

---

## **AIMMS Function Reference - Data Manager 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](http://www.aimms.com)

# Data Manager Functions

AIMMS supports the following Data Manager functions, that are not specific for cases or datasets only:

- CaseTypeCategories
- CaseTypeContents
- DataCategoryContents
- DataFileCopy
- DataFileExists
- DataFileGetAcronym
- DataFileGetComment
- DataFileGetDescription
- DataFileGetGroup
- DataFileGetName
- DataFileGetOwner
- DataFileGetPath
- DataFileGetTime
- DataFileReadPermitted
- DataFileSetAcronym
- DataFileSetComment
- DataFileWritePermitted
- DataImport220
- DataManagerFileNew
- DataManagerFileOpen
- DataManagerFileGetCurrent
- DataManagerExport
- DataManagerImport
- DataManagementExit

---

## CaseTypeCategories

The procedure `CaseTypeCategories` retrieves the sub-collection of data categories that is included in a specific case type.

```
CaseTypeCategories(  
    case_type,      ! (input) element of the set AllCaseTypes  
    category_set    ! (output) subset of AllDataCategories  
)
```

### Arguments:

*case\_type*

An element of the set `AllCaseTypes`, referring to the case type for which you want to retrieve the included data categories.

*category\_set*

A subset of the set `AllDataCategories`, on successful return this subset is filled with the data categories included in the case type.

### Return value:

The procedure returns 1 on success, 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [CaseGetType](#), [CaseTypeContents](#), [DataCategoryContents](#).

---

## CaseTypeContents

The procedure `CaseTypeContents` retrieves the sub-collection of identifiers that is contained in a specific case type.

```
CaseTypeContents(  
    case\_type,          ! (input) element of the set AllCaseTypes  
    identifier\_set      ! (output) subset of AllIdentifiers  
)
```

### Arguments:

*case\_type*

An element of the set `AllCaseTypes`, referring to the case type for which you want to retrieve the contents.

*identifier\_set*

A subset of the set `AllIdentifiers`, on successful return this subset is filled with all identifiers contained in the case type.

### Return value:

The procedure returns 1 on success, 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.
- The procedure returns the contents of the case type itself, as well as the contents of all data categories that are included in the case type.

### See also:

The procedures [CaseGetType](#), [CaseTypeCategories](#), [DataCategoryContents](#).

---

## DataCategoryContents

The procedure `DataCategoryContents` retrieves the sub-collection of identifiers that is contained in a specific data category.

```
DataCategoryContents(  
    data_category,      ! (input) element of the set AllDataCategories  
    identifier_set      ! (output) subset of AllIdentifiers  
)
```

### Arguments:

*data\_category*

An element of the set `AllDataCategories`, referring to the data category for which you want to retrieve the contents.

*identifier\_set*

A subset of the set `AllIdentifiers`, on successful return this subset is filled with all identifiers contained in the data category.

### Return value:

The procedure returns 1 on success, 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [CaseTypeCategories](#), [CaseTypeContents](#).

---

## DataFileCopy

With the procedure `DataFileCopy` you can copy a data file stored within a data manager file, to another data file within the same data manager file.

```
DataFileCopy(  
    datafile,      ! (input) element in the set AllDataFiles  
    acronym,      ! (input) string  
    copiedDatafile ! (output) element parameter into AllDataFiles  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`, `AllCases` or `AllDataSets`.

*acronym*

The name of the new data file to be created

*copiedDatafile*

On success, contains the element in `AllDataFiles` associated with the `datafile` into which the original data file was copied.

### Return value:

The procedure returns 1 if the data file has been copied, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.
- If a datafile with the given acronym already exists in the data manager file, the call to `DataFileCopy` will fail.

---

## DataFileExists

With the procedure `DataFileExists` you can check whether a specific element from the set `AllDataFiles` still refers to a valid case or dataset. Especially when multiple users have access to the same data file, an element may become invalid.

```
DataFileExists(  
    datafile      ! (input) element in the set AllDataFiles  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`, `AllCases` or `AllDataSets`.

### Return value:

The procedure returns 1 if the given `datafile` still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.
- Note that `AllCases` and `AllDataSets` are subsets of `AllDataFiles`.

### See also:

The procedure [DataFileName](#).

---

## DataFileGetAcronym

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileGetAcronym` you can obtain the acronym that is specified in the data manager for any element of the set `AllDataFiles` (cases or datasets).

```
DataFileGetAcronym(  
    datafile,    ! (input) element in the set AllDataFiles  
    acronym     ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*acronym*

A scalar string valued parameter. On return this parameter will contain the acronym of the datafile. If no acronym is specified, then an empty string is returned.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileName](#).

---

## DataFileGetComment

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileGetComment` you can obtain the comment that is specified in the data manager for any element of the set `AllDataFiles` (cases or datasets).

```
DataFileGetComment(  
    datafile,    ! (input) element in the set AllDataFiles  
    comment     ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*comment*

A scalar string valued parameter. On return this parameter will contain the comment of the datafile. If no comment is specified, then an empty string is returned.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures `DataFileExists`, `DataFileName`.

---

## DataFileGetDescription

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileGetDescription` you can obtain the description that the user entered via the properties of a case or dataset.

```
DataFileGetDescription(  
    datafile,      ! (input) element in the set AllDataFiles  
    description    ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*name*

A scalar string valued parameter. On return this parameter will contain the description of the datafile. If no description has been specified, then this string is empty.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetName](#), [DataFileGetAcronym](#).

---

## DataFileGetGroup

With the procedure `DataFileGetGroup` you can obtain the group name associated with the user that currently owns a specific case or dataset.

```
DataFileGetGroup(  
    datafile,      ! (input) element in the set AllDataFiles  
    group         ! (output) scalar string parameter  
)
```

### Arguments:

#### *datafile*

An element in the set `AllDataFiles`.

#### *group*

A scalar string valued parameter. On return this parameter will contain the group name associated with the user that owns the datafile. If there is no current owner, or if the project does not have a user database associated with it, then an empty string is returned.

### Return value:

The procedure returns 1 if the given datafile exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetOwner](#).

---

## DataFileGetName

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileGetName` you can obtain the name in the data manager for any element of the set `AllDataFiles` (cases or datasets).

```
DataFileGetName(  
    datafile,    ! (input) element in the set AllDataFiles  
    name        ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*name*

A scalar string valued parameter. On return this parameter will contain the name of the datafile. This name does not include the name of the folder(s) in which it is located.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetPath](#), [DataFileGetAcronym](#).

---

## DataFileGetOwner

With the procedure `DataFileGetOwner` you can obtain the name of the user that currently owns a specific case or dataset.

```
DataFileGetOwner(  
    datafile,      ! (input) element in the set AllDataFiles  
    owner         ! (output) scalar string parameter  
)
```

### Arguments:

#### *datafile*

An element in the set `AllDataFiles`.

#### *owner*

A scalar string valued parameter. On return this parameter will contain the name of the user that owns the datafile. If there is no current owner, or if the project does not have a user database associated with it, then an empty string is returned.

### Return value:

The procedure returns 1 if the given datafile exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetGroup](#).

---

## DataFileGetPath

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileGetPath` you can obtain the path in the data manager for any element of the set `AllDataFiles` (cases or datasets). The path of a datafile consists of a sequence folder names and the name of the datafile itself, separated by backslash characters.

```
DataFileGetPath(  
    datafile,    ! (input) element in the set AllDataFiles  
    path        ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*path*

A scalar string valued parameter. On return this parameter will contain the path of the datafile.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetName](#), [DataFileGetAcronym](#).

---

## DataFileGetTime

With the procedure `DataFileGetTime` you can obtain the time on which the data of a specific case or dataset was last modified (saved).

```
DataFileGetTime(  
    datafile,      ! (input) element in the set AllDataFiles  
    time           ! (output) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*time*

A scalar string valued parameter. On return this parameter will contain a string representation of the modification time, using AIMMS' standard date and time format: "YYYY-MM-DD hh:mm:ss".

### Return value:

The procedure returns 1 if the given datafile exists and contains saved data. If the datafile does not exist, or if no data has yet been saved in the datafile, then the procedure returns 0.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedures `DataFileExists`, `FileTime`.

---

## DataFileReadPermitted

With the procedure `DataFileReadPermitted` you can check whether the current user has read permission for the specified case or dataset. For example, you can use this procedure to issue your own error message if the permission is not granted. If the current user does not have read permission, then any call to a data manager procedure that involves a read operation will result in an error message, and fails.

```
DataFileReadPermitted(  
    datafile      ! (input) element in the set AllDataFiles  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

### Return value:

The procedure returns 1 if the current user does have read permission, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedure [DataFileWritePermitted](#).

---

## DataFileSetAcronym

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileSetAcronym` you can set the acronym for the data file corresponding to any element of the set `AllDataFiles` (cases or datasets).

```
DataFileSetAcronym(  
    datafile,    ! (input) element in the set AllDataFiles  
    acronym     ! (input) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*acronym*

A scalar string valued parameter. This parameter contains the acronym to be associated with the datafile. If an empty string is specified, any existing acronym will be deleted.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetAcronym](#).

---

## DataFileSetComment

The predefined set `AllDataFiles` (and its subsets `AllCases` and `AllDataSets`), is an integer set. The mapping of these integers onto the cases and datasets in the project is maintained by the data manager, and is not editable. With the procedure `DataFileSetComment` you can set the comment for the data file corresponding to any element of the set `AllDataFiles` (cases or datasets).

```
DataFileSetComment(  
    datafile,    ! (input) element in the set AllDataFiles  
    comment     ! (input) scalar string parameter  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

*comment*

A scalar string valued parameter. This parameter contains the comment to be associated with the datafile. If an empty string is specified, any existing comment will be deleted.

### Return value:

The procedure returns 1 if the given datafile still exists, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedures [DataFileExists](#), [DataFileGetComment](#).

---

## DataFileWritePermitted

With the procedure `DataFileWritePermitted` you can check whether the current user has write permission for the specified case or dataset. For example, you can use this procedure to issue your own error message if the permission is not granted. If the current user does not have write permission, then any call to a data manager procedure that involves a write (save) operation will result in an error message, and fails.

```
DataFileWritePermitted(  
    datafile      ! (input) element in the set AllDataFiles  
)
```

### Arguments:

*datafile*

An element in the set `AllDataFiles`.

### Return value:

The procedure returns 1 if the current user does have write permission, and 0 otherwise.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedure [DataFileReadPermitted](#).

---

## DataImport220

With the procedure `DataImport220` you can load a separate AIMMS case file, such as the case files that were created with AIMMS 2.20. After importing a case file using this procedure you can save the data as a new case node in the Data Management tree.

```
DataImport220(  
    filename      ! (input/output) a string parameter  
)
```

### Arguments:

*filename*

A string parameter, that on return will contain the name of the file that the user selected for importing.

### Return value:

The procedure returns 1 on success. If the user canceled the operation, then the procedure returns 0. If any other error occurs then the procedure returns -1 and `CurrentErrorMessage` will contain a proper error message.

### Remarks:

- This procedure is only applicable if the project option Data Management style is set to Single Data Manager file.
- This procedure is especially useful for converting old cases to the new AIMMS.

### See also:

The procedure [CaseSaveAs](#).

---

## DataManagerFileNew

With the procedure `DataManagerFileNew` you can create a new, empty data file. On success, the new data file will be used as the current data file for the project.

```
DataManagerFileNew(  
    filename,      ! (input) a scalar string expression  
    [UseAsDefault] ! (optional, default 1) a scalar binary expression  
)
```

### Arguments:

#### *filename*

A string containing the name of the new data file (relative to the project directory)

#### *UseAsDefault*

A binary value to indicate whether the new data file should be used as the default data file the next time the project is opened (value 1) or not (value 0).

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedures [DataManagerFileOpen](#), [DataManagerFileGetCurrent](#).

---

## DataManagerFileOpen

With the procedure `DataManagerFileOpen` you can open an existing data file. On success, the data file will be used as the current data file for the project.

```
DataManagerFileOpen(  
    filename,          ! (input) a scalar string expression  
    [UseAsDefault]    ! (optional, default 1) a scalar binary expression  
)
```

### Arguments:

#### *filename*

A string containing the name of the existing data file (relative to the project directory).

#### *UseAsDefault*

A binary value to indicate whether the data file should be used as the default data file the next time the project is opened (value 1) or not (value 0).

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedures [DataManagerFileNew](#), [DataManagerFileGetCurrent](#).

---

## DataManagerFileGetCurrent

With the procedure `DataManagerFileGetCurrent` you can obtain the name of the current data file.

```
DataManagerFileGetCurrent(  
    filename          ! (output) a scalar string  
)
```

### Arguments:

*filename*

A string to contain the name of the current data file (relative to the project directory).

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedures [DataManagerFileNew](#), [DataManagerFileOpen](#).

---

## DataManagerExport

With the procedure `DataManagerExport` you can export a collection of cases and datasets from the data management tree to a new data file.

```
DataManagerExport(  
    filename,      ! (input) a scalar string expression  
    datafiles     ! (input/output) a subset of AllDataFiles  
)
```

### Arguments:

#### *filename*

A string containing the name of the data file to which the cases and datasets must be exported.

#### *datafiles*

A subset of `AllDataFiles`, containing the cases and datasets that you want to export. Any dataset that is referred to by a case in this set is automatically added to the set.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- This function is only applicable if the project option Data Management style is set to Single Data Manager file.

### See also:

The procedure `DataManagerImport`.

---

## DataManagerImport

With the procedure `DataManagerImport` you can import the entire data management tree that is stored in another data file into your current data management tree. If the imported tree contains cases (or datasets) that already exist in the current tree, then you can choose whether these cases (or datasets) should overwrite the current nodes or should be imported as new nodes.

```
DataManagerImport(  
    filename,      ! (input) a scalar string expression  
    [overwrite]   ! (optional) 0, 1 or 2  
)
```

### Arguments:

#### *filename*

A string containing the name of the data file that must be imported.

#### *overwrite (optional)*

This integer indicates whether or not existing cases (or datasets) are overwritten by cases (or datasets) from the imported file. If 0 (the default), then a dialog box is displayed in which the user can decide to overwrite the existing node or to create a new node. If 1, then existing nodes are always overwritten. If 2, then all imported cases and datasets will result in new nodes in the tree.

### Return value:

The procedure returns 1 on success. If the user canceled the operation, then the procedure returns 0. If any other error occurs then the procedure returns -1 and `CurrentErrorMessage` will contain a proper error message.

### Remarks:

- This function is only applicable if the project option `Data Management style` is set to `Single Data Manager file`.

### See also:

The procedure [DataManagerExport](#).