
AIMMS Function Reference - Model Query 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

Model Query Functions

AIMMS supports the following functions to query the structure of the identifiers in the model:

- `AttributeToString`
- `DeclaredSubset`
- `DomainIndex`
- `IdentifierAttributes`
- `IdentifierDimension`
- `IdentifierText`
- `IdentifierType`
- `IdentifierUnit`
- `IndexRange`
- `IsRuntimeIdentifier`
- `ReferencedIdentifiers`
- `SectionIdentifiers`

```
SelectedIdentifiers := AllParameters ; ! Or some other selection.

put outf ;

outf.pagewidth := 255 ; ! Wide
put "type":20, " ", "name":32, " ", "dim ", "unit":20, " ", "Text", / ;
put "-"*20, " ", "-"*32, " ", "--- ", "-"*20, " ", "-"*40, / ;

for ( si ) do
    put IdentifierType( si ):20, " " ! Type
      si:32, " ", ! name
      "(" , IdentifierDimension( si ):1:0, ") ", ! dimension
      IdentifierUnit( si ):20, " ", ! unit
      IdentifierText( si ), / ! Documenting text.
endfor ;

putclose ;
```

AttributeToString

The function `AttributeToString` converts a specified attribute for a given identifier to a string.

```
AttributeToString(  
    IdentifierName,    ! (input) scalar element parameter  
    AttributeName     ! (input) scalar element parameter  
)
```

Arguments:

IdentifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which an attribute should be converted to a string.

AttributeName

An element expression in the predefined set `AllAttributeNames` specifying the attribute that should be converted to string format.

Return value:

This function returns a string representation of the attribute on success or the empty string otherwise and the predeclared identifier `CurrentErrorMessage` contains an appropriate error message.

Remarks:

When the identifier is in an encrypted section of the model, the string `Encrypted` is returned and the predeclared identifier `CurrentErrorMessage` contains an appropriate error message.

DeclaredSubset

The function `DeclaredSubset` returns 1 if both `subsetName` and `superName` refer to a one-dimensional set and `subsetName` is directly or indirectly declared to be a subset of `superName`.

```
DeclaredSubset(
    subsetName,      ! (input) scalar element parameter
    supersetName    ! (input) scalar element parameter
)
```

Arguments:

subsetName

An element expression in the predefined set `AllIdentifiers`.

supersetName

An element expression in the predefined set `AllIdentifiers`.

Return value:

This function returns 1 iff `subsetName` is directly or indirectly a subset of `supersetName`. If `subsetName` or `supersetName` does not refer to a one-dimensional set, this function will return 0 without any warning or error message.

Example:

With the following declarations:

```
SET:
  identifier : MasterSet
  index      : ms ;
```

```
SET:
  identifier : DomainSet
  subset of  : MasterSet
  index      : ds ;
```

```
SET:
  identifier : ActiveSet
  subset of  : DomainSet
  index      : as ;
```

```
FILE:
  identifier : outf
  name       : "outf.put" ;
```

The following statements:

```
put outf ;
put "ActiveSet(=DomainSet =", DeclaredSubset('ActiveSet', 'DomainSet'):0:0,/;
put "ActiveSet(=MasterSet =", DeclaredSubset('ActiveSet', 'MasterSet'):0:0,/;
```

```
put "MasterSet(=ActiveSet =", DeclaredSubset('MasterSet', 'ActiveSet'):0:0,;/;
put "MasterSet(=outf      =", DeclaredSubset('MasterSet', 'outf'      ):0:0,;/;
putclose ;
```

Return the following output.

```
ActiveSet(=DomainSet =1 ! ActiveSet is directly a subset of DomainSet
ActiveSet(=MasterSet =1 ! ActiveSet is indirectly a subset of MasterSet
MasterSet(=ActiveSet =0 ! But the reverse is not true.
MasterSet(=outf      =0 ! outf isn't even a set.
```

See also:

The function [IndexRange](#).

DomainIndex

The function `DomainIndex` returns the `indexPosition`-th index of `identifierName` as an element in `AllIdentifiers`.

```
DomainIndex(
  identifierName,    ! (input) scalar element parameter
  indexPosition)    ! (input) scalar integer parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which an index should be obtained.

indexPosition

An expression in the range $\{1..dim\}$ where *dim* is the dimension of `identifierName`.

Return value:

This function returns an element in the set `AllIdentifiers` representing the `indexPosition` index of `identifierName`. If `identifierName` is not an indexed parameter, variable or constraint, or if `indexPosition` is outside the range $\{1..dim\}$, the empty element is returned without further warning.

Example:

The following code uses the function `DomainIndex` to obtain the indices of the index domain of a parameter:

```
put outf ;
for ( IndexParameters | IdentifierDimension( IndexParameters ) > 0 ) do
  put IndexParameters:0, "(" ;
  while loopcount <= IdentifierDimension( IndexParameters ) do
    put DomainIndex( IndexParameters, loopcount ):0 ;
    if loopCount < IdentifierDimension( IndexParameters ) then put "," ; endif ;
  endwhile ;
  put ")", / ;
endfor ;
putclose ;
```

A fragment of the output of this code might look as follows:

```
LowFP(f,p)
UppFP(f,p)
Supply(c)
Demand(f)
```

See also:

The functions `IdentifierDimension`, `DeclaredSubset` and `IndexRange`.

IdentifierAttributes

The function `IdentifierAttributes` determines which attributes a specified identifier has.

```
IdentifierAttributes(  
  IdentifierName      ! (input) scalar element parameter  
)
```

Arguments:

IdentifierName

An element expression specifying the identifier for which the attributes should be determined.

Return value:

This function returns a subset of `AllAttributeNames` containing all the attributes for the specified identifier.

See also:

The predeclared identifier `AllAttributeNames`.

IdentifierDimension

The function `IdentifierDimension` returns the data dimension of `identifierName`.

```
IdentifierDimension(  
    identifierName)    ! (input) scalar element parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which the dimension should be obtained.

Return value:

This function returns a non-negative integer. If `identifierName` is not an identifier, an error message is issued. If `identifierName` is not an indexed parameter, variable or constraint, a 0 is returned without further warning.

Remarks:

This function replaces the deprecated suffix `.dim`.

See also:

- The functions `DomainIndex` and `IndexRange`.
- Section [23.4](#) of the Language Reference.
- The common example on page [344](#).

IdentifierText

The function `IdentifierText` returns the text of `identifierName` or, if the text is not specified, the name of the identifier.

```
IdentifierText(  
    identifierName)    ! (input) scalar element parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which the text should be obtained.

Return value:

This function returns a non-negative integer. If `identifierName` is not an identifier, an error message is issued. When the text is not specified, the name of the identifier is returned.

Remarks:

This function replaces the deprecated suffix `.txt`.

See also:

- The functions `IdentifierText`.
- Section [23.4](#) of the Language Reference.
- The common example on page [344](#).

IdentifierType

The function `IdentifierType` returns the type of `identifierName` as an element in `AllIdentifierTypes`.

```
IdentifierType(  
    identifierName)    ! (input) scalar element parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which the type should be obtained.

Return value:

This function returns a type as an element in `AllIdentifierTypes`. If `identifierName` is not an identifier, an error message is issued.

Remarks:

This function replaces the suffix `.type`; this suffix is deprecated.

See also:

- The functions `IdentifierDimension` and `IdentifierUnit`.
- Section [23.4](#) of the Language Reference.
- The common example on page [344](#).

IdentifierUnit

The function `IdentifierUnit` returns the unit of `identifierName` as it is declared.

```
IdentifierUnit(  
    identifierName)    ! (input) scalar element parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which the unit should be obtained.

Return value:

This function returns a unit. If `identifierName` is not an identifier, an error message is issued. If `identifierName` is not a parameter, variable or constraint, the unit `[]` is returned without further warning.

Remarks:

This function complements the suffix `.unit`; when the unit of an identifier is a unit parameter, this function will return that unit parameter, whilst the suffix `unit` will return the value of that unit parameter.

See also:

- The functions `IdentifierDimension` and `IdentifierType`.
- Section [23.4](#) of the Language Reference.
- The common example on page [344](#).

IndexRange

The function `IndexRange` returns the range of an index as an element in `AllIdentifiers`.

```
IndexRange(
  indexName      ! (input) scalar element parameter
)
```

Arguments:

indexName

An element expression in the predefined set `AllIdentifiers` specifying the index for which the range should be returned.

Return value:

This function returns the range of index `indexName` as an element in `AllIdentifiers`. If `indexName` is not an index or if it does not have a range the empty element is returned.

Example:

With the declarations

```
SET:
  identifier : MasterSet
  index      : a ;

INDEX:
  identifier : b
  range      : MasterSet ;

INDEX:
  identifier : c ;
```

The output of the statements

```
put "IndexRange( 'a' ) = \", IndexRange( 'a' ):10, \"\", / ;
put "IndexRange( 'b' ) = \", IndexRange( 'b' ):10, \"\", / ;
put "IndexRange( 'c' ) = \", IndexRange( 'c' ):10, \"\", / ;
```

is:

```
IndexRange( 'a' ) = "MasterSet "
IndexRange( 'b' ) = "MasterSet "
IndexRange( 'c' ) = "      "
```

See also:

The functions `DeclaredSubset` and `DomainIndex`.

IsRuntimeIdentifier

The function `IsRuntimeIdentifier` returns 1 when the argument `identifierName` is created at runtime.

```
IsRuntimeIdentifier(  
    identifierName)    ! (input) scalar element parameter
```

Arguments:

identifierName

An element expression in the predefined set `AllIdentifiers` specifying the identifier for which it should be determined whether or not it is created at runtime.

Return value:

This function returns 0 or 1. If `identifierName` is not an identifier, an error message is issued.

Remarks:

In order to determine whether or not the value of string parameter `myStr` is an identifier, you can use `StringToElement(AllIdentifiers, myStr)` or `myStr` in `AllIdentifiers`.

See also:

- The functions [StringToElement](#), [DeclaredSubset](#) and [IndexRange](#).
- Section [23.4](#) of the Language Reference.
- The common example on page [344](#).

ReferencedIdentifiers

The function `ReferencedIdentifiers` determines which identifiers are used in the specified attributes of a subset of `AllIdentifiers`.

```
ReferencedIdentifiers(  
    searchIdentSet    ! (input) subset of AllIdentifiers  
    searchAttrSet     ! (input) subset of AllAttributeNames  
    recursive        ! (optional) numerical expression  
)
```

Arguments:

searchIdentSet

The set of identifiers to search in for referenced identifiers.

searchAttrSet

The set of attributes to search in for referenced identifiers.

recursive

Optional argument, default 0, if 1 this function will also search in the referenced identifiers for identifier references.

Return value:

This function returns a subset of `AllIdentifiers` containing all the identifiers that are referenced in the attributes in *searchAttrSet* in one of the identifiers in *searchIdentSet*.

See also:

The predeclared identifiers `AllIdentifiers` and `AllAttributeNames`.

SectionIdentifiers

The function `SectionIdentifiers` determines which identifiers are declared within a specific section in the model tree.

```
SectionIdentifiers(  
  SectionName      ! (input) scalar element parameter  
)
```

Arguments:

SectionName

An element expression in the set `AllSections` specifying the section for which the identifiers should be listed.

Return value:

This function returns a subset of `AllIdentifiers` containing all the identifiers that are declared within the specified section, excluding the section itself and its prefix (if the section is a module or library). When `SectionName` is the empty element, the empty set is returned.

See also:

The predeclared identifiers `AllSections` and `AllIdentifiers`.