Project` interface.

The use of the AIMMS COM object is rather straightforward, as illustrated in the example in Chapter 1. All properties and methods corresponding to the four AIMMS COM interfaces are discussed in full detail in the subsequent chapters.

Example and reference

2.2 The Variant data type

Many of the properties and methods of the AIMMS COM interface make use of the Variant data type. This is a special data type that is commonly used in COM interfaces and can hold various other data types. The AIMMS COM interface uses this type to generalize the methods and functions and to avoid that methods and properties have different versions based on the data type that is used. The Variant type is used to:

Variant

- pass values from numerical identifiers, string parameters and element parameters, and
- to pass set elements either by an integer number or by string.

AIMMS supports three main data types for its identifiers: numerical, string and element valued. In the AIMMS COM interface this means that if you want to refer to the value of an identifier this can either be a double, a string or an integer. Therefore, instead of providing three separate methods for each type, the methods concerning identifier values use the Variant type. The context of a specific call then determines whether this Variant contains a double, a string or an integer. Although a method uses a Variant type, you can still pass the required double, string or integer directly. For example, to assign new strings to an indexed string parameter, you can pass an array of Variants containing strings, or simply pass an array of strings. At runtime, the method itself checks whether the given argument can be converted to the required data type.

Identifier values

To communicate set elements back and forth, the AIMMS COM offers three different modes *Set elements*

- using unique integer element numbers,
- using integer ordinal numbers, or
- using element names (i.e. strings).

Therefore, in the methods where elements or tuples of elements are passed, the corresponding argument is usually of type Variant, and special properties determine whether these Variants contain integers or strings. Similar as with the identifier values you may circumvent the Variant type and directly pass a string or integer.

2.3 Using the AIMMS COM interfaces

You can access the AIMMS COM interface from within any language or program that supports COM technology. The two most important and widely used languages are perhaps Visual Basic or any of the .NET languages, because these languages and the derived scripting languages are used frequently in spreadsheets and HTML pages.

In this document we describe how the interface looks from a Visual Basic point of view. You can also use the AIMMS COM object from any of the .NET languages in a manner very similar to that use from within Visual Basic. *Visual Basic and .NET languages*

If you are using another language (such as C/C++), you should refer to the documentation of that language to learn how to access COM objects. When using Visual C++, the appropriate way is to use the `#import` directive, which will create a proxy class which you can use to create and access the AIMMS COM object. This is illustrated in the Calling AIMMSCOM example. *Other languages*