
AIMMS Function Reference - Miscellaneous Functions

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

Miscellaneous Functions

AIMMS supports the following miscellaneous functions, that cannot easily be categorized in other sections:

- DebuggerBreakPoint
- Delay
- Execute
- ExitAimms
- GeoFindCoordinates
- IdentifierGetUsedInformation
- IdentifierMemory
- IdentifierMemoryStatistics
- IsRunningAsViewer
- ListingFileCopy
- ListingFileDelete
- MemoryInUse
- MemoryStatistics
- OpenDocument
- ProfilerStart
- ProfilerPause
- ProfilerContinue
- ProfilerRestart
- ScheduleAt
- SessionArgument
- SessionHasVisibleGUI
- ShowHelpTopic
- ShowMessageWindow
- ShowProgressWindow
- TestInternetConnection

DebuggerBreakPoint

The procedure `DebuggerBreakPoint` breaks execution and activates the debugger when needed.

```
DebuggerBreakPoint(  
  [only_if_active]      ! (optional, default 0) scalar binary expression  
)
```

Arguments:

only_if_active

When this argument equals 1, execution is only stopped when the debugger is active. If this argument equals 0 the execution is always stopped and the debugger is activated if necessary.

Remarks:

- The debugger and profiler are exclusive. When the profiler is active, this procedure has no effect.
- This procedure has no effect in end-user mode because the debugger is not available in end-user mode.

Delay

With the procedure `Delay` you can block the execution of your model for the indicated delay time. You can use this procedure, for instance, when you want to display intermediate results on a page using the procedure `PageRefreshAll`.

```
Delay(  
    delaytime          ! (input) scalar expression  
)
```

Arguments:

delaytime

The number of seconds that the execution should be blocked.

See also:

The procedure [PageRefreshAll](#).

Execute

With the Execute procedure you can start another application.

```
Execute(  
    executable,      ! (input) scalar string expression  
    [commandline,]  ! (optional) scalar string expression  
    [workdir,]      ! (optional) scalar string expression  
    [wait,]         ! (optional) 0 or 1  
    [minimized]    ! (optional) 0 or 1  
)
```

Arguments:

executable

A string representing the name of the program that you want to execute.

commandline (optional)

A string representing the arguments that you want to pass to the program.

workdir (optional)

A string representing the directory where the program should start in. If omitted, then the current project directory is used.

wait (optional)

This argument indicates whether or not AIMMS will wait for the program to finish. The default value is 0 (not wait).

minimized (optional)

This argument indicates whether or not the program should run in a minimized state. The default is 0 (not minimized).

Remarks:

As a general rule, you should not wait for interactive windowed applications. Waiting for the termination of a program is necessary when the program does some form of external data processing which is required for the execution of your model.

See also:

The procedure [OpenDocument](#).

ExitAimms

With the procedure `ExitAimms` you can exit the current AIMMS session from within a procedure.

```
ExitAimms(  
    [interactive]      ! (optional) 0 or 1  
)
```

Arguments:

interactive (optional)

This optional argument is still present for compatibility, but does no longer have any effect. You should use `MainTermination` to specify whether or not AIMMS should display a confirmation dialog box before closing the current project.

Remarks:

The procedure does not immediately exit AIMMS, but it will try to exit as soon as the execution of the current procedure has finished. If existing, the `logoff` procedure and the procedure `MainTermination` will be executed as normal.

GeoFindCoordinates

The procedure `GeoFindCoordinates` can be used to find the latitude/longitude coordinates for a given address. The procedure uses the free [OpenStreetMap](#) (OSM) geocoding service. You are advised to carefully read the OSM geocoder [usage policy](#) before using this procedure in your application.

```
GeoFindCoordinates(
  address,          ! (input) scalar string expression
  latitude,        ! (output) scalar numerical parameter
  longitude,       ! (output) scalar numerical parameter
  email            ! (optional) scalar string parameter
)
```

Arguments:

address

A string representing the address for which the latitude and longitude coordinates have to be found.

latitude

A scalar numerical parameter that will contain the latitude coordinate of the specified address upon success.

longitude

A scalar numerical parameter that will contain the longitude coordinate of the specified address upon success.

email

An optional string representing the email address that the OSM organization will use to contact you in the event of problems (as mentioned in their [usage policy](#)).

Return value:

The procedure returns 1 on success, and 0 if the specified address could not be found. On failure, the pre-defined identifier `CurrentErrorMessage` will contain a proper error message.

Examples:

The following calls to the procedure `GeoFindCoordinates` return valid latitude and longitude coordinates

```
GeoFindCoordinates("NL", Latitude, Longitude);
GeoFindCoordinates("Haarlem, NL", Latitude, Longitude);
GeoFindCoordinates("2034 Haarlem, NL", Latitude, Longitude);
GeoFindCoordinates("Schipholweg, Haarlem, NL", Latitude, Longitude);
GeoFindCoordinates("1 Schipholweg, Haarlem, NL", Latitude, Longitude);
GeoFindCoordinates("1 Schipholweg, 2034 Haarlem, NL", Latitude, Longitude);
```

```
GeoFindCoordinates("US", Latitude, Longitude);
GeoFindCoordinates("Kirkland, WA, US", Latitude, Longitude);
GeoFindCoordinates("Lake Washington Boulevard NE, Kirkland, US", Latitude, Longitude);
GeoFindCoordinates("5400 Carillon Point, Lake Washington Boulevard NE, Kirkland, US",
    Latitude, Longitude);

GeoFindCoordinates("Singapore", Latitude, Longitude);
GeoFindCoordinates("Chulia Street, Singapore", Latitude, Longitude);
```

assumed that *Latitude* and *Longitude* are declared as numerical parameters in your model.

Remarks:

- With the introduction of AIMMS 3.9.5 and AIMMS 3.10 PR, this procedure has been disabled because Microsoft discontinued support to the Virtual Earth geocoder service that was used to locate the address. In AIMMS 3.11 FR2, the `GeoFindCoordinates` procedure was enabled again by using the OSM [geocoding service](#) instead.
- *'One of the hard things about geocoding is parsing addresses into something intelligible'* (see the OpenStreetMap [wiki](#) for details on address formats). As a result, you may need to slightly play around with the address format in order for the geocoder to correctly parse your address.
- To discourage *'bulk geocoding'* (see the OSM [usage policy](#) for more details), AIMMS inserts a small delay in case the time between two consecutive geocoding requests is smaller than a second.

IdentifierGetUsedInformation

With the procedure `IdentifierGetUsedInformation` you can obtain information on whether an identifier in the model is still referenced in either a page, a user menu or a case type/data category.

```
IdentifierGetUsedInformation(  
  identifier          ! (input) element parameter  
  isUsedInPages,     ! (output) scalar numerical identifier  
  isUsedInMenus,     ! (output) scalar numerical identifier  
  isUsedInDataCategories ! (output) scalar numerical identifier  
)
```

Arguments:

identifier

The identifier, given as element in the set `AllIdentifiers`, whose usage info you want to retrieve.

isUsedInPages

On return this value is set to 1 if the identifier is referenced in either a page, template or print page. It is set to 0 otherwise.

isUsedInMenus

On return this value is set to 1 if the identifier is referenced in a menu item or submenu of a user menu. It is set to 0 otherwise.

isUsedInDataCategories

On return this value is set to 1 if the identifier is referenced in either a data category or case type. It is set to 0 otherwise.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

The function only indicates whether the identifier is used in either of the three GUI areas. To figure out in which specific page, menu or data category the identifier is used you can use the drag-and-find feature of the IDE: if you drag an identifier from the Model Explorer, holding down both the Control and Shift key, and drop it on either the Page Manager, Template Manager, Menu Builder or Data Management Setup tree, all items that reference the identifier will be highlighted.

See also:

The procedure [PageGetUsedIdentifiers](#).

IdentifierMemory

With the function `IdentifierMemory` you can determine the total amount of memory occupied by the identifier.

```
IdentifierMemory(  
    Identifier,          ! (input) scalar element parameter  
    IncludePermutations ! (optional, default 1) scalar binary expression  
)
```

Arguments:

Identifier

An element expression in the set `AllIdentifiers` specifying the identifier for which the amount of occupied memory should be determined.

IncludePermutations

An 0-1 value indicating whether the amount of memory occupied by permutations of the identifier should also be included in the total memory determination.

Return value:

The function reports the sum of the memory occupied by the identifier, its suffices and the associated hidden identifiers (that are introduced as temporary identifiers by the AIMMS compiler/execution engine).

Remarks:

The return value of this function differs from the value reported in the 'Memory Usage' column of the **Identifier Cardinalities** dialog box because in the **Identifier Cardinalities** dialog box the value for hidden identifiers and suffices are reported separately.

IdentifierMemoryStatistics

With the procedure `IdentifierMemoryStatistics` you can obtain a report containing the statistics collected by AIMMS' memory manager for a single or multiple high dimensional identifiers.

```
IdentifierMemoryStatistics(
  IdentSet,          ! (input) a set of identifiers
  OutputFileName,   ! (input) scalar string expression
  AppendMode,       ! (optional, default 0) scalar numerical expression
  MarkerText        ! (optional) scalar string expression
  ShowLeaksOnly     ! (optional) scalar expression
  ShowTotals        ! (optional) scalar expression
  ShowSinceLastDump ! (optional) scalar expression
  ShowMemPeak       ! (optional) scalar expression
  ShowSmallBlockUsage ! (optional) scalar expression
  doAggregate       ! (optional, default 0) scalar expression
)
```

Arguments:

IdentSet

A subset of `AllIdentifiers` whose memory statistics are to be reported.

OutputFileName

A string expression holding the name of the file to which the statistics must be written.

AppendMode

An 0-1 value indicating whether the file must be overwritten or whether the statistics must be appended to an existing file.

MarkerText

A string printed at the top of the memory statistics report.

ShowLeaksOnly

A 0-1 value that is only used internally by Paragon Decision Technology. The value specified doesn't influence the memory statistics report.

ShowTotals

A 0-1 value indicating whether the report should include detailed information about the total memory use in AIMMS' own memory management system until the moment of calling `IdentifierMemoryStatistics`.

ShowSinceLastDump

A 0-1 value indicating whether the report should include basic and detailed information about the memory use in AIMMS' own memory management system since the previous call to `IdentifierMemoryStatistics`.

ShowMemPeak

A 0-1 value indicating whether the report should include detailed in-

formation about the memory use in AIMMS' own memory management system, when the memory consumption was at its peak level prior to calling `IdentifierMemoryStatistics`.

ShowSmallBlockUsage

A 0-1 value indicating whether the detailed information about the `MemoryStatistics` memory use in AIMMS' own memory management system is included at all in the memory statistics report. Setting this value to 0 results in a report with only the most basic statistical information about the memory use.

doAggregate

A 0-1 value (default 0) indicating whether a single aggregated report is to be presented or multiple individual reports.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The procedure prints a report of the statistics collected by AIMMS' memory manager since the last call to `IdentifierMemoryStatistics`.
- AIMMS will only collect memory statistics if the option `memory_statistics` is on.

IsRunningAsViewer

With the function `IsRunningAsViewer` you can detect whether the current AIMMS session is a Viewer session.

```
IsRunningAsViewer()
```

Return value:

The function returns 1 if the session is an AIMMS Viewer session.

Remarks:

The AIMMS Viewer offers limited functionality, as described in [Section 16.3](#). Using the function `IsRunningAsViewer` you can detect whether the current session is a Viewer session and adapt the execution of your model to adapt to the limited capabilities of the AIMMS Viewer version.

ListingFileCopy

With the procedure `ListingFileCopy` you can copy the current contents of the listing file to a given file.

```
ListingFileCopy(  
    toFileName,      ! (input) string expression  
    overwrite        ! (optional) default 1.  
)
```

Arguments:

toFileName

The file name of the file to which the contents of the listing file must be copied.

overwrite

if equal to 0 then do not overwrite an existing file, otherwise overwrite an existing file when needed.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The procedure [ListingFileDelete](#).

ListingFileDelete

The function `ListingFileDelete` deletes the current contents of the listing file associated with an AIMMS project.

`ListingFileDelete()`

Return value:

The function returns 1 on success, or 0 otherwise.

See also:

The function [ListingFileCopy](#).

MemoryInUse

With the function `MemoryInUse` you can obtain the current amount of memory in use as it is reported by the operating system.

```
MemoryInUse()
```

Return value:

This function returns the amount of memory in use in [Mb].

Remarks:

See also the functions `MemoryStatistics`, `IdentifierMemory`, `GMP::Instance::GetMemoryUsed`

MemoryStatistics

With the procedure `MemoryStatistics` you can obtain a report containing the statistics collected by AIMMS' memory manager.

```
MemoryStatistics(
  OutputFileName,    ! (input) scalar string expression
  AppendMode,        ! (optional, default 0) scalar numerical expression
  MarkerText,        ! (optional, default empty) scalar string expression
  ShowLeaksOnly,     ! (optional, default 0) scalar numerical expression
  ShowTotals,        ! (optional, default 1) scalar numerical expression
  ShowSinceLastDump, ! (optional, default 1) scalar numerical expression
  ShowMemPeak,       ! (optional, default 0) scalar numerical expression
  ShowSmallBlockUsage, ! (optional, default 0) scalar numerical expression
  GlobalOnly         ! (optional, default 0) scalar numerical expression
)
```

Arguments:

OutputFileName

A string expression holding the name of the file to which the statistics must be written color to modify.

AppendMode

An 0-1 value indicating whether the file must be overwritten or whether the statistics must be appended to an existing file.

MarkerText

A string printed at the top of the memory statistics report.

ShowLeaksOnly

A 0-1 value that is only used internally by Paragon Decision Technology. The value specified doesn't influence the memory statistics report.

ShowTotals

A 0-1 value indicating whether the report should include detailed information about the total memory use in AIMMS' own memory management system until the moment of calling `MemoryStatistics`.

ShowSinceLastDump

A 0-1 value indicating whether the report should include basic and detailed information about the memory use in AIMMS' own memory management system since the previous call to `MemoryStatistics`.

ShowMemPeak

A 0-1 value indicating whether the report should include detailed information about the memory use in AIMMS' own memory management system, when the memory consumption was at its peak level prior to calling `MemoryStatistics`.

ShowSmallBlockUsage

A 0-1 value indicating whether the detailed information about the memory use in AIMMS' own memory management system is included at all in the memory statistics report. Setting this value to 0 results in a report with only the most basic statistical information about the memory use.

GlobalOnly

A 0-1 value indicating whether only memory used by the global memory manager (i.e. the 'main' memory manager of AIMMS, as opposed to separate memory manager for individual higher-dimensional identifiers) is reported in the memory statistics file.

Return value:

The procedure prints a report of the statistics collected by AIMMS' memory manager since the last call to `MemoryStatistics`.

Remarks:

AIMMS will only collect memory statistics if the option `memory_statistics` is on.

OpenDocument

The procedure `OpenDocument` uses the current association of Windows to open documents, run programs, etc. Its procedureality is similar to that of the **Run** command in the **Start Menu** of Windows. You can use it, for instance, to display an HTML file using the default web browser, open a Word document, or initiate an e-mail session.

```
OpenDocument(  
    document          ! (input) string expression  
)
```

Arguments:

document

A string expression representing the document or program you want to open.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Examples:

```
OpenDocument( "http://www.aimms.com" );  
OpenDocument( "mailto:info@aimms.com" );  
OpenDocument( "anyfile.doc" );  
OpenDocument( "c:\\windows" );
```

See also:

The procedure [Execute](#).

ProfilerStart

The procedure `ProfilerStart` starts measuring the execution time of statements and definitions.

`ProfilerStart`

Remarks:

When the option `profiler_store_data` has been set to `On` profiling information is stored in the predefined identifier `ProfilerData`.

See also:

The procedures `ProfilerPause`, `ProfilerContinue` and `ProfilerRestart` and the predefined identifier `ProfilerData`.

ProfilerPause

The procedure `ProfilerPause` temporarily disables measuring the execution time of statements and definitions.

`ProfilerPause`

Remarks:

- This procedure is the programmatic counterpart of the **Profiler - Pause** menu command.
- This procedure only has effect when the profiler has been activated.

See also:

The procedure `ProfilerContinue` and `ProfilerRestart`.

ProfilerContinue

The procedure `ProfilerContinue` continues measuring the execution time of statements and definitions.

`ProfilerContinue`

Remarks:

- This procedure is the programmatic counterpart of the **Profiler - Continue** menu command.
- This procedure only has effect when the profiler has been activated.

See also:

The procedure `ProfilerPause` and `ProfilerRestart`.

ProfilerRestart

The procedure `ProfilerRestart` clears the execution time measurement data of all statements and definitions.

`ProfilerRestart`

Remarks:

- This procedure is the programmatic counterpart of the **Profiler - Restart** menu command.
- This procedure only has effect when the profiler has been activated.

See also:

The procedure `ProfilerContinue` and `ProfilerPause`.

ScheduleAt

With the procedure `ScheduleAt` you schedule a specific procedure to be run at a specified moment in time.

```
ScheduleAt(  
    starttime,      ! (input) scalar string expression  
    procedure      ! (input) element of the set AllProcedures  
)
```

Arguments:

starttime

A string representing the time at which you want to start the execution of the specified procedure. This time must be represented using AIMMS' standard time format: "YYYY-MM-DD hh:mm:ss".

procedure

An element in the set `AllProcedures`. This procedure cannot have any arguments.

Return value:

The procedure returns 1 on success, and 0 if AIMMS could not schedule the procedure at the specified start time. On failure, the pre-defined identifier `CurrentErrorMessage` will contain a proper error message.

Remarks:

If at the specified start time AIMMS is busy running some other task, then the procedure will start as soon as AIMMS has finished this task. If you want to run a procedure at regular intervals, then you can re-schedule the procedure from within the scheduled procedure itself.

SessionArgument

With the procedure `SessionArgument` you can retrieve the string value of any user defined command line argument, that was specified during startup of AIMMS.

```
SessionArgument(  
    argno,          ! (input) integer number  
    argument       ! (output) string valued parameter  
)
```

Arguments:

argno

An integer greater or equal to 1, representing the argument that you want retrieve. If the argument does not exist, then the procedure returns 0.

argument

A string valued parameter, to hold the string of the requested command line argument.

Return value:

The procedure returns 1 on success, and 0 if the request argument number does not exist.

Remarks:

When you open an AIMMS project from the command line, AIMMS allows you to add an arbitrary number of additional arguments directly after the project name. The procedure `SessionArgument` gives you access to these arguments. You can use these arguments, for instance, to specify a varying data source name from which you want to read data into your model, or run your project in different modes.

SessionHasVisibleGUI

With the function `SessionHasVisibleGUI` you can retrieve the visibility status of the current AIMMS session.

```
SessionHasVisibleGUI()
```

Return value:

The function returns 1 if the session has a visible graphical user interface and 0 if the session does not have a visible interface.

Remarks:

If you have an application that can be run both as a standalone application or as a component, this function can help you to distinguish between the two.

ShowHelpTopic

With the procedure ShowHelpTopic you can jump to a specific help topic in a help file.

```
ShowHelpTopic(  
    topic,          ! (input) scalar string  
    [helpfile]     ! (optional) scalar string  
)
```

Arguments:

topic

A string representing the help topic to jump to.

helpfile (optional)

A string representing the help file to open. If not specified, then AIMMS will use the help file that is specified in the project options.

Remarks:

AIMMS supports the following help file formats: WinHelp or WinHelp2000 (*.hlp), compiled HTML Help (*.chm), and Acrobat Reader (*.pdf).

ShowMessageWindow

With the procedure ShowMessageWindow you programmatically open or close the AIMMS message window.

```
ShowMessageWindow(  
    [do_show]      ! (optional) scalar expression  
)
```

Arguments:

do_show (optional)

A scalar 0-1 expression, indicating whether the message window should be opened (value is 1) or should be closed (value is 0). The default is 1.

See also:

The procedure [ShowProgressWindow](#).

ShowProgressWindow

With the procedure ShowProgressWindow you programmatically open or close the AIMMS progress window.

```
ShowProgressWindow(  
    [do_show]      ! (optional) scalar expression  
)
```

Arguments:

do_show (optional)

A scalar 0-1 expression, indicating whether the progress window should be opened (value is 1) or should be closed (value is 0). The default is 1.

See also:

The procedure [ShowMessageWindow](#).

TestInternetConnection

With the procedure `TestInternetConnection` you can verify whether an internet connection to a given URL is possible.

```
TestInternetConnection(  
    url                ! (input) scalar string expression  
)
```

Arguments:

url

A string representing the address of the internet site AIMMS will try to reach.

Return value:

The procedure returns 1 on success, and 0 if AIMMS could not establish a connection to the specified address (by pinging). On failure, the pre-defined identifier `CurrentErrorMessage` will contain a proper error message.

Remarks:

This procedure will only check whether the host as specified in the `url` can be reached, not whether a certain service is running nor whether a certain internet page exists.