
**AIMMS COM Object User's Guide and Reference - AIMMS.Procedure
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. \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 6

The Aimms.Procedure Object

The Procedure object allows you run procedures within the AIMMS project. The Procedure object offers a small number of properties and methods to retrieve information about the arguments and to actually run the procedure. If your procedure does not require arguments and if you do not have to run the procedure multiple times, then you do not have to use the Procedure object. Instead, you can use the Run method of the Project object directly.

Because a procedure only exists within the context of an AIMMS project, you cannot create a Procedure object directly using a standard COM object creator function. Instead, you must create a Procedure object via a specific call to the Aimms.Project object, namely `Project.GetProcedure`.

Object creation

The following example shows a small Visual Basic program, that opens an AIMMS project, creates a Procedure object, and runs that procedure.

Example

```
Dim MyProject As Aimms.Project
Dim MainExec As Aimms.Procedure

'create project object, and open the project file
set MyProject = New Aimms.Project
MyProject.ProjectOpen "C:\Examples\Transport.prj"

'create procedure object for the AIMMS procedure 'MainExecution'
set MainExec = MyProject.GetProcedure( "MainExecution" )

'run it
MainExec.Run
```

The Procedure object exposes the following methods and properties:

Methods and Properties

- `Procedure.Run`
- `Procedure.NumberOfArguments`
- `Procedure.ArgumentInOutType`
- `Procedure.ArgumentStorageType`

Procedure.Run

With the method `Run` you can execute a procedure within the AIMMS model. If the procedure expects arguments, these arguments can be filled in too.

Synopsis:

```
Long Run(  
    [var_args]           ! (optional) variable number of arguments  
)
```

Arguments:

var_args (optional)

A variable number of arguments, corresponding with the arguments of the procedure in the AIMMS model.

Return value:

The method `Run` returns the value that is returned from the AIMMS procedure. The return value of a procedure in AIMMS is optional, and is usually omitted. In that case, the default return value 0 is used.

Example:

The following two pieces of code give the same result:

```
'Running a procedure using the Procedure object  
Dim proc As Aimms.Procedure  
Set proc = MyProject.GetProcedure( "MainExecution" )  
proc.Run
```

See also:

The methods `Project.GetProcedure` `Project.Run`

Procedure.NumberOfArguments

This property gives the number of arguments the procedure expects according to its declaration in AIMMS. *Property*

Synopsis:

Int NumberOfArguments

Remarks:

The property `NumberOfArguments` is read-only.

Procedure.ArgumentInOutType

This property indicates whether the n -th argument is an input argument, an output argument or both input and output. *Property*

Synopsis:

```
Integer ArgumentInOutType(  
    n          ! (input) Integer  
)
```

Arguments:

n
The sequence number of the argument. The first argument must be referred to using $n = 1$.

Remarks:

The property `ArgumentInOutType` is integer valued and can have any of the following defined values:

- `ARG_INPUT` = 1
- `ARG_OUTPUT` = 2
- `ARG_INOUT` = 3

If an argument is of type `ARG_OUTPUT` or `ARG_INOUT`, you must make sure that you pass your argument *by reference*.

Procedure.ArgumentStorageType

This property indicates the data type of the n -th argument of the AIMMS procedure. *Property*

Synopsis:

```
Integer ArgumentStorageType(  
    n          ! (input) Integer  
)
```

Arguments:

n
The sequence number of the argument. The first argument must be referred to using $n = 1$.

Remarks:

The property `ArgumentStorageType` is integer valued and can have any of the following defined values:

- `STORAGE_HANDLE = 0`
- `STORAGE_DOUBLE = 1`
- `STORAGE_INTEGER = 2`
- `STORAGE_BINARY = 3`
- `STORAGE_STRING = 4`

If the storage type is `STORAGE_HANDLE`, you must pass a reference to an `Aimms.Identifier` object.