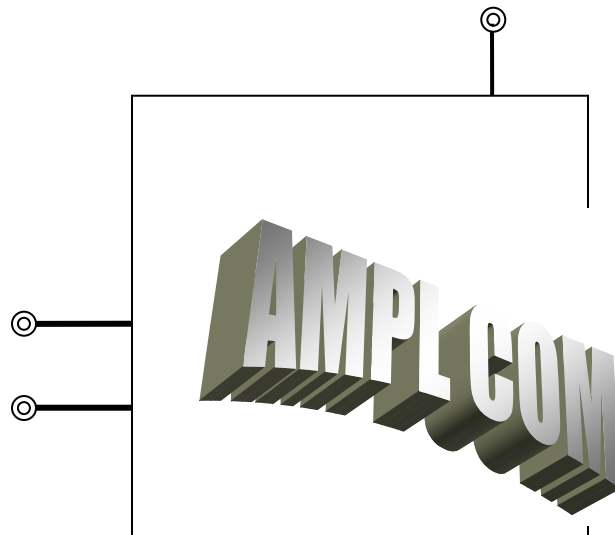


---

# AMPL COM Component Library

---

**User's Guide  
Version 1.6**



Draft

**Internal T.Report:**

Version: Draft1  
Start Date: 8/20/2005  
Revision Date: 11/8/2005  
Finish Date:  
Author: Mustapha Sadki  
Revision(s):

## Introduction

**Component Object Model, or COM**, is a specification and implementation developed by Microsoft Corporation which provides a framework for integrating components. Using COM enables developers and end users to integrate AMPL into a complete application solution.

**AMPL COM** contains within this framework an extensive set of object class (methods, properties, collections and objects), where each particular object provides implementations of functions for all the interfaces its class supports. Thereby it allows the integration of all AMPL modelling system features into an end-user application with embedded “Optimization Modelling & Solution” capabilities, using different programming platforms, such as *VBA* for *Excel/Access*, *Visual Basic*, *Visual C++/C#*, *Delphi*, *Java*, and standard scripting languages for the *Web*.

### Performing registration of AMPLCOM.dll

Before we can use the AMPLCOM.dll we need to register it within our system. To register or unregister such DLL, we need to run the windows command regsvr32 with the appropriate syntax:

Syntax

```
REGSVR32 [/U] [/S] [/C] [/I:[Command_Line]] DLL_Name
```

Options:

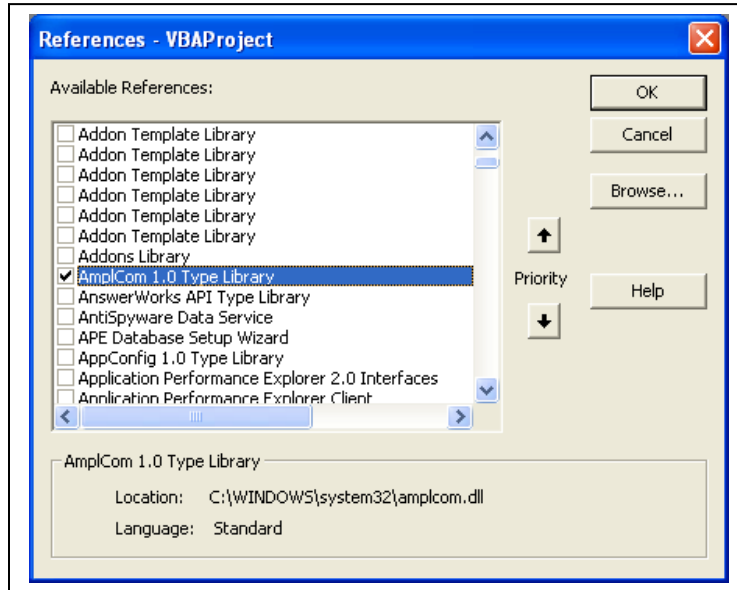
```
/u      Unregister Server.  
/s      Silent - no dialogue boxes.  
/c      Console output.  
/n      Don't call DllRegisterServer  
/i      Call DllInstall (or DllUninstall if /u is specified)  
Command_Line An optional command line for DllInstall
```

To register AMPLCOM .dll run :

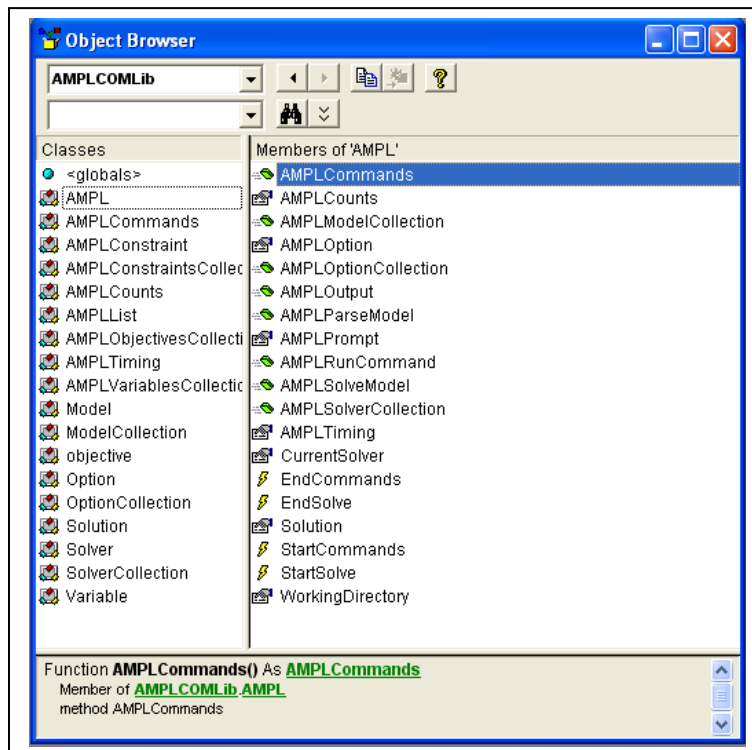
```
regsvr32 AmplCom.dll
```

## Adding reference to VB project

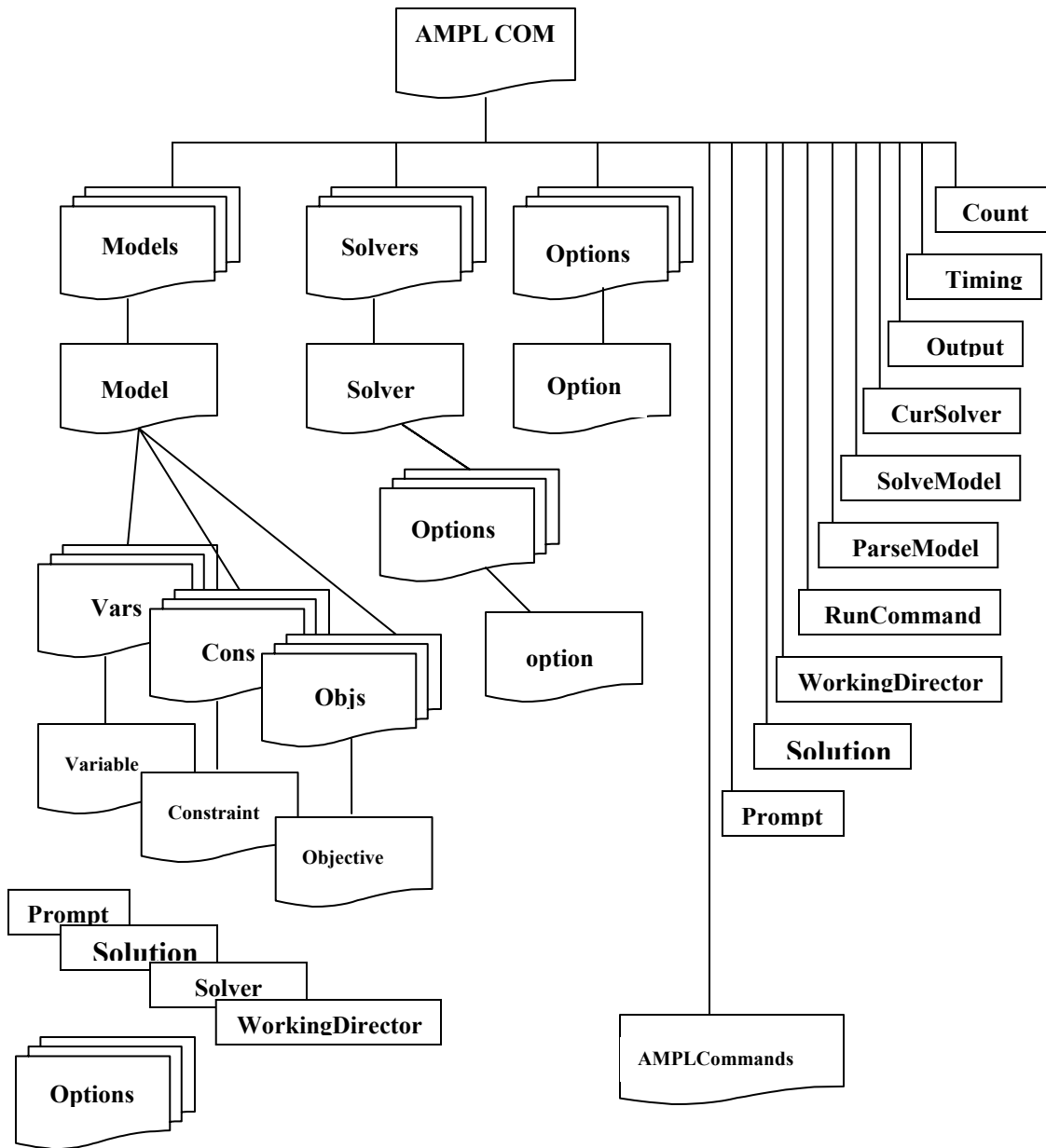
To be able to use the VB examples attached in this document we need to add reference from IDE VB to our project:



Also by making this reference we will be able to use the VB object browser to browse the AMPLCOMLib



# AMPL-COM Object Architecture



---

## The AMPL COM Object

The **AMPL COM** object is the main object in **AMPLCOM.dll** Library and is normally the first object that programmer has to instantiate in each project that uses **AMPL COM**. All other objects in the **AMPL** object hierarchy are reachable from the objects contained by the **AMPL** object and methods.

### Collections contained by the AMPL COM Object

#### **AMPLModelCollection()**

This method returns a Models Collection object, a collection which holds independent models

#### **AMPLSolverCollection**

The method returns a Solvers Collection object, which serves to store different solvers and/ or solvers with different option settings

### Methods of the AMPL COM Object

#### **AMPLOutput**

The method **AMPLOutput** returns the last AMPL output string as sent by AMPL after executing the last command

#### **AMPLSolveModel**

The method solves a given model Object and returns a Solution object

#### **AMPLRunCommand(CmdString)**

The method **AMPLRunCommand** allows to run AMPL commands like we do in AMPL command line. It allows also to enter the model, data and script directly in your code program instead of reading them from files.

#### **AMPLParseModel**

The method that parses the Model Model and returns 0 if success

#### **AMPLOptionCollection**

The method **AMPLOptionCollection** returns the **OptionCollection** object which stores user AMPL options

#### **AMPLCommands**

The method returns the **AMPLCommands** interface. This interface will work like the AMPL command line does.

## Properties of the AMPL COM Object

### **WorkingDirectory**

A string property containing the working directory for AMPL instance (read/write)

### **CurrentSolver**

A Solver object property, which allows to get and set the solver for the next solve command (read/write)

### **AMPLOption**

A String property to set and get AMPL option

### **AMPLPrompt**

A string property that contains the last prompt from AMPL (Read only)

### **AMPLTiming**

A Timing object property, to get AMPLTiming object

### **Solution**

A Solution object property, gets Solution object associated to the last AMPL solve command

### **AMPLCounts**

A Counts object property, which gets Counts class that holds a statistic counting for the model.

## Visual Basic Example:

```
Dim MyAmpl As AMPL
Dim result As String

Set MyAmpl = New AMPL
result = MyAmpl.AMPLRunCommand("options;" )
Debug.Print result
' or using AMPLPrompt
Debug.Print MyAmpl. AMPLPrompt
```

---

## Handling errors

All of the interfaces and associated methods and properties throw exceptions when they fail, to see in more detail the cause of the exception and find out about the errors. One can catch the exceptions and look at the returned error or at the AMPLoutput string property

### Visual Basic Example:

```
Sub Command1 ()
    On Error GoTo error_handler

    Dim result as String

    result = MyAmpl.AMPLRunCommand("param syntax_error ??? ;" )

    Debug.Print result

Exit Sub
error_handler:
    If Err.Number <> 0 Then
        Debug.print Err.Source + " raised Error " + vbCrLf + _
            "Description """" + Err.Description + vbCrLf
        Err.Clear
    End If
End Sub
```

In this example the visual basic caught the error object generated by AMPL COM when AMPL reports a syntax error, reading the entity param syntax\_error

### C++ Error Client

```
#include "AmplCom.tlb"
Using namespace AMPLCOMLib

int main( )
{    HRESULT hr;
    IAmpl *iAmpl= NULL;

    CoInitialize(NULL);
    try
    {
```

```

        hr = CoCreateInstance( CLSID_Ampl, NULL, CLSCTX_ALL, IID_IAmpl,
                               reinterpret_cast<void**>( &iAmpl));
        BSTR cmd = L"options;\n";
        BSTR ret;
        hr = iAmpl->get_AMPLRunCommand(cmd, &ret );

    }
    catch( const _com_error & E)
    {
        printf (" %s hex 0x%x \n", E.ErrorMessage() ,E.Error() );
    }
    if ( hr == S_OK )printf ( "ret  %ls \n", ret );
}

```

---

## Events handler

Applications can catch 4 events from AMPLCOM and do extra processing when an event occurs by implementing associated handlers. The events are as follows:

- 1) When AMPL is about to call solve command.
- 2) When AMPL returns from solve command
- 3) When AMPL is about to run commands
- 4) And when AMPL has finished running commands

### Visual Basic Example:

```
Dim WithEvents MyAmpl2 As AMPL
```

```
' code for the handlers
```

```
Public Sub MyAmpl2_StartSolve()
```

```
    Debug.Print "Got Start_Solve event from AMPL Obj"
```

```
    '... extra , your code for the event
```

```
End Sub
```

```
Public Sub MyAmpl2_EndSolve()
```

```
    Debug.Print "Got End_Solve event from AMPL Obj "
```

```
    '... extra , your code for the event
```

```
End Sub
```

```
Public Sub MyAmpl2_EndCommands()
```

```
    Debug.Print "Got End Commands event from AMPL Obj"
```

```
    '... extra, your code for the event
```

```
End Sub
```

```
Public Sub MyAmpl2_StartCommands()
```

```
    Debug.Print "Got Start Commands event from AMPL obj
```

```
    '... extra, your code for the event "
```

```
End Sub
```

---

## The Model Object

The Model object is an element of the AMPLModelCollection. It is a class that has its own AMPL instance and keeps its state and properties independent from other Models. In consequence, it will have model, data, solver and options and keeps the connection open with its AMPL instance during its existence.

### Collections contained by the Model Object

#### **Variables**

Returns the Variables collection AMPLVariablesCollection of all the variables contained in model (Read only)

#### **Constraints**

Returns the Constraints collection AMPLConstraintsCollection of all the constraints contained in model (Read only)

#### **Objectives**

Returns the Objectives collection AMPLObjectivesCollection of all the objectives contained in model (Read only)

#### **VariableVectors(filter)**

Returns a Variables collection AMPLVariablesCollection of selective variables with respect of (filter) in model (Read only)

#### **ConstraintVectors(filter)**

Returns a Constraints collection AMPLVariablesCollection of selective constraints with respect of (filter) in model (Read only)

#### **ObjectiveVectors(filter)**

Returns an Objectives collection AMPLVariablesCollection of selective objectives with respect of (filter) in model (Read only)

### Methods for Model Object

#### **ReadData ( FileName )**

Read data file FileName. Returns 0 if successful and throws an exception if there is an error, in which case the APMLOutput will contain the message

#### **ReadModel(FileName)**

Read model file FileName. Returns 0 if successful and throws an exception if there is an error, in which case the APMLOutput will contain the message

#### **AMPLOutput ()**

Returns a string containing the last AMPL output

#### **Solve()**

Solves the current model and returns Solution object

#### **Solution()**

Return a solution object for the current model

**Title**

A string property containing the title of the model. This plays the role of the key in the collection

**WorkingDirectory**

A string property to store a working directory for the AMPL instance associated to the current Model (read-write)

**Timing**

A Timing Object property associated to the current Model. See AMPLTiming object (read-only)

**AMPLCounts**

AMPLCounts object property Gets AMPLCounts object. See AMPLCounts object (read-only)

**Option**

An Option object to set/get AMPL options

**Solver**

A Solver object property sets/gets Current solver object

---

## The ModelCollection Object

### Item(index)

Parameter Index : can either be the model number in the collection or name of the model in the collection.  
Returns : a specified model object from the collection (Read only).

### Count()

Returns a number of item Model objects that are in the collection (Read only).

### Add(title)

Adds a new Method object to the collection. The Add method returns the newly added Model object if successful.

### Clear()

Method used to clear the collection and returns a long value for number of model in the collection before clear operation.

### Remove(ModelName)

This method removes the Model object having ModelName as name from a collection

### Example visual Basic

```
Set Cmodel = MyAmpl.AMPLModelCollection
Set model = Cmodel.Add ("STEEL")
Set model = Cmodel.Add ("DIET")
```

```
..
Debug.Print "Count = " & Cmodel.Count()
```

```
model = Cmodel.Item("STEEL")
```

```
'-----
model.ReadModel ("..\models\steel.mod")
model.ReadData ("..\models\steel.dat")
model.Solve
Debug.Print model.AmplOutput
```

```
'-----
Set onemodel = Cmodel.Item ("DIET")
onemodel.ReadModel ("..\models\diet.mod")
onemodel.ReadData ("..\models\diet2a.dat")
onemodel.Solve
Debug.Print onemodel.AmplOutput
```

```
'-----
Debug.Print Cmodel.Count()
```

```
Cmodel.Clear
```

---

## The Commands Object

The Commands object is the object which gives access to all AMPL commands, as if we are in the command line. The name of the methods for this object are the names of AMPL commands. When calling one of these methods, we call it with the same name as we do with AMPL, except that we remove the command name from the parameter. For example the AMPL command `AMPL: cd "path";` using this Commands Object. say `lcmd`, the command will become: `lcmd.cd("path")`

### Methods for Commands Object

#### **cd**

Method to change current directory

#### **check(stringcheck)**

Method to check perform all check commands

#### **close(filename)**

Method to close filename

#### **data( filename)**

This method reads data filename and returns the AMPL message

#### **display(paramstring)**

This method displays `paramstring` as if we are using AMPL display command, `paramstring` can be either formulae or model entity name. This method returns the AMPL display .

#### **drop(ConObjName)**

This method drops a constraint or objective named `ConObjName` , from being considered as constraint or objective (resp)

#### **end(str)**

This method ends the input from current input file

#### **environ(Env)**

method, `environ` sets environment for a problem instance

#### **fix(VarName)**

method, fixes 'freezes' a variable at its current value

#### **let(Format)**

method , changes data values (e-g : `let ("v := 1")`)

#### **load(FunctionName)**

Method, loads dynamic function library

#### **model(FileName)**

Switchs from the AMPL current mode to AMPL model mode; optionally include file contents, returns AMPL output.

#### **objective(ObjectiveName)**

Selects an objective to be optimized for the next solve command.

**option(Optiondef)**

This method sets or displays option values

**options()**

This method returns a string containing all AMPL options;

**print(FormatnItems)**

method print, having **FormatnItems** as parameter this method returns in a string model entities and expressions unformatted .

**printf(Format)**

Method printf, returns model entities and expressions as formatted in Format parameter;

**problem(ProblemName)**

This method defines a problem or switch to a named problem

**Purge(EntityName)**

This method purges (removes) model entity from the model

**read(FileName)**

Method reads (takes) input from that file.

**redeclare(Entitydef)**

This Method redeclares (changes) declaration of an entity as define in Entitydef string

**reload(libraryname)**

Method , reloads dynamic function library

**remove(filename)**

Method, removes remove file

**reset(string)**

Method, resets specified entities to their initial state

**Restore(string)**

Method Restore, restores, (undo a drop commands) previously executed

**Show(string )**

Method show, returns a string containing explicit definition of entities defined in a string parameter

**Update(string)**

Method, updates and allows updating specified data

**unfix**

This method, unfixes undo a 'fix command' previously executed

**write**

This method writes out problem instance.

**xref(EntName)**

Method. Xref, returns a string containing a dependencies among entities define in EntName.

**unload(LibName)**

This method unloads dynamic function library

**solve(name)**

Method solve, sends current instance to a solver and retrieve solution which is returned in a string

**delete\_entity(EntName)**

Method, deletes a model entities as defined in EntName and returns AMPL output.

**commands(FileName)**

Method commands, reads and interprets commands from file, the result then is returns as string.

**call(ImpFunName)**

This method calls imported function and returns a result as string

**Example visual Basic**

```
Dim MyAmpl as AMPLCOMLib .AMPL
Dim cmd As AMPLCOMLib.AMPLCommands

Set MyAmpl = New AMPL
Set cmd = MyAmpl.AMPLCommands

cmd.cd (" ")
cmd.Model ("steel.mod")
cmd.Data ("steel.dat")
cmd.Solve
Debug.Print MyAmpl.Output
```

## The AMPLConstraint Object

### **Value**

A double value property containing the Body, current value of constraint

### **LowerBound**

A double value property containing the Lower bound

### **LowerBoundSolver**

A double value property containing the Lower bound for solver (adjusted for fixed variable

### **CurrentInitialDualVariable**

A double value property containing the Current initial guess for dual variable

### **InitialDualVariable**

A double value property containing the Initial guess for dual variable

### **CurrentDualVariable**

A double value property containing the Current dual variable

### **LowerDual**

A double value property containing the Lower dual value (for body  $\geq$  lower bound

### **LowerSlack**

A double value property containing the Lower slack (body - lower bound)

### **Slack**

A double value property containing the Slack (min (lowerslack - upper Slack)

### **SolverStatus**

A string property containing solver status for this constraint (Read/Write)

### **Status**

A string property containing the constraint status

### **UpperBound**

A double value property containing the upper bound value for the constraint

### **UpperBoundSolver**

A double value property containing the Upper bound for solver (adjusted for fixed variable)

### **UpperDualValue**

A double value property containing the Upper dual value (for body  $\leq$  upper bound

### **UpperSlackValue**

A double value property containing the Upper slack (upper - body)

### **Name**

A string property containing the name of the constraint

### **Declaration**

A string property containing the expanded constraint declaration

### **SolDeclaration**

A string property containing the form in which AMPL has sent a constraint to the solver

**RHSValue**

A double value property containing the RHSValue, ( Read/Write)

---

## The Variable Object

### Properties for Variable Object

**Name**

A string property containing the Name of the variable

**CurrentValue**

A double value property containing the Current Value of variable

**CurrentInitialGuess**

A double value property containing the Current initial guess

**InitialGuess**

A double value property containing the Initial guess set by data , default or by assignement

**CurrentLowerBound**

A double value property containing the Current lower bound

**InitialLowerBound**

A double value property containing the Initial lower bound

**CurrentUpperBound**

A double value property containing the Current upper bound

**InitialUpperBound**

A double value property containing the Initial upper bound"]]

**ReducedCost**

A double value property containing the Reduced Cost

**LowerReducedCost**

A double value property containing the Lower Reduced Cost (for variable  $\geq$  lower bound)

**UpperReducedCost**

A double value property containing the Upper Reduced Cost ( for variable  $\leq$  upper bound)

**Slack**

A double value property containing the Slack value

**UpperSlack**

A double value property containing the Upper Slack (upper bound – value

**LowerSlack**

A double value property containing the Lower Slack ( value - lower bound)

**Declaration**

This string property contains Declaration for this variable in the model

**AmplStatus**

This string property contains AMPL status.

**SolverStatus**

This string property contains SolverStatus (Read/Write)

**Methods for Variable Object**

**IsUsed**

This method tests if the variable is used, returns boolean value

**IsFixed**

This method tests if the variable is fixed by let , returns boolean value

**IsEliminated**

This method tests if the variable is eliminated by pre-solve, returns boolean value

**IsSubstituted**

This method tests whether the variable is substituted, or defined variable, returns boolean value

---

## The AMPLVariables Collection Object

The Variables Collection Object is used to store a set of variable objects. This Collection is used by the parent object: Model to store a list of all the variables defined in the formulated model

### Properties for the AMPLVariablesCollection collection

#### **Item(index)**

Parameter Index : can either be the variable number in the collection or name of the variable in the model.  
Returns : a specified Variable object from the collection (Read only).

#### **Count()**

Returns a number of item Variable objects that are in the collection (Read only).

#### **VariablesCount ()**

Returns a number of item Variable objects that are currently in the model.

#### **BinariesCount()**

Returns a number of binary(0,1) variables objects that are currently in the model.

#### **IntegersCount()**

Returns a number of general integer variables(excluding binaries objects that are currently in the model).

#### **NLVariablesCount()**

Returns a number of nonlinear variables that are currently in the model.

---

## The Objective Object

### **Name**

A string name property contains a name of the an objective object that is currently in the model

### **CurrentValue**

"A double value property containing current value of objective

---

## The AMPLObjectives Collection Object

### **Item(index)**

Paramater Index : can either be the Objective number in the collection or name of the Objective in the model.

Returns : a specified Objective object from the collection (Read only).

### **Count()**

Returns a number of item Objective objects that are in the collection (Read only).

### **Add( ObjName)**

This method creates an Objective object with the name ObjName and adds it in the Objectives Collection, and return the newly created object

### **ObjectivesCount**

A long value property containg a number of objectives

### **NLObjectivesCount**

A long value property containing a number of nonlinear objectives

### **ObjectiveGradientNZerosCount**

A long value property containing a number of objective gradient nonzeros

---

## The AMPLConstraints Collection Object

### Properties for the AMPLConstraintsCollection collection

#### **Item(index)**

Parameter Index : can either be the constraint number in the collection or name of the constraint in the model.

Returns : a specified Constraint object from the collection (Read only).

#### **Count**

Returns a number of item Constraint objects that are in the collection (Read only).

#### **ConstraintsJNNonZeroCount**

A long value property contains a number of constraints Jacobian matrix nonzeros (Read only)

#### **ConstraintsCount**

A long value property contains a number of Constraints (Read only)

#### **CompConditionsCount**

A long value property contains a number of complementarity constraints before presolve (Read only)

#### **PresolveCompConditionsCount**

A long value property contains a number of complementarity constraints after presolve (Read only)

#### **LNetworkConstraintsCount**

A long value property contains a number of linear network constraints (Read only)

#### **NLCompConstraintsCount**

A long value property contains a number of nonlinear complementarity constraints (Read only)

#### **NLConstraintsCount**

A long value property contains a number of nonlinear constraints (Read only)

#### **NLNetworkConstraintsCount**

A long value property contains a number of nonlinear network constraints (Read only)

#### **LCompConstraintsCount**

A long value property contains a number of linear complementarity constraints (Read only)

## The Solution Object

---

The solution object is a property of the parents Model object and AMPL COM object (the main object).

This solution object contains the latest information about the solution associated with its parents. The Solution object properties are only available after the model has been solved.

### Properties for the Solution object

#### **ResultString**

A string property containing the result string of the last solver run (Read only)

#### **ResultNum**

A long value property containing the result number of the last solver run (Read only)

#### **Message**

A string property containing the message of the last solver run (Read only)

#### **ResultTable**

A string property containing the solver table for returned code (Read only)

#### **ExitCode**

A long value property containing the solver exit code for last solver run (Read only)

#### Visual Basic Example:

```
Dim MyAmpl As AMPL
Dim colmodel As ModelCollection
Dim model As Model
Dim sol As Solution
Set MyAmpl = New AMPL
Set colmodel = MyAmpl.AMPLModelCollection

colmodel.Add ("Diet")
Set twomodel = colmodel.Item("Diet")
model.ReadModel ("..\models\Diet.mod")
model.ReadData ("..\models\diet2a.dat")
model.Solve
Debug.Print model.AmplOutput

Set sol = model.Solution

Debug.Print "ExitCode" & sol.ExitCode
Debug.Print "Message" & sol.Message
Debug.Print "ResultTable" & sol.ResultTable
Debug.Print "ResultNum" & sol.ResultNum
Debug.Print "ResultString" & sol.ResultString
```

---

## The Timing Object

The Timing object is a property of the parents Model object and AMPL object (the main object). This object contains the latest information about the timing associated to AMPL instance in the parent.

### Properties for the Timing Object

#### **Elapsed\_time**

A double value property containing the elapsed time in seconds since the start of AMPL process instance

#### **System\_time**

A double value property containing the system CPU time in seconds used by AMPL process itself.

#### **User\_time**

A double value property containing the used CPU time in seconds used by AMPL process itself.

#### **Solve\_elapsed\_time**

A double value property containing the elapsed time in seconds for most recent solve command.

#### **Solve\_system\_time**

A double value property containing the system CPU time in seconds used by most recent solve command.

#### **Solve\_user\_time**

A double value property containing the user CPU time in seconds used by most recent solve command.

#### **Total\_solve\_elapsed\_time**

A double value property containing the elapsed time in seconds used by all solve commands called from AMPL process instance .

#### **Total\_solve\_system\_time**

A double value property containing the system CPU time in seconds used by all solve commands called from AMPL process instance .

#### **Total\_solve\_user\_time**

A double value property, the user CPU time in seconds used by all solve commands called from AMPL process instance .

### Visual Basic Example:

```
Dim MyAmpl As New AMPL
Dim timing As AMPLTiming
```

```
Dim Cmodel As ModelCollection  
Dim Steelmodel As Model
```

```
Set Cmodel = MyAmpl.AMPLModelCollection  
Set Steelmodel = Cmodel.Add ("STEEL")
```

```
Steelmodel.ReadModel ("steel.mod")  
Steelmodel.ReadData ("steel.dat")  
Steelmodel.Solve
```

```
Set timing = Steelmodel.timing
```

```
Debug.Print " Time for Steel model :"  
Debug.Print "Elapsed_time           " & timing.Elapsed_time  
Debug.Print "Solve_elapsed_time      " & timing.Solve_elapsed_time  
Debug.Print "Solve_system_time         " & timing.Solve_system_time  
Debug.Print "Solve_user_time              " & timing.Solve_user_time  
Debug.Print "System_time                  " & timing.System_time  
Debug.Print "Total_solve_elapsed_time      " & timing.Total_solve_elapsed_time  
Debug.Print "Total_solve_system_time       " & timing.Total_solve_system_time  
Debug.Print "Total_solve_user_time         " & timing.Total_solve_user_time  
Debug.Print "User_time                     " & timing.User_time
```

---

## The AMPLCounts Object

The AMPLCounts Object is a property of Parent Model object or the AMPLCOM object and it contains all the count statistic for the associated model

### **ConstraintsCount**

A long value property containing a number of Constraints before presolve

### **CompConditionsCount**

A long value property containing a number of complementarity conditions before presolve

### **PresolveCompConditionsCount**

A long value property containing a number of complementarity conditions after presolve

### **LNetworkConstraintsCount**

A long value property containing a number of linear network constraints

### **NLCompConstraintsCount**

A long value property containing a number of nonlinear complementarity constraints

### **NLConstraintsCount**

A long value property containing a number of nonlinear constraints

### **ConstraintsJMNonZeroCount**

A long value property containing a number of constraints Jacobian matrix nonzeros

### **VariableCount**

A long value property containing a number of variables"]]

### **BinariesCount**

A long value property containing a property a number of binary(0,1) variables

### **IntegerCount**

A long value property contains number of general integer variables(excluding binaries

### **NLVariablesCount**

A long value property containing number of nonlinear variables

### **ObjectivesCount**

A long value property containing number of objectives

### **PresolveConstraintsCount**

A long value property containing number of ordinary constraints after presolve

### **NLObjectivesCount**

A long value property containing number nonlinear objectives

### **ObjectiveGradientNZerosCount**

A long value property containing number objective gradient nonzeros

### **LCompConstraintsCount**

A long value property containing number of linear complementarity constraints

### **NLNetworkConstraintsCount**

a long value property Number of nonlinear network constraints

### Visual Basic Example:

```
Dim ampobjcounts As AMPLCounts
Set ampobjcounts = MyAmpl.AMPLCounts

Debug.Print "-----Counts-----"

Debug.Print "BinariesCount :" & ampobjcounts.BinariesCount
Debug.Print ampobjcounts.CompConditionsCount
Debug.Print ampobjcounts.ConstraintsCount
Debug.Print ampobjcounts.ConstraintsJMNonZeroCount
Debug.Print "IntegerCount :" & ampobjcounts.IntegerCount
Debug.Print ampobjcounts.LCompConstraintsCount
Debug.Print ampobjcounts.NLCompConstraintsCount
Debug.Print ampobjcounts.NLVariablesCount
Debug.Print "ObjectivesCount :" & ampobjcounts.ObjectivesCount
Debug.Print "VariableCount :" & ampobjcounts.VariableCount
'....
Debug.Print MyAmpl.AMPLPrompt
```

---

## The Option Object

This object contains a pair of strings to store AMPL and solver option

### **Parameter**

This property contains the name of the option (Read/Write)

### **Value**

This property contains the value of the option (Read/Write)

---

## The OptionCollection Object

### **Item (Index)**

Parameter Index : can either be the variable number in the collection or name of the variable in the model.  
Returns : a specified Variable object from the collection (Read only).

### **Count()**

Returns a number or item Option objects that are in the collection (Read only).

**Add( OptionName,OptionValue)**

The method Add creates and adds a new option, and returns the new Option object

**Clear()**

This method clears the collection and returns a long value for number of option in the collection before clear operation.

**WriteOptionsToFile( filename)**

This method writes all the option in the collection to the file

**ReadOptionsFromFile(filename)**

This method reads the options from file and store them in the collection

---

## The Solver Object

**SolverName**

A string value property containing a solver name (Read/Write)

**Comment**

A string value property containing a comment about solver (Read/Write)

**Options**

OptionCollection object property containing solver options collection

**DriverName**

A string value property containing a driver name (Read/Write)

**Option (param, value)**

This method adds an option to the solver collection 'options

---

## The Solver Collection Object

**Item(Index)**

Parameter Index : can either be the variable number in the collection or name of the variable in the model.  
Returns : a specified Variable object from the collection (Read only).

**Count**

Returns a number or item Solver objects that are in the collection (Read only).

### Add(solver)

Parameter solver: can either be the Solver object or name of the solver (e-g: "cplex")

Method Add a given Solver object Or create and add a new one with the given name, returns a reference to a newly added solver.

### Clear()

This method clears the collection and returns a long value for number of solver in the collection before clear operation.

### Visual Basic Example:

```
Dim solver As AMPLCOMLib.solver
Dim OptionSolver As AMPLCOMLib.Option
Dim ColSolver As AMPLCOMLib.SolverCollection

....

Set ColSolver = MyAmpl.AMPLSolverCollection
' create a new solver object
Set solver = New AMPLCOMLib.solver

solver.SolverName = "Afortmp_display_option"
solver.DriverName = "afortmp"
solver.Comment = "fortmp uses display level 4 "
solver.Option "displaysolver", "4"

' Add a newly created solver to the collection
Set solver = ColSolver.Add(solver)

' add minus solver
Set solver = ColSolver.Add("minus")
solver.Comment = "solver minus student version "

' add cplex solver
Set solver = ColSolver.Add("cplex")
solver.Comment = "solver cplex8 student version "

For Each solver In ColSolver
  For each model in Colmodel
    model.solver = solver
    model.solve
    Debug.Print "model " & model.Title & _
      "Solver "& solve.SolverName & _
      "Result " & model.Solution.ResultString
  Next ' model
Next ' solver
```

## The AMPLConstraint Object

### **Value**

A Double value property contains property Body, current value of constraint

### **CurrentInitialDualVariable**

A Double value property contains property Current initial guess for dual variable

### **InitialDualVariable**

A Double value property contains property Initial guess for dual variable

### **CurrentDualVariable**

A Double value property contains Current dual variable

### **LowerBound**

A Double value property contains Lower bound

### **LowerBoundSolver**

A Double value property contains Lower bound for solver (adjusted for fixed variable)

### **LowerDual**

A Double value property contains Lower dual value (for body  $\geq$  lower bound)

### **LowerSlack**

A Double value property contains Lower slack (body - lower bound)

### **Slack**

A Double value property contains Slack ( min(lowerslack - upper Slack

### **SolverStatus**

A string property contains a Solver Status (Read|Write)

### **Status**

A string property contains AMPL Status of the constraint object

### **UpperBound**

A Double value property contains the Upper bound for this constraint object

### **UpperBoundSolver**

A Double value property contains the Upper bound for solver (adjusted for fixed variable) for this constraint object

### **UpperDualValue**

A Double value property contains the Upper dual value (for body  $\leq$  upper bound) for this constraint object

### **UpperSlackValue**

A Double value property contains the Upper slack (upper - body) for the constraint object

### **Name**

A string property contains the name of the constraint object (Read Only)

**Declaration**

A string property contains the expanded constraint for the constraint object (Read Only)

**SolDeclaration**

A string property contains the form in which AMPL has sent a constraint to solver (Read Only)

**RHSValue**

A Double value property contains a right hand side value of the constraint object (Read|Write)

Visual Basic Example:

```
Dim vbCollConstraint As AMPLConstraintsCollection
DIM c As AMPLConstraint

Set vbCollConstraint = model.ConstraintVectors("DIET")

For i = 1 To vbCollConstraint.count
    Set c = vbCollConstraint.Item(i)
    Debug.Print "Constraint name " & c.Name
    Debug.Print "Slack " & c.Slack
    Debug.Print "RHSValue " & c.RHSValue
    c.RHSValue = 10
Next i
```