

---

## **AIMMS Function Reference - GMPRow 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::Row Procedures and Functions

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

- GMP::Row::Add
- GMP::Row::Activate
- GMP::Row::Deactivate
- GMP::Row::Delete
- GMP::Row::DeleteIndicatorCondition
- GMP::Row::Generate
- GMP::Row::GetConvex
- GMP::Row::GetIndicatorColumn
- GMP::Row::GetIndicatorCondition
- GMP::Row::GetLeftHandSide
- GMP::Row::GetRelaxationOnly
- GMP::Row::GetRightHandSide
- GMP::Row::GetScale
- GMP::Row::GetStatus
- GMP::Row::GetType
- GMP::Row::SetConvex
- GMP::Row::SetIndicatorCondition
- GMP::Row::SetLeftHandSide
- GMP::Row::SetRelaxationOnly
- GMP::Row::SetRightHandSide
- GMP::Row::SetType

---

**GMP::Row::Activate**

The procedure `GMP::Row::Activate` activates a deactivated row in a generated mathematical program.

```
GMP::Row::Activate(  
  GMP,      ! (input) a generated mathematical program  
  row       ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The procedure returns 1 on success, and 0 otherwise.

**See also:**

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

---

## GMP::Row::Add

The procedure `GMP::Row::Add` adds an empty row to the matrix of a generated mathematical program.

```
GMP::Row::Add(
  GMP,           ! (input) a generated mathematical program
  row           ! (input) a scalar reference
)
```

### Arguments:

*GMP*  
An element in `AllGeneratedMathematicalPrograms`.

*row*  
A scalar reference to a row.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- Coefficients for this row can be added to the matrix by using the procedure `GMP::Coefficient::Set`.
- After calling `GMP::Row::Add` the type and the left-hand-side and right-hand-side values are set according to the definition of the corresponding symbolic constraint. By using the procedures `GMP::Row::SetType`, `GMP::Row::SetLeftHandSide` and `GMP::Row::SetRightHandSide` the row type and row bounds can be changed.
- Use procedure `GMP::Row::Generate` to generate a (non-empty) row according to the definition of its associated symbolic constraint.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Coefficient::Set`, `GMP::Row::Delete`, `GMP::Row::SetType`, `GMP::Row::SetLeftHandSide`, `GMP::Row::SetRightHandSide` and `GMP::Row::Generate`.

---

**GMP::Row::Deactivate**

The procedure `GMP::Row::Deactivate` deactivates a row in a generated mathematical program. A deactivated row will not be passed to a solver session.

```
GMP::Row::Deactivate(  
  GMP,          ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The procedure returns 1 on success, and 0 otherwise.

**See also:**

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

---

**GMP::Row::Delete**

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

```
GMP::Row::Delete(  
  GMP,           ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**Remarks:**

- A deleted row remains present in the generated mathematical program but its contents will not be copied to a solver session.
- The row 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.

**See also:**

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

---

**GMP::Row::DeleteIndicatorCondition**

The procedure `GMP::Row::DeleteIndicatorCondition` deletes an indicator column and condition from a row in a generated mathematical program.

```
GMP::Row::DeleteIndicatorCondition(  
    GMP,          ! (input) a generated mathematical program  
    row          ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The procedure returns 1 on success, and 0 otherwise.

**Remarks:**

This procedure transforms an indicator row into a normal row.

**See also:**

The routines `GMP::Row::GetIndicatorColumn`, `GMP::Row::GetIndicatorCondition` and `GMP::Row::SetIndicatorCondition`.

---

## GMP::Row::Generate

The procedure `GMP::Row::Generate` generates a row and adds it to the matrix of a generated mathematical program. The row is generated according to the definition of its associated symbolic constraint.

```
GMP::Row::Generate(
  GMP,          ! (input) a generated mathematical program
  row,          ! (input) a scalar reference
  [autoAddColumn] ! (optional) a binary scalar
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to a row.

*autoAddColumn*

A binary scalar indicating whether this procedure should automatically add columns that are not in the *GMP*. The default is 0 meaning that no columns are added.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- Before generating the row all existing matrix coefficients for this row are removed.
- The row type and the right-hand-side value (and, if the row type is 'ranged', the left-hand-side value) are set according to the constraint definition.
- This procedure cannot be used for a row that contains the objective variable.
- If the value of *autoAddColumn* equals 0, then this procedure will generate an error if it encounters a column that is not in the *GMP*. You then have to add that column before calling this procedure by using the procedure `GMP::Column::Add`.
- Setting the value of *autoAddColumn* to 1 should only be done if you know exactly which columns are automatically added by this procedure. Otherwise you might end up with a model in which some columns only appear in this row making this row redundant.
- This procedure will never add columns that were deleted before with the procedure `GMP::Column::Delete`.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Column::Add`, `GMP::Column::Delete`, `GMP::Row::Add` and `GMP::Row::Delete`.

---

**GMP::Row::GetConvex**

The function `GMP::Row::GetConvex` returns 1 for a row in a generated mathematical program if it has been marked as being convex; otherwise it returns 0.

```
GMP::Row::GetConvex(  
  GMP,          ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The function returns 1 if the row is convex, and 0 otherwise.

**Remarks:**

AIMMS cannot detect whether a row is convex or not. A row is marked as being convex if the procedure `GMP::Row::SetConvex` has been called before or if the Convex suffix has been set to 1 for the corresponding constraint.

**See also:**

The procedure `GMP::Row::SetConvex`. The Convex suffix is explained in full detail in Section [14.2.6](#) of the Language Reference.

---

**GMP::Row::GetIndicatorColumn**

The function `GMP::Row::GetIndicatorColumn` returns, for a row in a generated mathematical program, the column number of the indicator column.

```
GMP::Row::GetIndicatorColumn(  
  GMP,          ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The function returns the column number if the indicator column exists, and -1 otherwise.

**See also:**

The routines `GMP::Row::DeleteIndicatorCondition`, `GMP::Row::GetIndicatorCondition` and `GMP::Row::SetIndicatorCondition`.

---

**GMP::Row::GetIndicatorCondition**

The function `GMP::Row::GetIndicatorCondition` returns the indicator condition of a row in a generated mathematical program.

```
GMP::Row::GetIndicatorCondition(  
    GMP,          ! (input) a generated mathematical program  
    row           ! (input) a scalar reference  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

**Return value:**

The function returns the indicator condition.

**Remarks:**

This function fails if the row has no indicator column.

**See also:**

The routines `GMP::Row::DeleteIndicatorCondition`, `GMP::Row::GetIndicatorColumn` and `GMP::Row::SetIndicatorCondition`.

---

## GMP::Row::GetLeftHandSide

The function `GMP::Row::GetLeftHandSide` returns the left-hand-side value of a row as present in the generated mathematical program.

```
GMP::Row::GetLeftHandSide(
    GMP,          ! (input) a generated mathematical program
    row          ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The function returns the left-hand-side value of the specified row.

### Remarks:

If the row has a unit then the scaled left-hand-side value is returned (without unit).

### Examples:

Assume that 'c1' is a constraint in mathematical program 'MP' with a unit as defined by:

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

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

PARAMETER:
  identifier   : wght_upper
  unit        : ton
  initial value : 60 ;

CONSTRAINT:
  identifier   : c1
  unit        : ton
  definition   : wght_lower <= -x1 + 2 * x2 <= wght_upper ;
```

If we want to multiply the left-hand-side value by 1.5 and assign it as the new value by using function `GMP::Row::SetLeftHandSide` we can use

```
lhs1 := 1.5 * (GMP::Row::GetLeftHandSide( 'MP', c1 )) [ton];  
GMP::Row::SetLeftHandSide( 'MP', c1, lhs1 );
```

if 'lhs1' is a parameter with unit [ton], or we can use

```
lhs2 := 1.5 * GMP::Row::GetLeftHandSide( 'MP', c1 );  
GMP::Row::SetLeftHandSide( 'MP', c1, lhs2 * GMP::Row::GetScale( 'MP', c1 ) );
```

if 'lhs2' is a parameter without a unit.

**See also:**

The routines [GMP::Instance::Generate](#), [GMP::Row::SetLeftHandSide](#), [GMP::Row::GetRightHandSide](#) and [GMP::Row::GetScale](#).

---

## GMP::Row::GetRelaxationOnly

The function `GMP::Row::GetRelaxationOnly` returns 1 for a row in a generated mathematical program if it has been marked as being a relaxation-only row; otherwise it returns 0.

```
GMP::Row::GetRelaxationOnly(  
    GMP,          ! (input) a generated mathematical program  
    row          ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The function returns 1 if the row is a relaxation-only row, and 0 otherwise.

### Remarks:

A row is marked as being a relaxation-only row if the procedure `GMP::Row::SetRelaxationOnly` has been called before or if the `RelaxationOnly` suffix has been set to 1 for the corresponding constraint.

### See also:

The procedure `GMP::Row::SetRelaxationOnly`. The `RelaxationOnly` suffix is explained in full detail in Section [14.2.6](#) of the Language Reference.

---

## GMP::Row::GetRightHandSide

The function `GMP::Row::GetRightHandSide` returns the right-hand-side value of a row as present in the generated mathematical program.

```
GMP::Row::GetRightHandSide(
    GMP,          ! (input) a generated mathematical program
    row          ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The function returns the right-hand-side value of the specified row.

### Remarks:

If the row has a unit then the scaled right-hand-side value is returned (without unit).

### Examples:

Assume that 'c1' is a constraint in mathematical program 'MP' with a unit as defined by:

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

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

CONSTRAINT:
  identifier   : c1
  unit        : ton
  definition   : -x1 + 2 * x2 <= wght ;
```

If we want to multiply the right-hand-side value by 1.5 and assign it as the new value by using function `GMP::Row::SetRightHandSide` we can use

```
rhs1 := 1.5 * (GMP::Row::GetRightHandSide( 'MP', c1 )) [ton];
GMP::Row::SetRightHandSide( 'MP', c1, rhs1 );
```

if 'rhs1' is a parameter with unit [ton], or we can use

```
rhs2 := 1.5 * GMP::Row::GetRightHandSide( 'MP', c1 );  
GMP::Row::SetRightHandSide( 'MP', c1, rhs2 * GMP::Row::GetScale( 'MP', c1 ) );
```

if 'rhs2' is a parameter without a unit.

**See also:**

The routines [GMP::Instance::Generate](#), [GMP::Row::SetRightHandSide](#), [GMP::Row::GetLeftHandSide](#) and [GMP::Row::GetScale](#).

---

## GMP::Row::GetScale

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

```
GMP::Row::GetScale(  
  GMP,          ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The scaling factor for the specified row.

### See also:

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

---

## GMP::Row::GetStatus

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

```
GMP::Row::GetStatus(  
    GMP,          ! (input) a generated mathematical program  
    row          ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row 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 row was generated for the original generated mathematical program but deleted when the presolved mathematical program was created.

### See also:

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

---

## GMP::Row::GetType

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

```
GMP::Row::GetType(  
  GMP,           ! (input) a generated mathematical program  
  row           ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The function returns an element in the predefined set `AllRowTypes`.

### See also:

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

---

## GMP::Row::SetConvex

The procedure `GMP::Row::SetConvex` can be used to indicate that a row in a generated mathematical is convex. Some solvers (like BARON) can make use of this information.

```
GMP::Row::SetConvex(  
    GMP,          ! (input) a generated mathematical program  
    row,          ! (input) a scalar reference  
    value         ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*value*

A scalar reference to a 0-1 value.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

AIMMS cannot detect whether a row is convex or not. A row is marked as being convex after this procedure is called with the *value* argument equal to 1 or if the `Convex` suffix has been set to 1 for the corresponding constraint.

### See also:

The function `GMP::Row::GetConvex`. The `Convex` suffix is explained in full detail in Section 14.2.6 of the Language Reference.

---

## GMP::Row::SetIndicatorCondition

The procedure `GMP::Row::SetIndicatorCondition` assigns an indicator column and condition to a row in a generated mathematical program.

```
GMP::Row::SetIndicatorCondition(  
    GMP,          ! (input) a generated mathematical program  
    row,          ! (input) a scalar reference  
    column,      ! (input) a scalar reference  
    value        ! (input) a numerical expression  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*column*

A scalar reference to an existing column in the matrix.

*value*

A binary value that will be used as indicator condition.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

- Assigning an indicator column and condition to a row means that the row must (only) be satisfied if the level value of the indicator column equals the indicator condition.
- This procedure fails if the row is nonlinear or if the column is not binary.

### See also:

The routines `GMP::Row::DeleteIndicatorCondition`, `GMP::Row::GetIndicatorColumn` and `GMP::Row::GetIndicatorCondition`.

---

## GMP::Row::SetLeftHandSide

The procedure `GMP::Row::SetLeftHandSide` changes the left-hand-side of a row in a generated mathematical program.

```
GMP::Row::SetLeftHandSide(
  GMP,          ! (input) a generated mathematical program
  row,          ! (input) a scalar reference
  value         ! (input) a numerical expression
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*value*

The new value that should be assigned to the left-hand-side of the row.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

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

### Examples:

Assume that 'c1' is a constraint in mathematical program 'MP' with a unit as defined by:

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

CONSTRAINT:
  identifier : c1
  unit       : ton
  definition  : -x1 + 2 * x2 <= wght ;
```

Then if we run the following code

```
GMP::Row::SetLeftHandSide( 'MP', c1, 20 [ton] );
lhs1 := GMP::Row::GetLeftHandSide( 'MP', c1 );
display lhs1;
```

```
GMP::Row::SetLeftHandSide( 'MP', c1, 30 );  
lhs2 := GMP::Row::GetLeftHandSide( 'MP', c1 );  
display lhs2;  
  
GMP::Row::SetLeftHandSide( 'MP', c1, 40 * GMP::Row::GetScale( 'MP', c1 ) );  
lhs3 := GMP::Row::GetLeftHandSide( 'MP', c1 );  
display lhs3;
```

(where 'lhs1', 'lhs2' and 'lhs3' are parameters without a unit) we get the following results:

```
lhs1 := 20 ;  
lhs2 := 0.030 ;  
lhs3 := 40 ;
```

**See also:**

The routines [GMP::Instance::Generate](#), [GMP::Row::SetRightHandSide](#), [GMP::Row::GetLeftHandSide](#) and [GMP::Row::GetScale](#).

---

## GMP::Row::SetRelaxationOnly

The procedure `GMP::Row::SetRelaxationOnly` can be used to indicate that a row in a generated mathematical is a relaxation-only row. Some solvers (like BARON) can make use of this information.

```
GMP::Row::SetRelaxationOnly(  
    GMP,          ! (input) a generated mathematical program  
    row,          ! (input) a scalar reference  
    value         ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*value*

A scalar reference to a 0-1 value.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

A row is marked as being a relaxation-only row after this procedure is called with the *value* argument equal to 1 or if the `RelaxationOnly` suffix has been set to 1 for the corresponding constraint.

### See also:

The function `GMP::Row::GetRelaxationOnly`. The `RelaxationOnly` suffix is explained in full detail in Section [14.2.6](#) of the Language Reference.

---

## GMP::Row::SetRightHandSide

The procedure `GMP::Row::SetRightHandSide` changes the right-hand-side of a row in a generated mathematical program.

```
GMP::Row::SetRightHandSide(
  GMP,          ! (input) a generated mathematical program
  row,          ! (input) a scalar reference
  value         ! (input) a numerical expression
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*value*

The new value that should be assigned to the right-hand-side of the row.

### Return value:

The procedure returns 1 on success, and 0 otherwise.

### Remarks:

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

### Examples:

Assume that 'c1' is a constraint in mathematical program 'MP' with a unit as defined by:

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

CONSTRAINT:
  identifier : c1
  unit       : ton
  definition  : -x1 + 2 * x2 <= wght ;
```

Then if we run the following code

```
GMP::Row::SetRightHandSide( 'MP', c1, 20 [ton] );
rhs1 := GMP::Row::GetRightHandSide( 'MP', c1 );
```

```
display rhs1;

GMP::Row::SetRightHandSide( 'MP', c1, 30 );
rhs2 := GMP::Row::GetRightHandSide( 'MP', c1 );
display rhs2;

GMP::Row::SetRightHandSide( 'MP', c1, 40 * GMP::Row::GetScale( 'MP', c1 ) );
rhs3 := GMP::Row::GetRightHandSide( 'MP', c1 );
display rhs3;
```

(where 'rhs1', 'rhs2' and 'rhs3' are parameters without a unit) we get the following results:

```
rhs1 := 20 ;
rhs2 := 0.030 ;
rhs3 := 40 ;
```

**See also:**

The routines [GMP::Instance::Generate](#), [GMP::Row::SetLeftHandSide](#), [GMP::Row::GetRightHandSide](#) and [GMP::Row::GetScale](#).

---

## GMP::Row::SetType

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

```
GMP::Row::SetType(  
  GMP,          ! (input) a generated mathematical program  
  row,          ! (input) a scalar reference  
  type         ! (input) a element in AllRowTypes  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*row*

A scalar reference to an existing row in the matrix.

*type*

An element in `AllRowTypes`.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

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