

---

## **AIMMS Function Reference - GMPColumn Procedures and 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)

## GMP::Column Procedures and Functions

AIMMS supports the following procedures and functions for creating and managing matrix columns associated with a generated mathematical program instance:

- `GMP::Column::Add`
- `GMP::Column::Delete`
- `GMP::Column::Freeze`
- `GMP::Column::GetLowerBound`
- `GMP::Column::GetScale`
- `GMP::Column::GetStatus`
- `GMP::Column::GetType`
- `GMP::Column::GetUpperBound`
- `GMP::Column::SetAsObjective`
- `GMP::Column::SetLowerBound`
- `GMP::Column::SetType`
- `GMP::Column::SetUpperBound`
- `GMP::Column::Unfreeze`

---

## GMP::Column::Add

The procedure `GMP::Column::Add` adds a column to the matrix of a generated mathematical program.

```
GMP::Column::Add(  
    GMP,           ! (input) a generated mathematical program  
    column        ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to a column.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

Coefficients for this column can be added to the matrix by using the procedure `GMP::Coefficient::Set`. After calling `GMP::Column::Add` the type and the lower and upper bound of the column are set according to the definition of the corresponding symbolic variable. By using the procedures `GMP::Column::SetType`, `GMP::Column::SetLowerBound` and `GMP::Column::SetUpperBound` the column type and the lower and upper bound can be changed.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Coefficient::Set`, `GMP::Column::Delete`, `GMP::Column::SetType`, `GMP::Column::SetLowerBound` and `GMP::Column::SetUpperBound`.

---

**GMP::Column::Delete**

The procedure `GMP::Column::Delete` marks a column in the matrix of a generated mathematical program as deleted.

```
GMP::Column::Delete(  
  GMP,           ! (input) a generated mathematical program  
  column        ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**Remarks:**

- The column will not be printed in the constraint listing, nor be visible in the math program inspector and it will be removed from any solver maintained copies.
- Use `GMP::Column::Add` to undo this action.

**See also:**

The routines `GMP::Instance::Generate` and `GMP::Column::Add`.

---

## GMP::Column::Freeze

The procedure `GMP::Column::Freeze` freezes a column in the matrix of a generated mathematical program at the given value.

```
GMP::Column::Freeze(
  GMP,          ! (input) a generated mathematical program
  column,      ! (input) a scalar reference
  value        ! (input) a numerical expression
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

*value*

The new value that should be used to freeze the column value.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

- The column remains visible in the constraint listing and math program inspector. In addition, it will be retained in solver maintained copies of the generated math program.
- Use `GMP::Column::Unfreeze` to undo the freezing.
- During a call to function `GMP::Column::Freeze` AIMMS stores the upper and lower bound of the column before the function was called. This information is used when function `GMP::Column::Unfreeze` is called thereafter. This information is not copied by the function `GMP::Instance::Copy`.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Column::Unfreeze` and `GMP::Instance::Copy`.

---

## GMP::Column::GetLowerBound

The function `GMP::Column::GetLowerBound` returns the lower bound of a column in the generated mathematical program.

```
GMP::Column::GetLowerBound(
    GMP,          ! (input) a generated mathematical program
    column       ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The lower bound value for the specified column.

### Remarks:

- If the column has a unit then the scaled lower bound is returned (without unit).
- This function can be used to retrieve the lower bound after presolving in case the *GMP* was created by `GMP::Instance::CreatePresolved`, even if the column was deleted.

### Examples:

Assume that 'x1' is a variable in mathematical program 'MP' with a unit as defined by:

```
QUANTITY:
  identifier   : SI_Mass
  base unit   : kg
  conversions  : ton -> kg : # -> # * 1000 ;

PARAMETER:
  identifier   : min_wght
  unit         : ton
  initial value : 20 ;

VARIABLE:
  identifier   : x1
  range        : [min_wght, inf)
  unit         : ton ;
```

If we want to multiply the lower bound by 1.5 and assign it as the new value by using function `GMP::Column::SetLowerBound` we can use

```
lb1 := 1.5 * (GMP::Column::GetLowerBound( 'MP', x1 )) [ton];  
GMP::Column::SetLowerBound( 'MP', x1, lb1 );
```

if 'lb1' is a parameter with unit [ton], or we can use

```
lb2 := 1.5 * GMP::Column::GetLowerBound( 'MP', x1 );  
GMP::Column::SetLowerBound( 'MP', x1, lb2 * GMP::Column::GetScale( 'MP', x1 ) );
```

if 'lb2' is a parameter without a unit.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Column::SetLowerBound`, `GMP::Column::GetUpperBound`, `GMP::Column::GetScale` and `GMP::Instance::CreatePresolved`.

---

## GMP::Column::GetScale

The function `GMP::Column::GetScale` returns the scaling factor of a column in the generated mathematical program.

```
GMP::Column::GetScale(  
  GMP,          ! (input) a generated mathematical program  
  column       ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The scaling factor for the specified column.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Row::GetScale`.

---

## GMP::Column::GetStatus

The function `GMP::Column::GetStatus` returns the status of a column in the matrix of a generated mathematical program.

```
GMP::Column::GetStatus(
    GMP,           ! (input) a generated mathematical program
    column        ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

An element in the predefined set `AllRowColumnStatuses`. The set `AllRowColumnStatuses` contains the following elements:

- Active,
- Deactivated,
- Deleted,
- NotGenerated,
- PresolveDeleted.

### Remarks:

- This function will return 'PresolveDeleted' only if the generated mathematical program has been created with `GMP::Instance::CreatePresolved`. Status 'PresolveDeleted' means that the column was generated for the original generated mathematical program but deleted when the presolved mathematical program was created.
- Status 'Deactivated' is not possible for columns.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Instance::CreatePresolved`.

---

## GMP::Column::GetType

The function `GMP::Column::GetType` returns the type of a column in the matrix of a generated mathematical program.

```
GMP::Column::GetType(  
    GMP,          ! (input) a generated mathematical program  
    column       ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

An element in the predefined set `AllColumnTypes`.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Column::SetType`.

---

## GMP::Column::GetUpperBound

The function `GMP::Column::GetUpperBound` returns the upper bound of a column in the generated mathematical program.

```
GMP::Column::GetUpperBound(
    GMP,          ! (input) a generated mathematical program
    column       ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The upper bound value for the specified column.

### Remarks:

- If the column has a unit then the scaled upper bound is returned (without unit).
- This function can be used to retrieve the upper bound after presolving in case the GMP was created by `GMP::Instance::CreatePresolved`, even if the column was deleted.

### Examples:

Assume that 'x1' is a variable in mathematical program 'MP' with a unit as defined by:

```
QUANTITY:
  identifier   : SI_Mass
  base unit   : kg
  conversions  : ton -> kg : # -> # * 1000 ;

PARAMETER:
  identifier   : max_wght
  unit        : ton
  initial value : 20 ;

VARIABLE:
  identifier   : x1
  range       : [0, max_wght]
  unit        : ton ;
```

If we want to multiply the upper bound by 1.5 and assign it as the new value by using function `GMP::Column::SetUpperBound` we can use

```
ub1 := 1.5 * (GMP::Column::GetUpperBound( 'MP', x1 )) [ton];  
GMP::Column::SetUpperBound( 'MP', x1, ub1 );
```

if 'ub1' is a parameter with unit [ton], or we can use

```
ub2 := 1.5 * GMP::Column::GetUpperBound( 'MP', x1 );  
GMP::Column::SetUpperBound( 'MP', x1, ub2 * GMP::Column::GetScale( 'MP', x1 ) );
```

if 'ub2' is a parameter without a unit.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Column::SetUpperBound`, `GMP::Column::GetLowerBound`, `GMP::Column::GetScale` and `GMP::Instance::CreatePresolved`.

---

## GMP::Column::SetAsObjective

The procedure `GMP::Column::SetAsObjective` sets a column as the new objective of a generated mathematical program.

```
GMP::Column::SetAsObjective(  
    GMP,          ! (input) a generated mathematical program  
    column       ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

- The column should be linear and have at least one coefficient in the matrix.
- The column should be free, i.e., not have a lower or upper bound.
- After a call to `GMP::Column::SetAsObjective` the old objective column will be treated as a normal column.

### See also:

The routines `GMP::Column::Add` and `GMP::Instance::CreateDual`.

---

## GMP::Column::SetLowerBound

The procedure `GMP::Column::SetLowerBound` changes the lower bound of a column in the generated mathematical program.

```
GMP::Column::SetLowerBound(
  GMP,          ! (input) a generated mathematical program
  column,       ! (input) a scalar reference
  value         ! (input) a numerical expression
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

*value*

The new value assigned to the lower bound of the column.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

If the column has a unit then *value* should have the same unit. If *value* has no unit then you should multiply it by the column scale, as returned by the function `GMP::Column::GetScale`.

### Examples:

Assume that 'x1' is a variable in mathematical program 'MP' with a unit as defined by:

```
QUANTITY:
  identifier   : SI_Mass
  base unit    : kg
  conversions  : ton -> kg : # -> # * 1000 ;

PARAMETER:
  identifier   : min_wght
  unit        : ton
  initial value : 20 ;

VARIABLE:
  identifier   : x1
  range       : [min_wght, inf)
  unit        : ton ;
```

Then if we run the following code

```
GMP::Column::SetLowerBound( 'MP', x1, 20 [ton] );
lb1 := GMP::Column::GetLowerBound( 'MP', x1 );
display lb1;

GMP::Column::SetLowerBound( 'MP', x1, 30 );
lb2 := GMP::Column::GetLowerBound( 'MP', x1 );
display lb2;

GMP::Column::SetLowerBound( 'MP', x1, 40 * GMP::Column::GetScale( 'MP', x1 ) );
lb3 := GMP::Column::GetLowerBound( 'MP', x1 );
display lb3;
```

(where 'lb1', 'lb2' and 'lb3' are parameters without a unit) we get the following results:

```
lb1 := 20 ;
lb2 := 0.030 ;
lb3 := 40 ;
```

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Column::SetUpperBound`, `GMP::Column::GetLowerBound` and `GMP::Column::GetScale`.

---

## GMP::Column::SetType

The procedure `GMP::Column::SetType` changes the type of a column in the matrix of a generated mathematical program.

```
GMP::Column::SetType(  
    GMP,          ! (input) a generated mathematical program  
    column,      ! (input) a scalar reference  
    type         ! (input) a element in AllColumnTypes  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

*type*

An element in `AllColumnTypes`.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The functions `GMP::Instance::Generate` and `GMP::Column::GetType`.

---

## GMP::Column::SetUpperBound

The procedure `GMP::Column::SetUpperBound` changes the upper bound of a column in the generated mathematical program.

```
GMP::Column::SetUpperBound(
  GMP,          ! (input) a generated mathematical program
  column,       ! (input) a scalar reference
  value         ! (input) a numerical expression
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

*value*

The new value assigned to the upper bound of the column.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

If the column has a unit then *value* should have the same unit. If *value* has no unit then you should multiply it by the column scale, as returned by the function `GMP::Column::GetScale`.

### Examples:

Assume that 'x1' is a variable in mathematical program 'MP' with a unit as defined by:

```
QUANTITY:
  identifier   : SI_Mass
  base unit    : kg
  conversions  : ton -> kg : # -> # * 1000 ;

PARAMETER:
  identifier   : max_wght
  unit         : ton
  initial value : 20 ;

VARIABLE:
  identifier   : x1
  range        : [0, max_wght]
  unit         : ton ;
```

Then if we run the following code

```
GMP::Column::SetUpperBound( 'MP', x1, 20 [ton] );
ub1 := GMP::Column::GetUpperBound( 'MP', x1 );
display ub1;

GMP::Column::SetUpperBound( 'MP', x1, 30 );
ub2 := GMP::Column::GetUpperBound( 'MP', x1 );
display ub2;

GMP::Column::SetUpperBound( 'MP', x1, 40 * GMP::Column::GetScale( 'MP', x1 ) );
ub3 := GMP::Column::GetUpperBound( 'MP', x1 );
display ub3;
```

(where 'ub1', 'ub2' and 'ub3' are parameters without a unit) we get the following results:

```
ub1 := 20 ;
ub2 := 0.030 ;
ub3 := 40 ;
```

**See also:**

The routines [GMP::Instance::Generate](#), [GMP::Column::SetLowerBound](#), [GMP::Column::GetUpperBound](#) and [GMP::Column::GetScale](#).

---

## GMP::Column::Unfreeze

The procedure `GMP::Column::Unfreeze` removes the frozen status of a column in the matrix of a generated mathematical program.

```
GMP::Column::Unfreeze(
    GMP,          ! (input) a generated mathematical program
    column       ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

During a call to function `GMP::Column::Freeze` AIMMS stores the upper and lower bound of the column before the function was called. This information is used when function `GMP::Column::Unfreeze` is called thereafter. This information is not copied by the function `GMP::Instance::Copy`. Therefore the call to `GMP::Column::Unfreeze` in the following piece of code is useless:

```
GMP::Column::Freeze( gmp1, x1, 20 );
gmp2 := GMP::Instance::Copy( gmp1, "cpy" );
GMP::Column::Unfreeze( gmp2, x1 );
```

### See also:

The routines `GMP::Instance::Generate`, `GMP::Column::Freeze` and `GMP::Instance::Copy`.