
AIMMS Function Reference - GMPInstance 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

GMP::Instance Procedures and Functions

AIMMS supports the following procedures and functions for creating and managing generated mathematical program instances:

- GMP::Instance::AddIntegerEliminationRows
- GMP::Instance::CalculateSubGradient
- GMP::Instance::Copy
- GMP::Instance::CreateDual
- GMP::Instance::CreateMasterMIP
- GMP::Instance::CreatePresolved
- GMP::Instance::CreateProgressCategory
- GMP::Instance::CreateSolverSession
- GMP::Instance::Delete
- GMP::Instance::DeleteIntegerEliminationRows
- GMP::Instance::DeleteSolverSession
- GMP::Instance::FindApproximatelyFeasibleSolution
- GMP::Instance::FixColumns
- GMP::Instance::Generate
- GMP::Instance::GenerateRobustCounterpart
- GMP::Instance::GenerateStochasticProgram
- GMP::Instance::GetDirection
- GMP::Instance::GetLinearObjective
- GMP::Instance::GetMathematicalProgrammingType
- GMP::Instance::GetMemoryUsed
- GMP::Instance::GetNumberOfColumns
- GMP::Instance::GetNumberOfIndicatorRows
- GMP::Instance::GetNumberOfIntegerColumns
- GMP::Instance::GetNumberOfNonlinearColumns
- GMP::Instance::GetNumberOfNonlinearNonzeros
- GMP::Instance::GetNumberOfNonlinearRows
- GMP::Instance::GetNumberOfNonzeros
- GMP::Instance::GetNumberOfRows
- GMP::Instance::GetNumberOfSOS1Rows
- GMP::Instance::GetNumberOfSOS2Rows
- GMP::Instance::GetObjective
- GMP::Instance::GetOptionValue
- GMP::Instance::GetSolver
- GMP::Instance::GetSymbolicMathematicalProgram

- GMP::Instance::MemoryStatistics
- GMP::Instance::Rename
- GMP::Instance::SetCallbackAddCut
- GMP::Instance::SetCallbackAddLazyConstraint
- GMP::Instance::SetCallbackBranch
- GMP::Instance::SetCallbackHeuristic
- GMP::Instance::SetCallbackIncumbent
- GMP::Instance::SetCallbackIterations
- GMP::Instance::SetCallbackNewIncumbent
- GMP::Instance::SetCallbackStatusChange
- GMP::Instance::SetCPUSecondsLimit
- GMP::Instance::SetCutoff
- GMP::Instance::SetDirection
- GMP::Instance::SetIterationLimit
- GMP::Instance::SetMathematicalProgrammingType
- GMP::Instance::SetMemoryLimit
- GMP::Instance::SetOptionValue
- GMP::Instance::SetSolver
- GMP::Instance::Solve

GMP::Instance::AddIntegerEliminationRows

The procedure `GMP::Instance::AddIntegerEliminationRows` adds integer elimination rows to the generated mathematical program which will eliminate an integer solution.

```
GMP::Instance::AddIntegerEliminationRows(
  GMP,          ! (input) a generated mathematical program
  solution,     ! (input) a solution
  elimNo       ! (input) an elimination number
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

solution

An integer scalar reference to a solution.

elimNo

An integer scalar reference to an elimination number.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- If the GMP is not integer then this procedure will fail.
- Rows and columns added before for *elimNo* will be deleted first.
- If the GMP contains only binary variables then only one row will be added; if the GMP contains general integer variables then several rows and columns will be added.
- The exact definitions of the rows and columns that are added are as follows. Let x_i be an integer column whose level value lev_i is between its lower bound lb_i and upper bound ub_i , i.e., $lb_i < lev_i < ub_i$. Then columns $l_i \geq 0$ and $u_i \geq 0$ are added together with a binary column z_i . Also the following three constraints are added:

$$l_i + (lev_i - lb_i)z_i \leq (lev_i - lb_i) \quad (29.1)$$

$$u_i + (lev_i - ub_i)z_i \leq 0 \quad (29.2)$$

$$x_i - u_i + l_i = lev_i \quad (29.3)$$

Every call to `GMP::Instance::AddIntegerEliminationRows` also adds the following constraint:

$$\sum_{i \in S_{lo}} x_i - \sum_{i \in S_{up}} x_i + \sum_{i \in S_{in}} (l_i + u_i) \geq 1 + \sum_{i \in S_{lo}} lev_i - \sum_{i \in S_{up}} lev_i \quad (29.4)$$

where S_{lo} defines the set of integer columns whose level values equal their lower bounds, S_{up} the set of integer columns whose level values equal their upper bounds, and S_{in} the set of integer columns whose level values are between their bounds.

- By using the suffixes `.ExtendedConstraint` and `.ExtendedVariable` it is possible to refer to the rows and columns respectively that are added by `GMP::Instance::AddIntegerEliminationRows`:

- Variables `v.ExtendedVariable('EliminationLowerBoundk', i)`, `v.ExtendedVariable('EliminationUpperBoundk', i)` and `v.ExtendedVariable('Eliminationk', i)` are added for each integer variable `v(i)` with the level value between its bounds. (These variables correspond to l_i , u_i and z_i respectively.)
- Constraints `v.ExtendedConstraint('EliminationLowerBoundk', i)`, `v.ExtendedConstraint('EliminationUpperBoundk', i)` and `v.ExtendedConstraint('Eliminationk', i)` are added for each integer variable `v(i)` with the level value between its bounds. (These constraints correspond to (29.1), (29.2) and (29.3) respectively.)
- Constraint `mp.ExtendedConstraint('Eliminationk')`, where `mp` denotes the symbolic mathematical program, is added for every call to `GMP::Instance::AddIntegerEliminationRows`. (This constraint corresponds to (29.4).)

Here k denotes the value of the argument `elimNo`.

Examples:

The procedure `GMP::Instance::AddIntegerEliminationRows` can be used to find the five best integer solutions for some MIP model:

```
gmp_mip := GMP::Instance::Generate(MIP_Model);

cnt := 1;

while ( cnt <= 5 ) do
  GMP::Instance::Solve(gmp_mip);

  ! Eliminate previous found integer solution.
  GMP::Instance::AddIntegerEliminationRows(gmp_mip,1,cnt);

  cnt += 1;

  ! Copy solution at position 1 to solution at position cnt
  ! in solution repository.
  GMP::Solution::Copy(gmp_mip,1,cnt);
endwhile;
```

After executing this code, the five best integer solutions will be stored at positions 2 - 6 in the solution repository, with the best solution at position 2 and the 5th best at position 6.

See also:

The routines `GMP::Instance::DeleteIntegerEliminationRows` and `GMP::Solution::IsInteger`. See Section 21.3.6 of the Language Reference for more details on extended suffixes.

GMP::Instance::CalculateSubGradient

The procedure `GMP::Instance::CalculateSubGradient` can be used to solve $By = x$ for a given vector x , where B is the basis matrix of a linear program. This procedure can only be called after the linear program has been solved to optimality.

```
GMP::Instance::CalculateSubGradient(
    GMP,          ! (input) a generated mathematical program
    variableSet, ! (input) a set of variables
    constraintSet, ! (input) a set of constraints
    [session]    ! (input, optional) a solver session
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`. The mathematical program should have model type LP or RMIP.

variableSet

A subset of `AllVariables`.

constraintSet

A subset of `AllConstraints`.

session

An element in the set `AllSolverSessions`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- Use the `.ExtendedConstraint('RhsChange',*)` suffix of the constraints in *constraintSet* to assign values to the vector x .
- The suffix `.ExtendedVariable('RhsChange',*)` of the variables in *variableSet* will be used to store the subgradient y .
- The suffices `.ExtendedConstraint` and `.ExtendedVariable` have no unit and are not scaled.
- This procedure should be called after a normal solve statement or after a successful call to procedure `GMP::Instance::Solve`.
- This procedure can also be called after a successful call to procedure `GMP::SolverSession::Execute` or procedure `GMP::SolverSession::AsynchronousExecute`. In that case the solver session should be passed using the *session* argument.
- A column corresponding to a variable in *variableSet* that is not part of *GMP* will be ignored. A row corresponding to a constraint in *constraintSet* that is not part of *GMP* will also be ignored.

- This procedure is only supported by CPLEX 9.1 and higher.
- This procedure cannot be used if the *GMP* is created by `GMP::Instance::CreateDual`.

Examples:

Assume that 'MP' is a linear mathematical program and $c(i)$ is a constraint and $v(j)$ is a variable in this mathematical program. The following example shows how to calculate a subgradient after a normal solve statement.

```
solve MP;

! The next statement needs to be called once.
AllGMPExtensions += { 'RhsChange' };

c.ExtendedConstraint('RhsChange', i) := 1.0;

GMP::Instance::CalculateSubGradient('MP', AllVariables, AllConstraints);

display v.ExtendedVariable('RhsChange', j);
```

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::SolverSession::Execute` and `GMP::SolverSession::AsynchronousExecute`. See Section 21.3.6 of the Language Reference for more details on extended suffices.

GMP::Instance::Copy

The function `GMP::Instance::Copy` creates a copy of a generated mathematical program and an associated new element in the set `AllGeneratedMathematicalPrograms`.

```
GMP::Instance::Copy(
  GMP,           ! (input) a generated mathematical program
  [name]        ! (optional) a string expression
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

name

A string that contains the name for the copy of the generated mathematical program.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- The *name* argument should be different from the name of the original generated mathematical program.
- If an element with name specified by the *name* argument is already present in the set `AllGeneratedMathematicalPrograms` the corresponding generated mathematical program will be replaced (or updated in case the same symbolic mathematical program is involved).
- All solutions in the solution repository of the generated mathematical program are also copied.
- The solver selection as specified by `GMP::Instance::SetSolver` (if any) will not be copied.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Rename` and `GMP::Instance::SetSolver`.

GMP::Instance::CreateDual

The function `GMP::Instance::CreateDual` generates a mathematical program that is the dual representation of the specified generated mathematical program. The generated mathematical program should have model type LP.

```
GMP::Instance::CreateDual(
    GMP,          ! (input) a generated mathematical program
    name         ! (input) a string expression
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

name

A string that holds the name for the dual of the generated mathematical program.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- The *name* argument should be different from the name of the original generated mathematical program.
- If an element with name specified by the *name* argument is already present in the set `AllGeneratedMathematicalPrograms` the corresponding generated mathematical program will be replaced (or updated in case the same symbolic mathematical program is involved).
- Before a generated mathematical program is dualized, AIMMS first transforms it temporary into a standard form using the following rules:
 - Each column x_i that is frozen to 0 is deleted.
 - For each column x_i with upper bound u_i , $u_i \neq 0$ and $u_i < \infty$, an extra row $x_i \leq u_i$ is added.
 - For each column x_i with lower bound l_i , $l_i \neq 0$ and $l_i > -\infty$, an extra row $x_i \geq l_i$ is added.
 - Each ranged row $l_j \leq a^T x \leq u_j$ ($l_j > -\infty$ and $u_j < \infty$) is replaced by two rows $l_j \leq a^T x$ and $a^T x \leq u_j$.
- By using the suffix `.ExtendedConstraint` it is possible to refer to the rows that are added to create the standard form:
 - The constraint `v.ExtendedConstraint('DualUpperBound', i)` is added for a variable `v(i)` with an upper bound unequal to 0 and `inf`.

- The constraint `v.ExtendedConstraint('DualLowerBound', i)` is added for a variable `v(i)` with a lower bound unequal to 0 and `-inf`.
- The constraints `c.ExtendedConstraint('DualLowerBound', j)` and `c.ExtendedConstraint('DualUpperBound', j)` replace a ranged constraint `c(j)`.
- The objective variable for the dual mathematical program will become `mp.ExtendedConstraint(DualObjective)` and the objective constraint will be `mp.ExtendedVariable(DualDefinition)`, where `mp` denotes the symbolic mathematical program.

Examples:

Assume that 'PrimalModel' is a mathematical program with the following declaration (in aim format):

```
VARIABLE:
  identifier : x1
  range      : [0, 5] ;

VARIABLE:
  identifier : x2
  range      : nonnegative ;

VARIABLE:
  identifier : obj
  definition : - 7 * x1 - 2 * x2 ;

CONSTRAINT:
  identifier : c1
  definition : -x1 + 2 * x2 <= 4 ;

MATHEMATICAL PROGRAM:
  identifier : PrimalModel
  objective  : obj
  direction  : minimize
  type       : lp ;
```

Then `GMP::Instance::CreateDual` will create a dual mathematical program with variables

name	lower	upper
<code>c1</code>	<code>-inf</code>	<code>0</code>
<code>obj_definition</code>	<code>-inf</code>	<code>inf</code>
<code>x1.ExtendedConstraint('DualUpperBound')</code>	<code>-inf</code>	<code>0</code>
<code>PrimalModel.ExtendedConstraint('DualObjective')</code>	<code>-inf</code>	<code>inf</code>

and constraints

```
x1:
  - c1 + 7 * obj_definition + x1.ExtendedConstraint('DualUpperBound') >= 0 ;

x2:
  + 2 * c1 + 2 * obj_definition >= 0 ;
```

```
obj:
  obj_definition = 1 ;

PrimalModel.ExtendedVariable('DualDefinition'):
  - 4 * c1 - 5 * x1.ExtendedConstraint('DualUpperBound')
  + PrimalModel.ExtendedConstraint('DualObjective') = 0 ;
```

See also:

The function `GMP::Instance::Generate`. See Section 21.3.6 of the Language Reference for more details on extended suffices.

GMP::Instance::CreateMasterMIP

The function `GMP::Instance::CreateMasterMIP` creates a Master MIP copy of the specified generated mathematical program. The copy will remove all nonlinear rows from the GMP.

```
GMP::Instance::CreateMasterMIP(
    GMP,          ! (input) a generated mathematical program
    name         ! (input) a string expression
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

name

A string that holds the name for the Master MIP.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- The *name* argument should be different from the name of the original generated mathematical program.
- If an element with name specified by the *name* argument is already present in the set `AllGeneratedMathematicalPrograms` the corresponding generated mathematical program will be replaced (or updated in case the same symbolic mathematical program is involved).
- The generated mathematical program should have type MINLP (or MIQP or MIQCP). It can also have type NLP in which case the created GMP will have type LP.
- If the objective constraint is nonlinear, `GMP::Instance::CreateMasterMIP` adds an extra row and column to the Master MIP. If `mp` denotes the symbolic mathematical program then the extra row will be associated with `mp.ExtendedConstraint(MasterMIPObjective)` and the extra column with `mp.ExtendedVariable(MasterMIPObjective)`. The extra row will be

$$\text{objvar} - \text{mp.ExtendedVariable(MasterMIPObjective)} = 0$$

where `objvar` denotes the objective variable of the GMP. Column `mp.ExtendedVariable(MasterMIPObjective)` will become the objective column of the Master MIP.

See also:

The function `GMP::Instance::Generate`. See Section 21.3.6 of the Language Reference for more details on extended suffices.

GMP::Instance::CreatePresolved

The function `GMP::Instance::CreatePresolved` generates a mathematical program that is the presolved representation of the specified generated mathematical program. The generated mathematical program can be a linear or nonlinear model, and should be generated using the function `GMP::Instance::Generate`.

```
GMP::Instance::CreatePresolved(
    GMP,          ! (input) a generated mathematical program
    name         ! (input) a string expression
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

name

A string that holds the name for the presolved mathematical program.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms`, with the name as specified by the *name* argument, if the presolver did not find an infeasibility. Else, the empty element.

Remarks:

- By using the functions `GMP::Column::GetStatus` and `GMP::Row::GetStatus` it is possible to check whether a column or row was deleted when the presolved mathematical program was created.
- By using the functions `GMP::Column::GetLowerBound` and `GMP::Column::GetUpperBound` it is possible to retrieve the lower and upper bound of a column in the presolved mathematical program.
- If the original *GMP* is deleted then the presolved *GMP* created by `GMP::Instance::CreatePresolved` will also be deleted.
- If the option `MINLP_probing` is switched on, then this function will change the mathematical programming type from MINLP (NLP) into MIP (LP) if the presolved model contains no nonlinear constraints.

Examples:

Assume that 'MP' is a mathematical program and 'gmpMP' and 'gmpPre' are element parameters with range 'AllGeneratedMathematicalPrograms'. To solve the presolved model using GMP functions we can use:

```
gmpMP := GMP::Instance::Generate( MP );
gmpPre := GMP::Instance::CreatePresolved( gmpMP, "PresolvedModel" );

GMP::Instance::Solve( gmpPre );
```

In case the GMP variant of the AOA module is used we can use:

```
gmpMP := GMP::Instance::Generate( MP );  
gmpPre := GMP::Instance::CreatePresolved( gmpMP, "PresolvedModel" );  
  
GMPOuterApprox::DoOuterApproximation( gmpPre );
```

Here 'GMPOuterApprox' is the prefix used by the GMP Outer Approximation Module.

See also:

The functions `GMP::Instance::Delete`, `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::Column::GetStatus`, `GMP::Row::GetStatus`, `GMP::Column::GetLowerBound` and `GMP::Column::GetUpperBound`.

GMP::Instance::CreateProgressCategory

The function `GMP::Instance::CreateProgressCategory` creates a new GMP progress category for a generated mathematical program. This progress category can be used to display GMP related information in the progress window.

```
GMP::Instance::CreateProgressCategory(
    GMP,                ! (input) a generated mathematical program
    [Name]              ! (input, optional) a string expression
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Name

A string that holds the name of the progress category.

Return value:

The function returns an element in the set `AllProgressCategories`.

Remarks:

- If no progress category is specified for the generated mathematical program then the GMP progress will be displayed in the general AIMMS progress category for GMP progress. This general AIMMS progress category will be used by all generated mathematical programs for which no progress category is specified. (Progress information for a normal solve is always displayed in the general AIMMS progress category.)
- After calling `GMP::Instance::CreateProgressCategory` solver progress will by default be displayed in the solver progress category of the generated mathematical program, and no longer in the general AIMMS progress category for solver progress.
- If the *Name* argument is not specified then the name of the generated mathematical program will be used to name the element in the set `AllProgressCategories`.
- The information displayed in a GMP progress category is controlled by AIMMS and cannot be modified by the user.
- A progress category created before for the generated mathematical program will be deleted.

See also:

The routines `GMP::ProgressWindow::DeleteCategory` and `GMP::SolverSession::CreateProgressCategory`.

GMP::Instance::CreateSolverSession

The function `GMP::Instance::CreateSolverSession` creates a new solver session for a generated mathematical program.

```
GMP::Instance::CreateSolverSession(
    GMP,                ! (input) a generated mathematical program
    [Name],             ! (input, optional) a string expression
    [Solver]            ! (input, optional) a solver
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Name

A string that holds the name of the solver session.

Solver

An element in the set `AllSolvers`.

Return value:

The function returns an element in the set `AllSolverSessions`.

Remarks:

- The function `GMP::Instance::CreateSolverSession` also determines which solver is assigned to the solver session. After the solver session is created it is not possible to change the solver assigned to the solver session! The solver is determined by:
 - the *Solver* argument if it is specified (and not an empty string), else
 - the solver that was assigned to the *GMP* if procedure `GMP::Instance::SetSolver` was called before, else
 - the default solver in AIMMS for the *GMP* its model type.
- If the *Name* argument is not specified, or if it is the empty string, the names of the symbolic mathematical program, the solver and the host (if any) are used to create a new element in the set `AllGeneratedMathematicalPrograms`.

See also:

The routines `GMP::Instance::DeleteSolverSession`, `GMP::Instance::SetSolver`, `GMP::SolverSession::GetInstance` and `GMP::SolverSession::GetSolver`.

GMP::Instance::Delete

The procedure `GMP::Instance::Delete` deletes a generated mathematical program from the set `AllGeneratedMathematicalPrograms`.

```
GMP::Instance::Delete(  
  GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

All memory associated with the generated mathematical program is also freed.

See also:

The function `GMP::Instance::Generate`.

GMP::Instance::DeleteIntegerEliminationRows

The procedure `GMP::Instance::DeleteIntegerEliminationRows` deletes a particular set of integer elimination rows and columns of a generated mathematical program.

```
GMP::Instance::DeleteIntegerEliminationRows(  
    GMP,          ! (input) a generated mathematical program  
    elimNo       ! (input) an elimination number  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

elimNo

An integer scalar reference to an elimination number.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The procedure `GMP::Instance::AddIntegerEliminationRows`.

GMP::Instance::DeleteSolverSession

The procedure `GMP::Instance::DeleteSolverSession` deletes the specified solver session.

```
GMP::Instance::DeleteSolverSession(  
    solverSession    ! (input) a solver session  
)
```

Arguments:

solverSession
An element in the set `AllSolverSessions`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The functions `GMP::Instance::CreateSolverSession` and `GMP::SolverSession::GetInstance`.

GMP::Instance::FindApproximatelyFeasibleSolution

The procedure `GMP::Instance::FindApproximatelyFeasibleSolution` tries to find an approximately feasible solution of a generated mathematical program. It uses the column level values of the first solution as a starting point. The approximately feasible solution is stored in the second solution.

The algorithm used to find the approximately feasible solution is based on the constraint consensus method as developed by John W. Chinneck. The constraint consensus method is an iterative projection algorithm. In each iteration a new point (i.e., a vector of column values) is constructed in such a way that it is likely that it is closer to the feasible region (as defined by the generated mathematical program) than the previous point.

```
GMP::Instance::FindApproximatelyFeasibleSolution(
  GMP,          ! (input) a generated mathematical program
  solution1,    ! (input) a solution
  solution2,    ! (input) a solution
  nrIter,       ! (output) a scalar numerical parameter
  [maxIter],    ! (optional) a scalar value
  [feasTol],    ! (optional) a scalar value
  [moveTol],    ! (optional) a scalar value
  [imprTol]     ! (optional) a scalar value
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

solution1

An integer scalar reference to a solution.

solution2

An integer scalar reference to a solution.

nrIter

The number of iterations used by the algorithm.

maxIter

The maximal number of iterations that can be used by the algorithm.
If its value is 0 (the default) then there is no iteration limit.

feasTol

The feasibility distance tolerance. The default is 1e-5.

moveTol

The movement tolerance. The default is 1e-5.

imprTol

The improvement tolerance. The default is 0.01.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The constraint consensus method is described in: John W. Chinneck, The Constraint Consensus Method for Finding Approximately Feasible Points in Nonlinear Programs, INFORMS Journal on Computing 16(3) (2004), pp. 255-265.
- The algorithm terminates if:
 - The iteration limit *maxIter* is exceeded.
 - The feasibility distance of each row is smaller than the feasibility distance tolerance *feasTol*. The feasibility distance of a row at a point is defined as the row violation normalized by the length of the gradient of the row at that point.
 - The length of the movement vector is smaller than the movement tolerance *moveTol*. The movement vector is the vector along which the point moves from one iteration to another.
 - The relative improvement was smaller than the improvement tolerance *imprTol* for 10 successive iterations. The improvement is defined as the difference between the length of the movement vector of the current iteration and that of the previous iteration.
- The procedure `GMP::Solution::Check` can be used to get the sum and number of infeasibilities before and after calling the procedure `GMP::Instance::FindApproximatelyFeasibleSolution`.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve` and `GMP::Solution::Check`.

GMP::Instance::FixColumns

The procedure `GMP::Instance::FixColumns` sets the lower and upper bounds of a set of columns in a generated mathematical program (*GMP1*) equal to the level values of the corresponding columns in a solution of a second generated mathematical program (*GMP2*).

```
GMP::Instance::FixColumns(
    GMP1,          ! (input) a generated mathematical program
    GMP2,          ! (input) a generated mathematical program
    solution,      ! (input) a solution
    variableSet,  ! (input) a set of variables
    [round]       ! (optional) a binary scalar value
)
```

Arguments:

GMP1

An element in `AllGeneratedMathematicalPrograms`.

GMP2

An element in `AllGeneratedMathematicalPrograms`.

solution

An integer scalar reference to a solution in the solution repository of *GMP2*.

variableSet

A subset of `AllVariables`.

round

A binary scalar indicating whether the level values of the integer columns should be rounded to the nearest integer value before fixing the columns. The default is 0 (no rounding).

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- A column corresponding to a variable in *variableSet* that is not part of *GMP1* will be ignored. This procedure will fail if a column corresponding to a variable in *variableSet* is not part of *GMP2*.
- If the objective variable is part of the set *variableSet* then it will be ignored, i.e., the objective variable will not be fixed.

See also:

The functions `GMP::Instance::CreateSolverSession` and `GMP::SolverSession::GetInstance`.

GMP::Instance::Generate

The function `GMP::Instance::Generate` generates a mathematical program instance from a symbolic mathematical program.

```
GMP::Instance::Generate(
  MP,           ! (input) a symbolic mathematical program
  [name]       ! (optional) a string expression
)
```

Arguments:

MP

A symbolic mathematical program in the set `AllMathematicalPrograms`. The mathematical program should have model type LP, MIP, QP, MIQP, QCP, MIQCP, NLP, MINLP, RMIP or RMINLP.

name

A string that holds the name for the mathematical program to be generated.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- If the second argument is not specified, or if it is the empty string, the name of the symbolic mathematical program is used to create a new element in the set `AllGeneratedMathematicalPrograms`.
- If an element with name specified by the *name* argument is already present in the set `AllGeneratedMathematicalPrograms` the corresponding generated mathematical program will be replaced (or updated in case the same symbolic mathematical program is involved). In that case all existing solver sessions created for the generated mathematical program will be deleted.
- It is possible to generate indexed mathematical program instances. See the example in Section [21.11.1](#) of the Language Reference.

See also:

The function `GMP::Instance::Delete`.

GMP::Instance::GenerateRobustCounterpart

The function `GMP::Instance::GenerateRobustCounterpart` generates the robust counterpart of a (linear) mathematical program.

If the deterministic model is a linear program (LP) then the robust counterpart will be a LP if the uncertainty constraints are linear, or a second-order cone program (SOCP) if some of the uncertainty constraints are ellipsoidal.

If the deterministic model is a mixed-integer program (MIP) then the robust counterpart will be a MIP if the uncertainty constraints are linear, or a mixed-integer second-order cone program (MISOCP) if some of the uncertainty constraints are ellipsoidal.

SOCP and MISOCP problems can be solved by using CPLEX or MOSEK.

```
GMP::Instance::GenerateRobustCounterpart(
    MP,                ! (input) a symbolic mathematical program
    UncertainParameters, ! (input) a set of uncertain parameters
    UncertaintyConstraints, ! (input) a set of uncertainty constraints
    [Name]             ! (optional) a string expression
)
```

Arguments:

MP

A symbolic mathematical program in the set `AllMathematicalPrograms`. The mathematical program should have model type LP or MIP.

UncertainParameters

A subset of `AllUncertainParameters`.

UncertaintyConstraints

A subset of `AllUncertaintyConstraints`.

Name

A string that holds the name for the generated robust counterpart.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- If the *Name* argument is not specified, or if it is the empty string, then the name of the symbolic mathematical program followed by 'robust counterpart' is used to create a new element in the set `AllGeneratedMathematicalPrograms`.

- If AIMMS detects that the robust counterpart is infeasible during the generation, AIMMS will issue a warning and the robust counterpart will not be generated.
- As part of the generation, AIMMS will check whether the uncertainty set satisfies the Slater condition (controlled by the option 'Slater condition check'). To do so, AIMMS will solve a linear program (LP) or a second-order cone program (SOCP).
- The created GMP cannot be modified, e.g., it is not allowed to change row or columns in the robust counterpart.

See also:

The procedure `GMP::Instance::Solve`.

GMP::Instance::GenerateStochasticProgram

The function `GMP::Instance::GenerateStochasticProgram` generates the deterministic equivalent of a stochastic mathematical program.

```
GMP::Instance::GenerateStochasticProgram(
  MP,                ! (input) a symbolic mathematical program
  StochasticParameters, ! (input) a set of stochastic parameters
  StochasticVariables, ! (input) a set of stochastic variables
  Scenarios,         ! (input) a set of stochastic scenarios
  ScenarioProbability, ! (input) a double parameter
  ScenarioTreeMap,   ! (input) an element parameter
  RootScenarioName,  ! (input) a string expression
  [GenerationMode], ! (optional) a stochastic generation mode
  [Name]             ! (optional) a string expression
)
```

Arguments:

MP

A symbolic mathematical program in the set `AllMathematicalPrograms`. The mathematical program should have model type LP or MIP.

StochasticParameters

A subset of `AllStochasticParameters`.

StochasticVariables

A subset of `AllStochasticVariables`.

Scenarios

A subset of `AllStochasticScenarios`.

ScenarioProbability

A double parameter over `Scenarios` representing the objective probabilities of the scenarios.

ScenarioTreeMap

An element parameter that defines the scenario-and-stage to scenario mapping. The range of this parameter should be the set `Scenarios`.

RootScenarioName

A string that holds the name of the artificial element that will be added to the set `AllStochasticScenarios`. This element will be used to store the solution of non-stochastic variables in their respective `.Stochastic` suffices.

GenerationMode

An element in the predefined set `AllStochasticGenerationModes`. The default is `'SubstituteStochasticVariables'`.

Name

A string that holds the name for the generated stochastic mathematical program.

Return value:

A new element in the set `AllGeneratedMathematicalPrograms` with the name as specified by the *name* argument.

Remarks:

- If the *Name* argument is not specified, or if it is the empty string, then the name of the symbolic mathematical program preceded by 'Stochastic' is used to create a new element in the set `AllGeneratedMathematicalPrograms`.
- The set `AllStochasticGenerationModes` contains the elements 'CreateNonAnticipativityConstraints' and 'SubstituteStochasticVariables'.
- The objective of the symbolic mathematical program must be a defined variable.

See also:

The procedure `GMP::Instance::Solve`.

GMP::Instance::GetDirection

The function `GMP::Instance::GetDirection` returns the optimization direction of a generated mathematical program.

```
GMP::Instance::GetDirection(  
  GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the optimization direction as an element in `AllMatrixManipulationDirections`.

See also:

The routines `GMP::Instance::Generate` and the procedure `GMP::Instance::SetDirection`.

GMP::Instance::GetLinearObjective

The function `GMP::Instance::GetLinearObjective` returns the best known bound for a generated mathematical program.

```
GMP::Instance::GetLinearObjective(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in `AllGeneratedMathematicalPrograms`.

Return value:

In case of success, the function returns the best known bound. Otherwise it returns UNDF.

Remarks:

This function has only meaning for generated mathematical programs with model type MIP, MIQP or MIQCP.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetMathematicalProgrammingType` and `GMP::Instance::GetObjective`.

GMP::Instance::GetMathematicalProgrammingType

The function `GMP::Instance::GetMathematicalProgrammingType` returns the model type of a generated mathematical program.

```
GMP::Instance::GetMathematicalProgrammingType(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the model type as an element in `AllMatrixManipulationProgrammingTypes`.

See also:

The function `GMP::Instance::Generate` and the procedure `GMP::Instance::SetMathematicalProgrammingType`.

GMP::Instance::GetMemoryUsed

The function `GMP::Instance::GetMemoryUsed` returns for a generated mathematical program the amount of memory used by AIMMS to store it.

```
GMP::Instance::GetMemoryUsed(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in `AllGeneratedMathematicalPrograms`.

Return value:

The amount of megabytes used to store a generated mathematical program.

GMP::Instance::GetNumberOfColumns

The function `GMP::Instance::GetNumberOfColumns` returns the number of columns of a generated mathematical program.

```
GMP::Instance::GetNumberOfColumns(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of columns.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfRows` and `GMP::Instance::GetNumberOfNonzeros`.

GMP::Instance::GetNumberOfIndicatorRows

The function `GMP::Instance::GetNumberOfIndicatorRows` returns the number of indicator rows of a generated mathematical program.

```
GMP::Instance::GetNumberOfIndicatorRows(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of indicator rows.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::GetNumberOfRows`.

GMP::Instance::GetNumberOfIntegerColumns

The function `GMP::Instance::GetNumberOfIntegerColumns` returns the number of integer columns of a generated mathematical program.

```
GMP::Instance::GetNumberOfIntegerColumns(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of integer columns.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfColumns` and `GMP::Instance::GetNumberOfNonlinearColumns`.

GMP::Instance::GetNumberOfNonlinearColumns

The function `GMP::Instance::GetNumberOfNonlinearColumns` returns the number of nonlinear columns of a generated mathematical program.

```
GMP::Instance::GetNumberOfNonlinearColumns(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of nonlinear columns.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfColumns` and `GMP::Instance::GetNumberOfIntegerColumns`.

GMP::Instance::GetNumberOfNonlinearNonzeros

The function `GMP::Instance::GetNumberOfNonlinearNonzeros` returns the number of nonlinear nonzero elements in the coefficient matrix of a generated mathematical program.

```
GMP::Instance::GetNumberOfNonlinearNonzeros(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of nonlinear nonzeros.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::GetNumberOfNonzeros`.

GMP::Instance::GetNumberOfNonlinearRows

The function `GMP::Instance::GetNumberOfNonlinearRows` returns the number of nonlinear rows of a generated mathematical program.

```
GMP::Instance::GetNumberOfNonlinearRows(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of nonlinear rows.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::GetNumberOfRows`.

GMP::Instance::GetNumberOfNonzeros

The function `GMP::Instance::GetNumberOfNonzeros` returns the number of nonzero elements in the coefficient matrix of a generated mathematical program.

```
GMP::Instance::GetNumberOfNonzeros(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of nonzeros.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfColumns` and `GMP::Instance::GetNumberOfRows`.

GMP::Instance::GetNumberOfRows

The function `GMP::Instance::GetNumberOfRows` returns the number of rows of a generated mathematical program.

```
GMP::Instance::GetNumberOfRows(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of rows.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfColumns` and `GMP::Instance::GetNumberOfNonzeros`.

GMP::Instance::GetNumberOfSOS1Rows

The function `GMP::Instance::GetNumberOfSOS1Rows` returns the number of SOS rows of type 1 of a generated mathematical program.

```
GMP::Instance::GetNumberOfSOS1Rows(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of SOS rows of type 1.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfRows` and `GMP::Instance::GetNumberOfSOS2Rows`.

GMP::Instance::GetNumberOfSOS2Rows

The function `GMP::Instance::GetNumberOfSOS2Rows` returns the number of SOS rows of type 2 of a generated mathematical program.

```
GMP::Instance::GetNumberOfSOS2Rows(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the number of SOS rows of type 2.

See also:

The functions `GMP::Instance::Generate`, `GMP::Instance::GetNumberOfRows` and `GMP::Instance::GetNumberOfSOS1Rows`.

GMP::Instance::GetObjective

The function `GMP::Instance::GetObjective` returns the current objective function value of a generated mathematical program.

```
GMP::Instance::GetObjective(  
  GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in `AllGeneratedMathematicalPrograms`.

Return value:

In case of success, the function returns the current objective function value. Otherwise it returns UNDF.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve` and `GMP::Instance::GetLinearObjective`.

GMP::Instance::GetOptionValue

The function `GMP::Instance::GetOptionValue` returns the value of a solver specific option corresponding to a generated mathematical program as set with the procedure `GMP::Instance::SetOptionValue`.

```
GMP::Instance::GetOptionValue(
    GMP,          ! (input) a generated mathematical program
    OptionName    ! (input) a scalar string expression
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

OptionName

A string expression holding the name of the option.

Return value:

In case of success, the function returns the current option value. Otherwise it returns UNDF.

Remarks:

- If the procedure `GMP::Instance::SetOptionValue` has not been called then this function will fail and return UNDF.
- Options for which strings are displayed in the AIMMS **Options** dialog box, are also represented by numerical (integer) values. To obtain the corresponding option keywords, you can use the functions `OptionGetString` and `OptionGetKeywords`.

See also:

The routines `GMP::Instance::SetOptionValue`, `GMP::SolverSession::GetOptionValue`, `GMP::SolverSession::SetOptionValue`, `OptionGetString` and `OptionGetKeywords`.

GMP::Instance::GetSolver

The function `GMP::Instance::GetSolver` returns for a generated mathematical program the solver that is assigned to it.

```
GMP::Instance::GetSolver(  
  GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the solver as an element of `AllSolvers`.

Remarks:

The solver can be assigned by the procedure `GMP::Instance::SetSolver`, or derived by `AIMMS` as the default solver for the model class of the generated mathematical program.

See also:

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

GMP::Instance::GetSymbolicMathematicalProgram

The function `GMP::Instance::GetSymbolicMathematicalProgram` returns for a generated mathematical program the originating symbolic mathematical program.

```
GMP::Instance::GetSymbolicMathematicalProgram(  
  GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Return value:

The function returns the symbolic mathematical program as an element of `AllMathematicalPrograms`.

See also:

The function `GMP::Instance::Generate`.

GMP::Instance::MemoryStatistics

With the procedure `GMP::Instance::MemoryStatistics` you can obtain a report containing the statistics collected by AIMMS' memory manager for a single or multiple generated mathematical programs.

```
GMP::Instance::MemoryStatistics(
  gmpSet,          ! (input) a set of generated mathematical programs
  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:

gmpSet

A subset of `AllGeneratedMathematicalPrograms` with generated mathematical programs 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 `GMP::Instance::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 `GMP::Instance::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 `GMP::Instance::MemoryStatistics`.

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 `GMP::Instance::MemoryStatistics`.
- AIMMS will only collect memory statistics if the option `memory_statistics` is on.

GMP::Instance::Rename

The procedure `GMP::Instance::Rename` can be used to rename a generated mathematical program.

```
GMP::Instance::Rename(  
  GMP,          ! (input) a generated mathematical program  
  Name         ! (input) a string expression  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

Name

A string that holds the new name.

Return value:

`GMP::Instance::Rename` has no return value.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::Copy`.

GMP::Instance::SetCallbackAddCut

The procedure `GMP::Instance::SetCallbackAddCut` installs a callback procedure adding cuts during the solution process of a MIP model.

```
GMP::Instance::SetCallbackAddCut(
    GMP,          ! (input) a generated mathematical program
    callback     ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The procedure `GMP::SolverSession::GenerateCut` can be used inside a `CallbackAddCut` callback procedure to add cuts during the MIP branch & cut process.
- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackAddCut` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- A `CallbackAddCut` callback procedure will only be called when solving mixed integer programs with CPLEX 8.0 or higher, with Xpress 17 or higher, or with GUROBI 2.0 or higher.
- This procedure can also be used for MIQP and MIQCP problems.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Instance::SetCallbackNewIncumbent`, `GMP::SolverSession::GenerateBinaryEliminationRow` and `GMP::SolverSession::GenerateCut`.

GMP::Instance::SetCallbackAddLazyConstraint

The procedure `GMP::Instance::SetCallbackAddLazyConstraint` installs a callback procedure for adding lazy constraints during the solution process of a MIP model.

```
GMP::Instance::SetCallbackAddLazyConstraint(
    GMP,          ! (input) a generated mathematical program
    callback     ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The callback procedure is called by the solver in these situations
 - when the solver compares an integer-feasible solution (including an integer-feasible solution provided by a MIP start before any nodes exist) to lazy constraints;
 - when the LP at a node is unbounded, and a lazy constraint might cut off the primal ray.
- The procedure `GMP::SolverSession::GenerateCut` can be used inside a `CallbackAddLazyConstraint` callback procedure to add (globally or locally valid) lazy constraints during the MIP branch & cut process. Lazy constraints added to the problem are first put into a pool of lazy constraints, so they are not present in the subproblem LP until after the callback is finished.
- If lazy constraints have been added, the subproblem is re-solved and evaluated, and, if the LP solution is still integer feasible and not cut off, the lazy constraint callback is called again.
- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackAddLazyConstraint` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- A `CallbackAddLazyConstraint` callback procedure will only be called when solving mixed integer programs with CPLEX 12.3 or higher.

- This procedure can also be used for MIQP and MIQCP problems.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Instance::SetCallbackNewIncumbent` and `GMP::SolverSession::GenerateCut`.

GMP::Instance::SetCallbackBranch

The procedure `GMP::Instance::SetCallbackBranch` installs a callback procedure to be called after a branch has been selected but before the branch is carried out during the MIP optimization. In the callback routine, the branch selected by the solver can be changed to a user-selected branch.

```
GMP::Instance::SetCallbackBranch(
    GMP,           ! (input) a generated mathematical program
    callback      ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- This callback is not called when the subproblem is infeasible.
- In the callback procedure at most 2 branches can be specified.
- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackBranch` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- The `CallbackBranch` callback procedure cannot be used to get the column on which the solver will branch.
- A `CallbackBranch` callback procedure will only be called when solving mixed integer programs with CPLEX 9.1 or higher.

See also:

The routines `GMP::Solution::RetrieveFromSolverSession`, `GMP::Solution::SendToModel`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::SendToSolverSession`, `GMP::SolverSession::GenerateBranchLower`, `GMP::SolverSession::GenerateBranchUpperBound`, `GMP::SolverSession::GenerateBranchRow`, `GMP::SolverSession::GetNumberOfBranchNodes`, `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent` and `GMP::Instance::SetCallbackNewIncumbent`.

GMP::Instance::SetCallbackHeuristic

The procedure `GMP::Instance::SetCallbackHeuristic` installs a callback procedure that is called during the solution process of a MIP model every time the subproblem has been solved to optimality.

```
GMP::Instance::SetCallbackHeuristic(
    GMP,           ! (input) a generated mathematical program
    callback      ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- This callback is not called when the subproblem is infeasible or cut off.
- The callback should supply the solver with a heuristically-derived integer solution.
- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackHeuristic` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- A `CallbackHeuristic` callback procedure will only be called when solving mixed integer programs with CPLEX 8.0 or higher or with GUROBI 2.0 or higher.

See also:

The routines `GMP::Solution::RetrieveFromSolverSession`, `GMP::Solution::SendToModel`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::SendToSolverSession`, `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackIncumbent` and `GMP::Instance::SetCallbackNewIncumbent`.

GMP::Instance::SetCallbackIncumbent

The procedure `GMP::Instance::SetCallbackIncumbent` installs a callback procedure that is called every time an incumbent solution is found during the solution process of a MIP model. By using the procedure `GMP::SolverSession::RejectIncumbent` the incumbent solution can be rejected. If `GMP::SolverSession::RejectIncumbent` is not called inside the `CallbackIncumbent` callback procedure then the incumbent solution will be accepted and replace the best incumbent solution found by so far.

```
GMP::Instance::SetCallbackIncumbent(
    GMP,           ! (input) a generated mathematical program
    callback      ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackIncumbent` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- If a `CallbackNewIncumbent` callback procedure is installed by using the procedure `GMP::Instance::SetCallbackNewIncumbent`, then that callback will be called after the `CallbackIncumbent` callback procedure if the incumbent solution is not rejected inside `CallbackIncumbent`.
- A `CallbackIncumbent` callback procedure will only be called when solving mixed integer programs with CPLEX 8.0 or higher, or with XPRESS 19 or higher.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyCons`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackNewIncumbent` and `GMP::SolverSession::RejectIncumbent`.

GMP::Instance::SetCallbackIterations

The procedure `GMP::Instance::SetCallbackIterations` installs a callback procedure that is called after a specified number of iterations.

```
GMP::Instance::SetCallbackIterations(
  GMP,           ! (input) a generated mathematical program
  callback,     ! (input) an AIMMS procedure
  [value]       ! (optional) number of iterations
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

value

A scalar value indicating after which number of iterations the callback procedure should be called. The default value is 0.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackIterations` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.
- The number of iterations can also be set by the `CallbackIterations` suffix of the symbolic mathematical program, but will be overruled if the *value* is not equal to 0.
- During the MIP phase of a MIP solve, the iterations callback might be called less often.
- The iterations callback will be called less often if CPLEX uses dynamic search as the MIP Search Strategy instead of branch-and-cut.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Instance::SetCallbackNewIncumbent` and `GMP::Instance::SetCallbackStatusChange`.

GMP::Instance::SetCallbackNewIncumbent

The procedure `GMP::Instance::SetCallbackNewIncumbent` installs a callback procedure that is called every time a new incumbent solution is found during the solution process of a MIP model.

```
GMP::Instance::SetCallbackNewIncumbent(
    GMP,          ! (input) a generated mathematical program
    callback      ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackNewIncumbent` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyCons`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Instance::SetCallbackIterations` and `GMP::Instance::SetCallbackStatusChange`.

GMP::Instance::SetCallbackStatusChange

The procedure `GMP::Instance::SetCallbackStatusChange` installs a callback procedure that is called every time the status changes during the solution process.

```
GMP::Instance::SetCallbackStatusChange(
  GMP,          ! (input) a generated mathematical program
  callback      ! (input) an AIMMS procedure
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

callback

A reference to a procedure in the set `AllIdentifiers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

- The callback procedure should have exactly one argument; a scalar input element parameter into the set `AllSolverSessions`.
- The `CallbackStatusChange` callback procedure should have a return value of
 - 0, if you want the solution process to stop, or
 - 1, if you want the solution process to continue.
- To remove the callback the empty element should be used as the *callback* argument.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Instance::SetCallbackIterations` and `GMP::Instance::SetCallbackNewIncumbent`.

GMP::Instance::SetCPUSecondsLimit

The procedure `GMP::Instance::SetCPUSecondsLimit` limits the amount of CPU seconds to solve a generated mathematical program.

```
GMP::Instance::SetCPUSecondsLimit(  
    GMP,          ! (input) a generated mathematical program  
    seconds      ! (input) number of seconds  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

seconds

Maximum number of seconds available to solve the generated mathematical program.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::Instance::SetCutoff`, `GMP::Instance::SetIterationLimit` and `GMP::Instance::SetMemoryLimit`.

GMP::Instance::SetCutoff

The procedure `GMP::Instance::SetCutoff` specifies a cutoff value that is used during the solution process of the generated mathematical program.

```
GMP::Instance::SetCutoff(  
    GMP,          ! (input) a generated mathematical program  
    cutoff       ! (input) scalar numerical expression  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

cutoff

A scalar value.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

This procedure is only used for MIP models.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::Instance::SetCPUSecondsLimit`, `GMP::Instance::SetIterationLimit` and `GMP::Instance::SetMemoryLimit`.

GMP::Instance::SetDirection

The procedure `GMP::Instance::SetDirection` changes the direction of a generated mathematical program.

```
GMP::Instance::SetDirection(  
    GMP,           ! (input) a generated mathematical program  
    direction      ! (input) an optimization direction  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

direction

An element expression in the set `AllMatrixManipulationDirections`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::GetDirection`.

GMP::Instance::SetIterationLimit

The procedure `GMP::Instance::SetIterationLimit` limits the number of iterations that can be used to solve a generated mathematical program.

```
GMP::Instance::SetIterationLimit(  
    GMP,          ! (input) a generated mathematical program  
    iterations    ! (input) number of iterations  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

iterations

Maximum number of iterations allowed to solve the generated mathematical program.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::Instance::SetCutoff`, `GMP::Instance::SetCPUSecondsLimit` and `GMP::Instance::SetMemoryLimit`.

GMP::Instance::SetMathematicalProgrammingType

The procedure `GMP::Instance::SetMathematicalProgrammingType` changes the type of a generated mathematical program from MIP into RMIP (or vice versa), or from MINLP to RMINLP (or vice versa). Also the type can be changed from MIQP or MIQCP to RMINLP or from MIP to LP, but not vice versa.

```
GMP::Instance::SetMathematicalProgrammingType(  
    GMP,                                ! (input) a generated mathematical program  
    MathematicalProgrammingType       ! (input) a model type  
)
```

Arguments:

GMP

An element in the set `AllGeneratedMathematicalPrograms`.

MathematicalProgrammingType

One of the elements LP, MIP, RMIP, MINLP or RMINLP (in the set `AllMatrixManipulationProgrammingTypes`).

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The functions `GMP::Instance::Generate` and `GMP::Instance::GetMathematicalProgrammingType`.

GMP::Instance::SetMemoryLimit

The procedure `GMP::Instance::SetMemoryLimit` limits the amount of memory available to solve a generated mathematical program.

```
GMP::Instance::SetMemoryLimit(  
  GMP,          ! (input) a generated mathematical program  
  memory       ! (input) amount of memory  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

memory

Maximum number of megabytes available to solve the generated mathematical program.

Return value:

The procedure returns 1 on success, or 0 otherwise.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::Solve`, `GMP::Instance::SetCutoff`, `GMP::Instance::SetCPUSecondsLimit` and `GMP::Instance::SetIterationLimit`.

GMP::Instance::SetOptionValue

The procedure `GMP::Instance::SetOptionValue` sets the value of a solver specific option corresponding to a generated mathematical program.

```
GMP::Instance::SetOptionValue(
  GMP,          ! (input) a generated mathematical program
  OptionName,   ! (input) a scalar string expression
  Value         ! (input) a scalar numeric expression
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

OptionName

A string expression holding the name of the option.

Value

A scalar numeric expression representing the new value to be assigned to the option.

Return value:

The procedure returns 1 if the option exists and the value can be assigned to the option, or 0 otherwise.

Remarks:

- All solvers solving the generated mathematical program will use the option value as set by this procedure (provided that the solver contains the option).
- This procedure will be overruled by the procedure `GMP::SolverSession::SetOptionValue` in case a solver session is used to solve the generated mathematical program.
- Options for which strings are displayed in the AIMMS **Options** dialog box, are also represented by numerical (integer) values. To obtain the corresponding option keywords, you can use the functions `OptionGetString` and `OptionGetKeywords`.

See also:

The routines `GMP::Instance::GetOptionValue`, `GMP::SolverSession::GetOptionValue`, `GMP::SolverSession::SetOptionValue`, `OptionGetString` and `OptionGetKeywords`.

GMP::Instance::SetSolver

The procedure `GMP::Instance::SetSolver` can be used to select for a generated mathematical program the solver to be called in subsequent calls to `GMP::Instance::Solve`.

```
GMP::Instance::SetSolver(  
  GMP,          ! (input) a generated mathematical program  
  solver        ! (input) a solver  
)
```

Arguments:

GMP

An element in `AllGeneratedMathematicalPrograms`.

solver

An element in the set `AllSolvers`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

The solver set in this procedure will also be assigned to any solver session created with the function `GMP::Instance::CreateSolverSession` for the *GMP*, unless the *Solver* argument in the procedure `GMP::Instance::CreateSolverSession` is specified. Note that the procedure `GMP::Instance::SetSolver` cannot be used to change the solver assigned to a solver session after `GMP::Instance::CreateSolverSession` has been called.

See also:

The routines `GMP::Instance::CreateSolverSession`, `GMP::Instance::Generate`, `GMP::Instance::GetSolver` and `GMP::Instance::Solve`.

GMP::Instance::Solve

The procedure `GMP::Instance::Solve` starts up a solver session to solve a generated mathematical program. In addition, it copies the initial solution from the model identifiers via solution 1 in the solution repository and stores the final solution via solution 1 back in the model identifiers.

```
GMP::Instance::Solve(  
    GMP          ! (input) a generated mathematical program  
)
```

Arguments:

GMP
An element in `AllGeneratedMathematicalPrograms`.

Return value:

The procedure returns 1 on success, or 0 otherwise.

Remarks:

The procedure `GMP::Instance::Solve` automatically creates a solver session with the same name as the generated mathematical program in the set `AllSolverSessions`.

See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::CreateSolverSession`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::SendToSolverSession`, `GMP::SolverSession::Execute`, `GMP::Solution::RetrieveFromSolverSession` and `GMP::Solution::SendToModel`.