

---

**AIMMS Function Reference - GMPSolution 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::Solution Procedures and Functions

AIMMS supports the following procedures and functions for creating and managing solutions in the solution repository associated with a generated mathematical program instance:

- GMP::Solution::Check
- GMP::Solution::ConstructMean
- GMP::Solution::Copy
- GMP::Solution::Count
- GMP::Solution::Delete
- GMP::Solution::DeleteAll
- GMP::Solution::GetColumnValue
- GMP::Solution::GetCPUSecondsUsed
- GMP::Solution::GetDistance
- GMP::Solution::GetFirstOrderDerivative
- GMP::Solution::GetIterationsUsed
- GMP::Solution::GetLinearObjective
- GMP::Solution::GetMemoryUsed
- GMP::Solution::GetNodesUsed
- GMP::Solution::GetObjective
- GMP::Solution::GetPenalizedObjective
- GMP::Solution::GetProgramStatus
- GMP::Solution::GetRowValue
- GMP::Solution::GetSolutionsSet
- GMP::Solution::GetSolverStatus
- GMP::Solution::IsDualDegenerated
- GMP::Solution::IsInteger
- GMP::Solution::IsPrimalDegenerated
- GMP::Solution::Move
- GMP::Solution::RandomlyGenerate
- GMP::Solution::RetrieveFromModel
- GMP::Solution::RetrieveFromSolverSession
- GMP::Solution::SendToModel
- GMP::Solution::SendToModelSelection
- GMP::Solution::SendToSolverSession
- GMP::Solution::SetIterationCount
- GMP::Solution::SetMIPStartFlag
- GMP::Solution::SetProgramStatus

- `GMP::Solution::SetSolverStatus`
- `GMP::Solution::UpdatePenaltyWeights`

See also the section on Managing the solution repository, Section [21.4](#) of the language reference.

---

## GMP::Solution::Check

The procedure `GMP::Solution::Check` checks the validity of a solution for a generated mathematical program.

```
GMP::Solution::Check(
  GMP,          ! (input) a generated mathematical program
  solution,     ! (input) a solution
  numInfeas,   ! (output) number of infeasibilities
  sumInfeas,   ! (output) sum of infeasibilities
  [skipObj]    ! (optional, default 0) a scalar value
)
```

### Arguments:

*GMP*

An element in the set `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*numInfeas*

Number of infeasibilities for the solution.

*sumInfeas*

Sum of all infeasibilities for the solution.

*skipObj*

A scalar binary value to indicate whether constraints containing the objective variable should be skipped (value 1) or not (value 0).

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::RetrieveFromModel` and `GMP::Solution::RetrieveFromSolverSession`.

---

## GMP::Solution::ConstructMean

The procedure `GMP::Solution::ConstructMean` constructs the weighted average of two solutions of a generated mathematical program by using the column level values in both solutions. The first solution is replaced by the resulting mean solution.

```
GMP::Solution::ConstructMean(  
  GMP,          ! (input) a generated mathematical program  
  solution1,    ! (input) a solution  
  solution2,    ! (input) a solution  
  weight        ! (input) 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.

*weight*

The weight used for *solution1*.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

The *weight* argument defines the weight used for *solution1*; for *solution2* a weight of 1 is used. The constructed mean solution is divided by  $(weight+1)$ , and placed in *solution1*.

---

## GMP::Solution::Copy

The procedure `GMP::Solution::Copy` copies one solution to another solution in the solution repository of a generated mathematical program.

```
GMP::Solution::Copy(  
  GMP,           ! (input) a generated mathematical program  
  fromSolution, ! (input) a solution  
  toSolution     ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*fromSolution*

An integer scalar reference to a solution.

*toSolution*

An integer scalar reference to a solution.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Solution::Move`.

---

## GMP::Solution::Count

The function `GMP::Solution::Count` returns the number of non-empty solutions in the solution repository of a generated mathematical program.

```
GMP::Solution::Count(  
  GMP          ! (input) a generated mathematical program  
)
```

### Arguments:

*GMP*  
An element in `AllGeneratedMathematicalPrograms`.

### Return value:

The number of non-empty solutions stored in the solution repository.

### See also:

The functions `GMP::Instance::Generate` and `GMP::Solution::GetSolutionsSet`.

---

**GMP::Solution::Delete**

The procedure `GMP::Solution::Delete` deletes a solution from the solution repository of a generated mathematical program.

```
GMP::Solution::Delete(  
  GMP,          ! (input) a generated mathematical program  
  solution      ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**See also:**

The routines `GMP::Instance::Generate` and `GMP::Solution::DeleteAll`.

---

**GMP::Solution::DeleteAll**

The procedure `GMP::Solution::DeleteAll` empties the solution repository of a generated mathematical program.

```
GMP::Solution::DeleteAll(  
  GMP          ! (input) a generated mathematical program  
)
```

**Arguments:**

*GMP*  
An element in `AllGeneratedMathematicalPrograms`.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**See also:**

The routines `GMP::Instance::Generate` and `GMP::Solution::Delete`.

---

## GMP::Solution::GetColumnValue

The function `GMP::Solution::GetColumnValue` returns the level value of a column in a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetColumnValue(  
    GMP,           ! (input) a generated mathematical program  
    solution,     ! (input) a solution  
    column       ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The level value of the column.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Solution::GetRowValue`.

---

## GMP::Solution::GetCPUSecondsUsed

The function `GMP::Solution::GetCPUSecondsUsed` returns the number of CPU seconds used to create a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetCPUSecondsUsed(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The number of 1/100th seconds used to create a solution.

### See also:

The procedure `GMP::Instance::SetCPUSecondsLimit`.

---

**GMP::Solution::GetDistance**

The function `GMP::Solution::GetDistance` calculates the Euclidean distance between the vectors of column level values in a first and second solution of a generated mathematical program.

```
GMP::Solution::GetDistance(  
    GMP,          ! (input) a generated mathematical program  
    solution1,    ! (input) a solution  
    solution2     ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution1*

An integer scalar reference to a solution.

*solution2*

An integer scalar reference to a solution.

**Return value:**

In case of success, the Euclidean distance between both solutions. Otherwise it returns UNDF.

**Remarks:**

The level value of the objective column (if any) is not used.

---

## GMP::Solution::GetFirstOrderDerivative

The function `GMP::Solution::GetFirstOrderDerivative` returns the first order derivative for a column in a row in a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetFirstOrderDerivative(
    GMP,          ! (input) a generated mathematical program
    solution,     ! (input) a solution
    row,         ! (input) a scalar reference
    column       ! (input) a scalar reference
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*row*

A scalar reference to an existing row in the matrix.

*column*

A scalar reference to an existing column in the matrix.

### Return value:

The first order derivative of the column in the row.

### Remarks:

If this function is called for multiple rows and columns, then AIMMS will calculate the first order derivatives more efficiently if this function is called row wise instead of column wise. That is, it is better to call this function for all columns in a certain row before calling it for the next row.

### See also:

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

---

## GMP::Solution::GetIterationsUsed

The function `GMP::Solution::GetIterationsUsed` returns the number of iterations used to create a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetIterationsUsed(  
  GMP,          ! (input) a generated mathematical program  
  solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The number of iterations used to create a solution.

### See also:

The procedures `GMP::Instance::SetIterationLimit` and `GMP::Solution::SetIterationCount`.

---

**GMP::Solution::GetLinearObjective**

The function `GMP::Solution::GetLinearObjective` returns the the best known bound on a solution.

```
GMP::Solution::GetLinearObjective(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

**Return value:**

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

**Remarks:**

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

**See also:**

The procedure `GMP::Solution::GetObjective`.

---

## GMP::Solution::GetMemoryUsed

The function `GMP::Solution::GetMemoryUsed` returns the amount of memory used to create a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetMemoryUsed(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The amount of megabytes used to create a solution.

### See also:

The procedure `GMP::Instance::SetMemoryLimit`.

---

## GMP::Solution::GetNodesUsed

The function `GMP::Solution::GetNodesUsed` returns the number of nodes used to create a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetNodesUsed(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The number of nodes used to create a solution.

### Remarks:

- This function has only meaning for solver sessions belonging to a GMP with type MIP, MIQP or MIQCP.
- This function can be used inside a **cuts**, **heuristic** or **incumbent** callback.

### See also:

The routines `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackHeuristic` and `GMP::Instance::SetCallbackIncumbent`.

---

**GMP::Solution::GetObjective**

The function `GMP::Solution::GetObjective` retrieves the objective function value of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetObjective(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

**Return value:**

The objective function value of the solution.

**Remarks:**

The objective function value is only available if the solution has been retrieved from the solver.

**See also:**

The functions `GMP::Instance::Generate`, `GMP::Solution::GetProgramStatus` and `GMP::Solution::GetSolverStatus`.

---

## GMP::Solution::GetPenalizedObjective

The function `GMP::Solution::GetPenalizedObjective` calculates the penalized objective for a generated mathematical program by using the level values of the columns in a first solution and the shadow prices in a second solution as the penalty multipliers for the rows. To avoid a very large value, the penalized objective value is divided by the square of the number of rows.

```
GMP::Solution::GetPenalizedObjective(
    GMP,          ! (input) a generated mathematical program
    solution1,   ! (input) a solution
    solution2    ! (input) a solution
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution1*

An integer scalar reference to a solution.

*solution2*

An integer scalar reference to a solution.

### Return value:

In case of success, the penalized objective function value of the *GMP* associated with both solutions. Otherwise it returns UNDF.

### Remarks:

Assume that  $x$  denotes the level values of the columns in *solution1* and  $w$  the shadow prices of the rows in *solution2*. Then the penalized objective function  $P(x, w)$  is defined as

$$P(x, w) = f(x) + \sum_{i=1}^m (w_i * viol(g_i(x))),$$

where  $f(x)$  denotes the objective function value,  $m$  is the number of rows and the function  $viol(g_i(x))$  equals the absolute amount by which the  $i$ th row is violated at the point  $x$ .

### See also:

The procedure `GMP::Solution::UpdatePenaltyWeights`.

---

## GMP::Solution::GetProgramStatus

The function `GMP::Solution::GetProgramStatus` retrieves the program status of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetProgramStatus(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

An element in the set `AllSolutionStates`.

### Remarks:

The program status is only available if the solution has been retrieved from the solver, or if the procedure `GMP::Solution::SetProgramStatus` has been called before.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::GetSolverStatus`, `GMP::Solution::GetObjective` and `GMP::Solution::SetProgramStatus`.

---

## GMP::Solution::GetRowValue

The function `GMP::Solution::GetRowValue` returns the level value of a row in a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetRowValue(  
    GMP,           ! (input) a generated mathematical program  
    solution,     ! (input) a solution  
    row           ! (input) a scalar reference  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*row*

A scalar reference to an existing row in the matrix.

### Return value:

The level value of the row.

### See also:

The routines `GMP::Instance::Generate` and `GMP::Solution::GetColumnValue`.

---

**GMP::Solution::GetSolutionsSet**

The function `GMP::Solution::GetSolutionsSet` returns the set of non-empty solutions in the solution repository of a generated mathematical program.

```
GMP::Solution::GetSolutionsSet(  
    GMP          ! (input) a generated mathematical program  
)
```

**Arguments:**

*GMP*  
An element in `AllGeneratedMathematicalPrograms`.

**Return value:**

A subset of Integers.

**See also:**

The functions `GMP::Instance::Generate` and `GMP::Solution::Count` and the section on Managing the solution repository Section [21.4](#) of the language reference.

---

**GMP::Solution::GetSolverStatus**

The function `GMP::Solution::GetSolverStatus` retrieves the solver status of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::GetSolverStatus(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

**Return value:**

An element in the set `AllSolutionStates`.

**Remarks:**

The solver status is only available if the solution has been retrieved from the solver, or if the procedure `GMP::Solution::SetSolverStatus` has been called before.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Solution::GetProgramStatus` and `GMP::Solution::GetObjective` and `GMP::Solution::SetSolverStatus`.

---

## GMP::Solution::IsDualDegenerated

The function `GMP::Solution::IsDualDegenerated` checks whether the solution for a generated mathematical program, with model type LP, RMIP or QP, is dual degenerated.

```
GMP::Solution::IsDualDegenerated(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in the set `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The function returns 1 if the solution is dual degenerated, and 0 otherwise.

### Remarks:

- A solution is dual degenerated if a non-basic variable has a zero marginal, or if a non-equality constraint is non-basic and has a zero marginal. In that case the primal solution is not unique.
- This function will always return 0 if the barrier algorithm (without crossover) of CPLEX was used to solve the problem because the barrier algorithm (without crossover) of CPLEX does not provide a basic solution.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::IsPrimalDegenerated` and `GMP::Solution::RetrieveFromSolverSession`.

---

## GMP::Solution::IsInteger

The function `GMP::Solution::IsInteger` checks whether the solution for a generated mathematical program is an integer solution.

```
GMP::Solution::IsInteger(  
    GMP,          ! (input) a generated mathematical program  
    solution,     ! (input) a solution  
    [tolerance]  ! (optional) a tolerance  
)
```

### Arguments:

*GMP*

An element in the set `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*tolerance*

A numerical value. The default is 0.

### Return value:

The function returns 1 if the solution is integer, and 0 otherwise.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::RetrieveFromModel` and `GMP::Solution::RetrieveFromSolverSession`.

---

## GMP::Solution::IsPrimalDegenerated

The function `GMP::Solution::IsPrimalDegenerated` checks whether the solution for a generated mathematical program, with model type LP, RMIP or QP, is primal degenerated.

```
GMP::Solution::IsPrimalDegenerated(  
    GMP,          ! (input) a generated mathematical program  
    solution      ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in the set `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The function returns 1 if the solution is primal degenerated, and 0 otherwise.

### Remarks:

- A solution is primal degenerated if a basic variable is at a bound, or if a non-equality constraint is basic and at a bound. In that case the dual solution is not unique.
- This function will always return 0 if the barrier algorithm (without crossover) of CPLEX was used to solve the problem because the barrier algorithm (without crossover) of CPLEX does not provide a basic solution.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::IsDualDegenerated` and `GMP::Solution::RetrieveFromSolverSession`.

---

**GMP::Solution::Move**

The procedure `GMP::Solution::Move` moves one solution to another solution in the solution repository of a generated mathematical program.

```
GMP::Solution::Move(  
  GMP,          ! (input) a generated mathematical program  
  fromSolution, ! (input) a solution  
  toSolution    ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*fromSolution*

An integer scalar reference to a solution.

*toSolution*

An integer scalar reference to a solution.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**Remarks:**

After calling this procedure, the solution at position *fromSolution* in the solution repository will be empty. This is not the case if you use the procedure `GMP::Solution::Copy`.

**See also:**

The routines `GMP::Instance::Generate` and `GMP::Solution::Copy`.

---

## GMP::Solution::RandomlyGenerate

The procedure `GMP::Solution::RandomlyGenerate` generates random level values in a solution for all columns in a generated mathematical program. Each level value is sampled from the uniform distribution by using the lower and upper bound of the column as parameters.

```
GMP::Solution::RandomlyGenerate(
  GMP,           ! (input) a generated mathematical program
  solution,      ! (input) a solution
  [maxVarBound], ! (optional) a scalar value
  [startPoint], ! (input) a solution
  [perturbation] ! (optional) a scalar value
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*maxVarBound*

The maximal variable bound. If a column has no upper bound then the sampled level value will be smaller than the maximal variable bound, and if a column has no lower bound then the sampled level value will be greater than minus the maximal variable bound. The default is 1000.

*startPoint*

An integer scalar reference to a solution representing a starting point. If specified then the sampled level value of a column will be around its level value in the starting point. By default no starting point is used.

*perturbation*

Used in combination with argument *startPoint*. A value between 0 and 1 that represents the (relative) perturbation around the starting point. The default is 0.1.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- This procedure should be called after calling the function `GMP::Instance::CreatePresolved` if it is used in combination with that function. Otherwise the sampled level values might be outside the range of the columns in the presolved model.

- If argument *startPoint* is specified then for each column the sampled value will be in the range

$$[x - p * (x - lb), x + p * (ub - x)]$$

where  $x$  denotes the level value of the column,  $lb$  and  $ub$  its lower and upper bound respectively, and  $p$  the *perturbation* value.

- *startPoint* cannot be equal to *solution*.

**See also:**

The function `GMP::Instance::CreatePresolved`.

---

## GMP::Solution::RetrieveFromModel

The procedure `GMP::Solution::RetrieveFromModel` stores the solution from the model identifiers into the solution repository of a generated mathematical program.

```
GMP::Solution::RetrieveFromModel(  
  GMP,           ! (input) a generated mathematical program  
  solution       ! (input) a solution  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

A solution vector in the solution repository only contains solution data for the generated columns and rows of the GMP. Hence, no solution data is stored in the solution repository for columns and rows that were not generated.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::SendToModel`, `GMP::Solution::RetrieveFromSolverSession` and `GMP::Solution::SendToSolverSession`.

---

## GMP::Solution::RetrieveFromSolverSession

The procedure `GMP::Solution::RetrieveFromSolverSession` stores the solution from a solver session into the solution repository of a generated mathematical program.

```
GMP::Solution::RetrieveFromSolverSession(
    solverSession, ! (input) a solver session
    solution      ! (input) a solution
)
```

### Arguments:

*solverSession*

An element in the set `AllSolverSessions`.

*solution*

An integer scalar reference to a solution.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- For a solver session belonging to a GMP with type MIP, this procedure retrieves the best integer solution found by so far (i.e., the incumbent), except when this procedure is called inside a **branch**, **cuts**, **heuristic** or **lazy constraint** callback. In that case this procedure retrieves the LP solution of the current node (**branch**, **cuts**, **heuristic**) or an integer feasible solution (**lazy constraint**).
- The function `GMP::SolverSession::GetNodeObjective` can be used to get the objective value corresponding to the solution retrieved with this procedure inside a **branch**, **cuts**, **heuristic**, **incumbent** or **lazy constraint** callback.
- By using the procedure `GMP::SolverSession::RejectIncumbent` the incumbent solution can be rejected inside an **incumbent** callback.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Instance::SetCallbackAddCut`, `GMP::Instance::SetCallbackAddLazyConstraint`, `GMP::Instance::SetCallbackBranch`, `GMP::Instance::SetCallbackHeuristic`, `GMP::Instance::SetCallbackIncumbent`, `GMP::Solution::SendToSolverSession`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::SendToModel`, `GMP::SolverSession::GetNodeObjective` and `GMP::SolverSession::RejectIncumbent`.

---

**GMP::Solution::SendToModel**

The procedure `GMP::Solution::SendToModel` initializes the model identifiers with the values in the solution from the solution repository of a generated mathematical program.

```
GMP::Solution::SendToModel(  
  GMP,           ! (input) a generated mathematical program  
  solution       ! (input) a solution  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**Remarks:**

A solution vector in the solution repository only contains solution data for the generated columns and rows of the GMP. Hence, no solution data is stored in the solution repository for columns and rows that were not generated.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::RetrieveFromSolverSession` and `GMP::Solution::SendToSolverSession`.

---

## GMP::Solution::SendToModelSelection

The procedure `GMP::Solution::SendToModelSelection` initializes a part of the model identifiers with the values in the solution from the solution repository of a generated mathematical program.

```
GMP::Solution::SendToModelSelection(
    GMP,          ! (input) a generated mathematical program
    solution,     ! (input) a solution
    Identifiers,  ! (input) a set expression
    Suffices      ! (input) a set expression
)
```

### Arguments:

#### *GMP*

An element in `AllGeneratedMathematicalPrograms`.

#### *solution*

An integer scalar reference to a solution.

#### *Identifiers*

A subset of the predefined set `AllVariablesConstraints`, containing the set of all variables and constraints for which the values have to be changed into those of *solution*.

#### *Suffices*

A subset of the predefined set `AllSuffixNames`, containing the set of suffices for which the values of *Identifiers* have to be changed into those of *solution*.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- If the subset *Identifiers* contains a variable or constraint that is not part of the generated mathematical program, then that variable or constraint will be ignored and its data will not change.
- If the subset *Suffices* contains a suffix other than 'Level', 'Basic', 'Reduced-Cost', 'ShadowPrice', 'SmallestCoefficient', 'NominalCoefficient', 'LargestCoefficient', 'SmallestValue', 'LargestValue', 'SmallestRightHandSide', 'NominalRightHandSide', 'LargestRightHandSide', 'SmallestShadowPrice' and 'LargestShadowPrice', then that suffix will be ignored and its data will not change.
- A solution vector in the solution repository only contains solution data for the generated columns and rows of the GMP. Hence, no solution data is stored in the solution repository for columns and rows that were not generated.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Solution::RetrieveFromModel`, `GMP::Solution::RetrieveFromSolverSession`, `GMP::Solution::SendToSolverSession` and `GMP::Solution::SendToModel`

---

## GMP::Solution::SendToSolverSession

The procedure `GMP::Solution::SendToSolverSession` initializes a solver session with the values in the solution from the solution repository of a generated mathematical program.

```
GMP::Solution::SendToSolverSession(  
    solverSession, ! (input) a solver session  
    solution      ! (input) a solution  
)
```

### Arguments:

*solverSession*

An element in the set `AllSolverSessions`.

*solution*

An integer scalar reference to a solution.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::RetrieveFromSolverSession`, `GMP::Solution::RetrieveFromModel` and `GMP::Solution::SendToModel`.

---

## GMP::Solution::SetIterationCount

The procedure `GMP::Solution::SetIterationCount` sets the iteration count of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::SetIterationCount(  
    GMP,          ! (input) a generated mathematical program  
    solution,     ! (input) a solution  
    iterationCount ! (input) iteration count  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*iterationCount*

An integer scalar.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The routines `GMP::Instance::Generate`, `GMP::SolverSession::GetIterationsUsed` and `GMP::Solution::SetProgramStatus`.

---

## GMP::Solution::SetMIPStartFlag

The procedure `GMP::Solution::SetMIPStartFlag` can be used to mark a solution in the solution repository of a generated mathematical program such that it should be used as a MIP start during the a MIP solve (or a MIQP or MIQCP solve).

```
GMP::Solution::SetMIPStartFlag(
    GMP,          ! (input) a generated mathematical program
    solution,     ! (input) a solution
    flag,        ! (input) a scalar value
    [effortLevel] ! (optional, default 0) a scalar value
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*flag*

A scalar binary value to indicate whether the solution should be marked (value 1) or unmarked (value 0) as MIP start.

*effortLevel*

A scalar value to specify the level of effort that the solver should apply to the solution when using it as MIP start solution. The default value of 0 indicates that the solver should decide; the other values are explained below.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

- The levels of effort and their effect as specified by argument *effortLevel* are:
  - Level 0: The solver decides.
  - Level 1: The solver checks feasibility of the corresponding MIP start.
  - Level 2: The solver solves the fixed LP problem specified by the MIP start.
  - Level 3: The solver solves a subMIP.
  - Level 4: The solver attempts to repair the MIP start if it is infeasible.
- This procedure is only supported by CPLEX 11.2 or higher.

**See also:**

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

---

## GMP::Solution::SetProgramStatus

The procedure `GMP::Solution::SetProgramStatus` sets the program status of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::SetProgramStatus(  
    GMP,          ! (input) a generated mathematical program  
    solution,     ! (input) a solution  
    status        ! (input) a status  
)
```

### Arguments:

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*status*

An element in the set `AllSolutionStates`.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### See also:

The routines `GMP::Instance::Generate`, `GMP::Solution::GetProgramStatus` and `GMP::Solution::SetSolverStatus`.

---

**GMP::Solution::SetSolverStatus**

The procedure `GMP::Solution::SetSolverStatus` sets the solver status of a solution in the solution repository of a generated mathematical program.

```
GMP::Solution::SetSolverStatus(  
  GMP,          ! (input) a generated mathematical program  
  solution,     ! (input) a solution  
  status       ! (input) a status  
)
```

**Arguments:**

*GMP*

An element in `AllGeneratedMathematicalPrograms`.

*solution*

An integer scalar reference to a solution.

*status*

An element in the set `AllSolutionStates`.

**Return value:**

The procedure returns 1 on success, or 0 otherwise.

**See also:**

The routines `GMP::Instance::Generate`, `GMP::Solution::GetSolverStatus` and `GMP::Solution::SetProgramStatus`.

---

## GMP::Solution::UpdatePenaltyWeights

The procedure `GMP::Solution::UpdatePenaltyWeights` updates the penalty weights which are stored as shadow prices in a first solution of a generated mathematical program. The shadow price of a row in this solution is compared with the shadow price of the same row in the second solution, and replaced by the maximum of both shadow prices.

```
GMP::Solution::UpdatePenaltyWeights(
    GMP,           ! (input) a generated mathematical program
    solution1,     ! (input) a solution
    solution2,     ! (input) a solution
    [minValue]    ! (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.

*minValue*

The minimum value for each shadow price. The default is 0.

### Return value:

The procedure returns 1 on success, or 0 otherwise.

### Remarks:

If for a certain row both the shadow prices in *solution1* and *solution2* are smaller than *minValue*, the new value assigned to the shadow price in *solution1* will be *minValue*.

### See also:

The function `GMP::Solution::GetPenalizedObjective`.