Active X

User Guide

ProModel

Visualize Analyze Optimize VAO

www.promodel.com

Disclaimer

The information in this addendum has been provided by ProModel Corporation to document the ActiveX capabilities of ProModel, MedModel, and ServiceModel. The information in this addendum is subject to change without notice and does not represent a commitment on the part of ProModel Corporation. The software described in this addendum is supplied under a license agreement and may be copied only under the terms of the license agreement. No part of this addendum may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopying, for any purpose other than the owner's personal use, without the express written permission of ProModel Corporation.

Copyright Information

Printed in the United States of America

6/18/2014

Table of Contents

Introduction

ProModel's ActiveX Components

ProModel's ActiveX Automation capability allows you to use any ActiveX-enabled language (e.g. Microsoft Visual Basic, VBA, Visual C#, or Visual C++) to:

- Build customized user interfaces for ProModel
- Add, Change or Delete model data from external data sources, such as spreadsheets, databases or ASCII text files
- Control ProModel from another application
- Extract output data and place it in a spreadsheet or database

The intent of this manual is to give you information that is specific to the ActiveX components automatically installed with your PROMODEL product. We have included plenty of examples to help you understand how to use these powerful tools. All of the examples that follow use Visual Basic commands and syntax. The variable names used in this manual are merely examples, you will want to replace them with names that are meaningful to you.

ProModel Objects

All of the Type Libraries have an Application object, a Data object and a Runtime object. Each object gives you access to different parts of ProModel. The following is a brief description of these objects:

- Application – gives you operational control of ProModel. For example, using this object you can load a model or start a simulation.
- Data – allows access to model information (like Locations or Processing records).
- Runtime – gives you methods that can update or capture the values of statistics while a simulation is running.

When you declare object variables, we recommend that you use the ProModel object types. These object types include the ProModel, ProModelPM, ProModelMM and ProModelSM type libraries. The ProModel type library behaves in a unique manner compared with the other three. It is generic in the sense that it can be used to control either ProModel, MedModel or ServiceModel. Whereas, the ProModelPM type library is specific to the ProModel product, ProModelMM is specific to the MedModel product and ProModelSM is specific to the ServiceModel product.

ProModel, MedModel and ServiceModel include a useful utility called ProSetter. If your system includes ProModel and MedModel (and/or ServiceModel) then this utility may be used in order to change which product is launched by the generic ProModel type library.
Here are examples of how to declare and create an instance of each of the ProModel objects.

ProModel Type Library

Dim *ObjVarName* as ProModel.CProModel
Set *ObjVarName* = CreateObject("ProModel")

```
    Dim ObjVarName as proModel.CProModelData
    Set ObjVarName = Create Object("ProModelData")

    Dim ObjVarName as ProModel.CRuntime
    Set ObjVarName = CreateObject("ProModel.CRuntime")
```

ProModelPM Type Library (ProModel product)

```
    Dim ObjVarName as PromodelPM.Application
    Set ObjVarName = CreateObject("PromodelPM.Application")

    Dim ObjVarName as PromodelPM.Data
    Set ObjVarName = CreateObject("PromodelPM.Data")

    Dim ObjVarName as PromodelPM.Runtime
    Set ObjVarName = CreateObject("PromodelPM.Runtime")
```

PromodelMM Type Library (MedModel product)

```
    Dim ObjVarName as PromodelMM.Application
    Set ObjVarName = CreateObject("PromodelMM.Application")

    Dim ObjVarName as PromodelMM.Data
    Set ObjVarName = CreateObject("PromodelMM.Data")

    Dim ObjVarName as PromodelMM.Runtime
    Set ObjVarName = CreateObject("PromodelMM.Runtime")
```

PromodelSM Type Library (ServiceModel product)

```
    Dim ObjVarName as PromodelSM.Application
    Set ObjVarName = CreateObject("PromodelSM.Application")

    Dim ObjVarName as PromodelSM.Data
    Set ObjVarName = CreateObject("PromodelSM.Data")

    Dim ObjVarName as PromodelSM.Runtime
    Set ObjVarName = CreateObject("PromodelSM.Runtime")
```

Other Objects

ProModel also provides a few other objects that will enable you to:

- Access the data found in the classic output statistics files (*.rdb, *.rdt).
- Capture and respond to events that take place in ProModel There are two files that allow you to work with the classic output statistics.

They are:

- Rdbsvr.dll – This DLL gives you access to the General Statistics data.
- Rdtdata.dll – This DLL gives you access to the Time Series data.

The "ProModel Events 8.0 Type Library" (pmcp80.dll) gives you a way to respond to things that happen (events) in ProModel. With it, you can:

- Trap and respond to errors without user intervention
- Hide informational messages
- Get information about changes the user makes during simulation, such as changing the animation speed or the view.
- Handle ProModel "Prompt" dialogs or replace them with your own dialogs

Methods

Each ProModel ActiveX ***Object*** has one or more ***methods*** (actions it can perform).
Each method may or may not have ***Parameters*** (details that you provide or that ProModel returns).

Syntax

Most methods can be used in two different ways. The first example shown below does not have a variable to receive the result code. The second example uses a slightly different syntax that will enable you to determine whether or not the method executed successfully.

Syntax 1

If you don't need to know whether the method succeeded or failed, use the method name, followed by a space, then each parameter. Use a comma to separate the parameters.

```
Sub FindRecord()
    Dim pmObject As ProModel.CProModel
    Dim pmDataObject As ProModel.CProModelData


    Set pmObject = CreateObject("ProModel")
    Set pmDataObject = CreateObject("ProModelData")
    pmDataObject.Populate


    pmDataObject.SelectMainRecordByName 1, "MyLocation"
```

```
    Set pmObject = Nothing
    Set pmDataObject = Nothing
End Sub
```

Syntax 2:

When you use this syntax, you can determine what happened when the method was executed. You will need to declare a variable to hold the result code returned by ProModel. In the method call, start with that variable, then an equals (=) sign, followed by the method name and a left parenthesis. As with Syntax 1, the parameters are listed in order, separated by commas. Finally, a right parenthesis closes the method call.

```
Sub FindRecord()
    Dim pmObject As ProModel.CProModel
    Dim pmDataObject As ProModel.CProModelData
    Dim vResultCode as Long

    Set pmObject = CreateObject("ProModel")
    Set pmDataObject = CreateObject("ProModelData")
    pmDataObject.Populate

    vResultCode = pmDataObject.SelectMainRecordByName (1, "MyLocation")

    If vResultCode <> 0 Then
      MsgBox "Record Not Found"
    End If

    Set pmObject = Nothing
    Set pmDataObject = Nothing
End Sub
```

Result Codes

Almost all of the methods in the ProModelData object may optionally be written to return a result code. Result codes help you know if the execution of the method was successful. For a list of result codes, see Locations Table (1): PMD_Table_ Location on page 108.

Definitions

Record Type = Table Number

Record = Row

Record Index = Row Number

Field = Column

Field Index = Column Number

Parameters

ProModel has two kinds of parameters: Input Parameters and Output Parameters. Each parameter has a specific data type. In this document, the data types shown are Visual Basic data types. If you are programming in another ActiveX enabled language, you will need to use the equivalent types in that language.

For input parameters, you may place a value of the correct type directly in the method call, or place the value into a variable of the appropriate type, then use the variable to satisfy the parameter requirements. For output parameters you must supply a variable of the appropriate type to receive the values returned by ProModel.
For output parameters, you must use a variable.

Although some programming languages will automatically convert one data type to another, you may need to use variables to work with some methods. In the example for the Zoom method, shown on Zoom on page 44, you will notice that a variable of type Double is declared, then the numeric value is placed in that variable, and the variable is used in the method call. If the Zoom method is executed with a number as the parameter, an error is raised, because Visual Basic automatically converts the number, but not to the correct type. So, by placing the number in a variable explicitly defined as a Double, the problem of incorrect conversion is avoided. Other languages may have similar limitations.

PM Constants

You may have noticed that the first parameter in the “SelectMainRecordByName” method used in the code sample in the Syntax section contains a number. This number references the Locations table. Because it can be difficult to remember the number codes associated with so many tables, fields, status codes, etc. we have developed text values that you may use in place of the numbers. To use the pre-defined constants, simply include the PMConstants.bas in one of your program modules (PMConstants.bas can be found in the PowerTools directory where your ProModel application was installed). Whether you use the pre-defined constants, your own constants or the numbers in your code makes no difference to ProModel. You can even use constants and numbers in the same method call. See the Appendix: Table Definitions for a complete list of pre-defined constants, their values and descriptions.

Working with Data

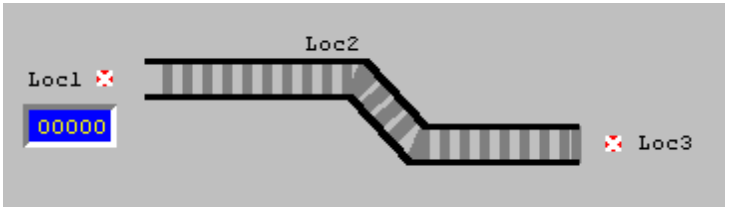
When you create a model, what you are actually doing is recording information about your business processes. When you run a simulation, ProModel takes that information and uses it to display an animated representation of your processes and to generate the statistical information about your processes.

You can think of each model you create as a database. Within each model database there are many tables, such as Locations, Path Networks, and Processing. ProModel's ActiveX uses numbers to reference each table. In the appendix, you will find a list of tables and their corresponding numbers.

Each table contains one or more fields. You may have records in many of these tables, with information in some of the fields. If you are accustomed to spreadsheets, each model would be a workbook, each table a sheet. Fields would correspond to columns and records would be the rows. Fields and records are also referenced by number. A complete list of tables and fields, along with their reference numbers, can be found in Appendix A on Appendix: Table Definitions on page 106.

With a database, you have an element that is not as easy to create in a spreadsheet, the parent-child relationship. Since ProModel is a database environment, there are many tables that have that type of relationship. Let's look at something a little more concrete.

Suppose you have just started a new model and have defined the locations shown below:



For each of these three locations, there is more that one graphic icon. In addition, the conveyor for Loc2 has a couple of joints (bends) in it. So, your Locations table would have three records. Each of these records would have more than one graphic. Information about the graphics is stored in a different table, which is a child of the Locations table. The child table (Location Graphics) also has a child of its own, the Q/Conveyor Joints table. A diagram of the table relationships would look something like this:

Locations	Location Graphics	Q/Conveyor Joints
1. Loc1	1. Part Spot 2. Text Box 3. Counter	
2. Loc2	1. Conveyor	1. Start Point 2. First Bend 3. Second Bend 4. End Point

3. Loc3	2. Text Box 1. Part Spot 2. Text Box	

If we want to change the name of Loc1 to “EntryPoint”, we would need to make that change in the first record of the Locations table (table number 1). So, we would select record one in table one.

If we want to move Loc3 to a different place in the layout, we would first need to select the Loc3 record in the parent table (Locations). Then we can select records in the child table (Location Graphics – table 45) and change the X and Y values of each.

Now, if we want to move the end of the Loc2 conveyor closer to our new Loc3 position, we first need to select Loc2 in the parent table (Locations). Then, we would select Conveyor in the child table (Location Graphics) for Loc2. Finally, we can select the End Point record in the second level child table (Q/Conveyor Joints – table 61) and change its X and Y values.

You may have realized from the description above that you must always select a record before you can work with any of the information in that record. This holds true for all of the data tables in ProModel. Records can be selected either by name (if it has one) or by index number. The index number is the number in the upper right corner of the data window in ProModel and represents the record’s position in the table (its row number). When a new record is added to a table in ProModel, that record is automatically selected. If a record is deleted from a table, the index numbers of following records will be changed to close the gap.

The fields (columns) are also numbered, from left to right. One caveat: the first field is not always the one you might think it is. For example, the Name field in the Locations table is actually number two. Field one is no longer used.

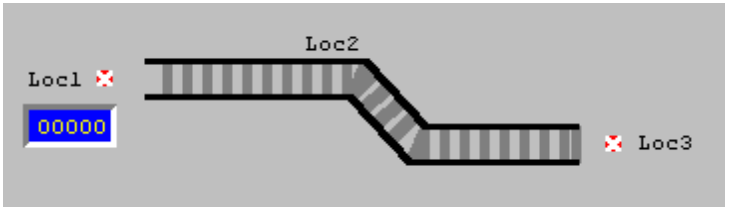
Working with Graphics

Though the basic principles for working with data in the graphics tables are the same as working with any other data table in ProModel, there are some things that work a bit differently. This section covers those differences.

We have already indicated that the Location Graphics table is a child of the Locations table. Entities and Resources also have their own graphics tables. The graphics information for the Path Networks table are actually included in the parent table (it has other children).

With most data tables in ProModel, there are two methods available for adding new records – the AppendRecord and InsertRecord methods. When working with any of the Graphics child tables, only the AppendRecord() is used, InsertRecord() will return an error.

Because there are many types of graphics associated with locations, working with the Location Graphics table can be a bit tricky at first. Let’s look at the example we used previously:



Suppose that we want to add a Counter graphic to Loc3.

First, we must select the record for Loc3 in the Locations table. Then, we would use the AppendRecord method with the Location Graphics table. A library graphic is automatically created. However, we don't want a Library graphic, so we must use the SetIntFieldValue method to change the value in the Type field to 4 (Counter).

With other data tables, the SetIntFieldValue method would simply replace the value in the specified field. Not so in the Location Graphics table. When using this method on the Type field, it actually deletes the library graphic record, then appends a new record (counter graphic) and sets the value of the Type field to 4. This is fine if you have just appended a new record, but what if you want to change the Type of an existing graphic?

Unless the graphic you wish to change is the last record in the table, the index number will change (because the record is deleted and then a new record added to the end of the table). This also means that the index number of other graphics may change, as outlined earlier.

Changing the graphic type also modifies the apparent structure of the Location Graphics table itself. Each graphic type has different fields associated with it. (see Locations Table (1): PMD_Table_Location on page 108). A field by the same name may have different numeric values for different graphic types. The one constant is the Type field, it is always field 1.

What if we create a location, then we append a graphic and we want to change it to a Queue? When we change the value in the Type field to 2, the same process as above takes place, but a few more things happen as well. The Graphic Style field will have the default value of 2 (Line style), and two records will be added to the next level child table (Q Joints table, #61). The first record in the Q Joints table will have the X Position and Y Position both set to zero (0), indicating that the queue starts at the top left corner of the layout. The second record will have default X and Y values of 100, indicating that the queue ends 100 pixels to the right and 100 pixels down from there.

After that, you may work with the other fields of the Location Graphics table to further define how the Queue or Conveyor will look and function. You may also change the X and Y position values of the two records in the Queue/Conveyor Joints table, or add more joints by appending records. However, you cannot delete the two default records, since you cannot have a queue or conveyor with only one end. To delete the queue or conveyor, you must delete the record in the Location Graphics table.

You may notice as you work with Library Graphics that there is no easy way to determine what the Graphic ID number is for any given graphic icon. However, the Graphic ID is a required field when the graphic type is 1. The Graphic Editor is an application that is included with ProModel for your convenience. Unfortunately, as you add, change and delete icons in your graphics libraries, you may end up with numbers out of sequential order or missing altogether.

The following chapters will give you detailed information and examples for each of the objects and methods available with your installation of ProModel.

The ProModel Application Object

The ProModel Application object is found in ProModel.exe, MedModel.exe and ServiceModel.exe as "*ProModel", or "ProModel Type Library*". The *"ProModelPM",* or *"ProModel PM Type Library"* is found exclusively in ProModel.exe. Likewise MedModel.exe contains *"ProModelMM"* or *"ProModel MM Type Library"* and ServiceModel.exe contains *"ProModelSM",* or *"ProModel SM Type Library"*. If one or more of these libraries are not available in your Visual Basic References list, run ProModel.exe, MedModel.exe or ServiceModel.exe (found in the corresponding product folder) with the " /regserver" command line option.

The ProModel Application object may be used with the Professional, Gold Runtime and Student versions of ProModel. The Silver Runtime version does not support ProModel's ActiveX interface at all. The ProModel Application object has methods that control all the commonly used functions of ProModel, as well as the method used to initialize the Events object. The available methods are listed below.

EndReplication
GetEventsObject
GetSimTime
GetVersion
LoadModel
MergeModel
New
Quit
RedrawTables
RunScenarios
SaveAs
SetMenus
SetPan
SetViewRect
ShowTranslationDlg
Zoom
EndSimulation
GetOutputPath
GetStatus
LoadDefaults
MenuCommand
MsgBox
OpenModule
RedrawLayout
ReleaseEventsObject
Save
SetMacro
SetMessageModes
SetView
SetWindowPos
Simulate

EndReplication

Syntax: EndReplication

Description: Terminates a Replication before completion. Simulation will then proceed to the next replication. If EndReplication is called during the final replication, the simulation will end.

Parameters: None

Returns: Nothing

Example: This example loads a model, runs simulation for 30 minutes (1800 seconds), then terminates the current replication.

```
Sub QuitSimEarly()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime AS Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel (mfg_cost.mod)
  pmObject.Simulate

  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.EndReplication

  Set pmObject = Nothing
End Sub
```

EndSimulation

Syntax: EndSimulation
Description: Terminates a simulation run before completion. The EndSimulation method will have no effect if called when there is no simulation running. If running multiple replications, the entire simulation will end (remaining replications will not be run). The output viewer will be launched automatically.
Parameters: None
Returns: Nothing
Example: This example loads a model, runs it for 30 minutes (1800 seconds), then terminates the simulation.

```
Sub QuitSimEarly()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime AS Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel (mfg_cost.mod)
  pmObject.Simulate

  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.EndSimulation

  Set pmObject = Nothing
End Sub
```

GetEventsObject

Syntax: GetEventsObject
Description: Use this method to initialize the ProModel Events object. For further instructions see the Events Handler section.
Parameters: None
Returns: Event Handler object.
Example: This example is incomplete, but the code given shows how to use the GetEventsObject method. This must be executed before using the Events Handler.

```
Sub GenericSub()
  Dim pmObject As ProModel.CProModel
  Dim pmEventObject As PMCPLib.PMEvents

  Set pmObject = CreateObject("ProModel")
    Set pmEventObject = pmObject.GetEventsObject
    **More Code…
End Sub
```

GetSimTime

Syntax: GetSimTime (Seconds)
Description: Gets the current simulation time in seconds.
Parameters:
Seconds (Double) Output parameter that receives return value.
Returns: (Double) Current simulation clock time in seconds.
Example: This example loads a model, runs it for 30 minutes (1800 seconds), then terminates the simulation.

```
Sub QuitSimEarly()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime AS Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel (mfg_cost.mod)
  pmObject.Simulate

  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.EndSimulation
  Set pmObject = Nothing
End Sub
```

GetStatus

Syntax: GetStatus
Description: Gets the current state of a loaded model. Use this method to trap events triggered by ProModel or the user. Note: This method is easier to use than the Events Handler, but less robust and reliable. For instance, the GetStatus method will work fine for simple applications, but for more complicated situations, such as running a series of models, it would work better to use the event handler approach to avoid problems with timing issues.
Parameters: None
Returns: Long. Status code (0 – 9) of the loaded model.

Status Codes:

1. Status unknown
2. No model loaded
3. Loading a model
4. Model loaded
5. Load error
6. Translating model
7. Simulation in progress
8. Simulation or translation terminated pre-maturely
9. Simulation completed
10. Simulation frozen

Example: This example starts the simulation for the loaded model and continuously checks the status. Then, when the status changes to 8 (normal completion), shows a message to let the user know it's done.

```
Sub WhatsUp()
  Dim pmObject As ProModel.CProModel
  Dim vStatus As Long

  Set pmObject = CreateObject("ProModel")
  pmObject.Simulate
   vStatus = pmObject.GetStatus


  Do
    DoEvents
    vStatus = pmObject.GetStatus
  Loop Until vStatus = 8

  MsgBox "I'm done."
```

```
    Set pmObject = Nothing
End Sub
```

GetOutputPath

Syntax: GetOutputPath
Description: Returns the fully qualified path where simulation ouptut will be written to. This path may also be found by using the ProModel user interface under Simulation Options | Output Path.
Parameters: None.
Returns: A string containing the fully qualified path where simulation output for a given model will be stored.
Example: This example loads a model, and then opens a VBA message box that displays the model's output path.

```
Sub DisplayOutputPath()
  Dim pmObject As ProModel.CProModel
  Dim outputPath As String

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  outputPath = pmObject.GetOutputPath
  MsgBox "Simulation output will be written to the following location: " & outputPath
  Set pmObject = Nothing
End Sub
```

GetVersion

Syntax: GetVersion
Description: Returns the version of the product you are using. This could be used to make certain that the user's version of ProModel is recent enough to contain certain features.
Parameters: None.
Returns: (String) Version number and build number shown in the 'Help > About' dialog box in ProModel.
Example: This example opens ProModel then returns the version information to the variable.

```
Sub VerNum()
  Dim pmObject As ProModel.CProModel
  Dim pmVersion As String

  Set pmObject = CreateObject("ProModel")
   pmVersion = pmObject.GetVersion
  MsgBox "You are using version " & pmVersion
  Set pmObject = Nothing
End Sub
```

LoadDefaults

Syntax: LoadDefaults ININame

Description: Reloads the INI file, which contains the ProModel defaults.

Parameters:

ININame The name of the INI file with the desired defaults.

LoadModel

Syntax: LoadModel FileName
Description: Loads the specified model.
Parameters:
FileName (String) Any valid path and model filename. Model files must have a .mod extension.
Returns: Nothing
Example: This example loads a model, runs it for 30 minutes (1800 seconds), then terminates the simulation.

```
Sub QuitSimEarly()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime AS Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel (mfg_cost.mod)
  pmObject.Simulate


  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.EndSimulation

  Set pmObject = Nothing
End Sub
```

MenuCommand

Syntax: MenuCommand CmdNum, lParam
Description: Executes a specified menu option (as if the user had selected it).
Parameters:

CmdNum (Long) The ID number of the menu command you wish to use. See Locations Table (1): PMD_Table_Location on page 108.

lParam (Long) For almost all MenuIDs, Parameter has no meaning, so a zero should be passed. For #4825 (Interactive Subroutine), Parameter will activate a specific subroutine. The number in Parameter refers to the order of the subroutine in the "Interact" list available during simulation, not it's index number in the Subroutine table.

Returns: Nothing

Example: The following example loads a model, then opens the General Information dialog.

```
Sub ChangeView()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel (mfg_cost.mod)
  pmObject.MenuCommand 2863, 0

  Set pmObject = Nothing
End Sub
```

MergeModel

Syntax: MergeModel FileName, Xpos, Ypos, Tag, Flags
Description: Merges the specified model into the currently loaded model. Model merging is useful when you wish to join sub models or fragmented model components to the original model.
Parameters:

FileName (String) The name of the file you wish to merge.
Xpos (Double) Horizontal placement of submodel (number of pixels over from left margin).
Ypos (Double) Vertical placement of submodel (number of pixels down from the top margin).
Tag (String) A label attached to the beginning or end of every identifier in the merged model.
Flags (Long) You can use flags to specify merge options such as whether to attach a tag to the beginning or end of identifiers, or whether or not to merge graphics libraries. Flags should be combined using the "Or" operator.

Returns: (Long) Result of the merge operation (1 = successful, 0 = fail).
Flags: When using more than one flag, use the keyword "Or" to connect, or simply add the values of the flags together, and enter that value for all flags.

2 Don't prompt
8 Prefix tag (default is suffix)
16 Merge new model's graphic library into the current model and save with original name
32 Merge new model's graphic library into the current model and save as the original name plus the model tag
64 Just use the original model's graphic library

Example: This example loads a model, then merges in mfg_cost.mod at an offset of 20x20 pixels from the upper-left corner of the layout. All identifiers associated with mfg_cost.mod will have "sub_" added to the beginning.

```
Sub AddClock()
  Dim pmObject As ProModel.CProModel
  Dim MySubMod As String

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  MySubMod = "mfg_cost.mod"
  pmObject.MergeModel MySubMod, 175, 125, "sub_", 8 Or 64

  Set pmObject = Nothing
End Sub
```

MsgBox

Syntax: MsgBox *Message*
Description: Displays a message box containing the indicated text. The message box will display in ProModel. If you wait for a response from within Excel or a custom VB interface, you may not see the message box. This may delay operations while the dialog awaits a response.
Parameters:
Message (String) The message you wish to display.
Returns: Nothing
Example: This example loads a model, then sends a message to the ProModel window that the user must clear before the next line of code will be executed.

```
Sub Hal()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.MsgBox "Good Morning, Dave."

  Set pmObject = Nothing
End Sub
```

New

Syntax: New
Description: Invokes the File/New command without launching the General Information Dialog.
Parameters: None
Returns: Nothing
Example: This example opens ProModel and loads a new (empty) model.

```
Sub NewMod()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.New

  Set pmObject = Nothing
End Sub
```

OpenModule

Syntax: OpenModule Module
Description: Opens a specific build module. A module is a table in the ProModel Build menu. Passing a zero (0) value will close any open modules.
Parameters:
Module (Long) A module ID number from the list below.
Returns: Nothing
Module ID Numbers:
0 No modules open
1 Locations
2 Arrivals
5 Entities
6 Resources
10 Variables
11 Attributes
12 Arrays
15 Function tables
16 Distribution tables
17 Cycle tables
18 Subroutines
19 Processing
24 Path Networks
25 Macros
26 Streams
27 External Files
34 Shift Assignments
36 Background Graphics (Behind grid)
37 Background Graphics (Front of grid)
Example: This example loads a model, then opens the Entities module, as if the user had selected Build >> Entities.

```
Sub OpenEntMod()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.OpenModule 5

  Set pmObject = Nothing
End Sub
```

Quit

Syntax: Quit
Description: Closes ProModel. The “Save Changes?” prompt will appear, if applicable.
Parameters: None
Returns: Nothing
Example: This example opens ProModel, then closes it.

```
Sub ClosePM()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
   pmObject.Quit

  Set pmObject = Nothing
End Sub
```

RedrawLayout

Syntax: RedrawLayout
Description: Refreshes the layout window. Useful when you add graphical objects via the ProModelData interface.
Parameters: None
Returns: Nothing
Example: The following examples loads a model, adds a location record, attaches a library graphic to the location , and redraws the layout.

```
Sub AddLoc()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim Xpos As Double
  Dim Ypos As Double

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  Xpos = 20
  Ypos = 20

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate
  pmDataObject.AppendRecord pmdTblLocation
  pmDataObject.AppendGraphicIcon pmdTblLocation, 1, 1, Xpos, Ypos
  pmObject.RedrawLayout

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

RedrawTables

Syntax: RedrawTables
Description: Refreshes any open tables. When using the ProModel Data Object to update model data, if the table being update is open, the screen will not show the changes until RedrawTables is executed or the module is closed and re-opened.
Parameters: None
Returns: Nothing
Example: This example loads a model, adds a new record to the Entities table, then refreshes the table to show the new record.

```
Sub AddEnt()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmObject.OpenModule 5
  pmDataObject.Populate
  pmDataObject.AppendRecord pmdTblEntity
  pmDataObject.SetStringFieldValue pmdTblEntity, pmdFldEntName, "MyNewEntity"
  pmObject.RedrawTables

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

ReleaseEventsObject

Syntax: ReleaseEventsObject
Description: This method is only useful when used with the GetEventsObject method. Its purpose is to remove the connection between the ProModel events object and any event handler(s) contained in the user's code. After this connection is removed, the user's event handler(s) will no longer receive event notifications emitted by the ProModel events object.
Parameters: None
Returns: Nothing
Example: This example releases an events object from the ProModel application object. The code also contains an events handler that listens for events emitted by the events object. Whenever the events object emits an event, it is received by the events handler. This causes a message to be printed to the VBA Immediate window (Ctrl+G to open). After the simulation has been running for 30 minutes, pmObject.ReleaseEventsObject() is called. This causes all events to stop printing to the Immediate window.

```
Private WithEvents pmEventsObject As PMCPLib.PmEvents

Sub HandleEventsUntilRelease()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime As Double

  CurSimTime=0
  Set pmObject = CreateObject("ProModel")
  Set pmEventsObject = pmObject.GetEventsObject()
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.Simulate

  Do While CurSimTime , 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.ReleaseEventsObject
  pmObject.EndSimulation

  Set pmObject = Nothing
  Set pmEventsObject = Nothing
End Sub
```

The ProModel events handler

```
Private Function pmEventsObject_PmEventsHandler(ByVal PmEventID As Long, ByVal Description As String) As
Long
    Debug.Print "Event ID:" & PmEventID & Chr(13) & "Event Description:" & Description
End Function
```

RunScenarios

Syntax: RunScenarios
Description: Runs scenarios defined in the model.
Parameters: None
Returns: Nothing
Example: This example loads a model, then runs the defined simulation scenarios.

```
Sub MfgScenarios()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel ("mfg_cost.mod")
  pmObject.RunScenarios

  Set pmObject = Nothing
End Sub
```

Save

Syntax: Save
Description: Saves any changes to the loaded model (to the same file). This method is only available with a professional license.
Parameters: None
Returns: Nothing
Examle: This example loads a model, adds a new record to the Entities table, then saves the changes.

```
Sub AddEnt()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate
  pmDataObject.AppendRecord 2
  pmDataObject.SetStringFieldValue 2, 2, "MyNewEntity"
  pmObject.Save

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SaveAs

Syntax: SaveAs ModelName
Description: Saves the loaded model to a specified file. This method is only available with a professional license.
Parameters:
ModelName (String) Any valid model path & filename.
Returns: Nothing
Example: This example loads a model, adds a record to the Entities table, then saves the changes to a new file.

```
Sub SaveModAs()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate
  pmDataObject.AppendRecord 2
  pmDataObject.SetStringFieldValue 2, 2, "MyNewEntity"
  pmObject.SaveAs "mfg_cost_new.mod"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SetMacro

Syntax: SetMacro Macro Name, Macro Value
Description: Sets the value of a ProModel macro.
Parameters:
MacroName (String) ID of any defined macro in loaded model.
MacroValue (String) New value for specified macro.
Returns: Nothing
Example: This example loads a model, changes a macro value, then runs the simulation with the new value.

```
Sub Use3Ops()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetMacro "Number_of_Operators", 3
  pmObject.Simulate

  Set pmObject = Nothing
End Sub
```

SetMenus

Syntax: SetMenus MenuSet, MenuOptions
Description: Modifies the menu display within ProModel. Currently, only the options used in the example are implemented.
Parameters:
MenuSet(Long) 1 = Edit Menu Set, 2 = Simulation Menu
MenuOptions(Long) 1 = Minimal
Returns: Nothing
Example: The following example loads the mfg_cost.mod, sets the Simulation menu to minimal, then runs the simulation. This will cause the menu during the simulation to show only "Simulation >> Pause/Resume" and "Help" options.

```
Sub RunMinMenu()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetMenus 2, 1
  pmObject.Simulate

  Set pmObject = Nothing
End Sub
```

SetMessageMode

Syntax: SetMessageMode Flags
Description: This method only applies to standard message boxes, it will have no effect on error messages, translation dialog or data dialogs accessed from the menu. With this method you can control what types of messages to display or not to display. Types are based on which control buttons are shown in the message box. The type(s) included in the *Flags* parameter will be the only types shown, all others will execute the default option, without showing the message box. If you wish to execute a non-default option without showing the message box, you can do so through the ProModel Event Handler.
Parameters:
Flags (Long) See list of flags below.
Returns: Nothing
Flags: Flags can be combinged using "+" or "OR", except for values 0 and 1.
0 Do not display any messages
1 Show Information messages
2 Show messages with OK button
4 Show messages with OK & Cancel buttons
8 Show messages with Retry & Cancel buttons
16 Show messages with Yes & No buttons
32 Show messages with Yes, No & Cancel buttons
64 Show messages with Abort, Retry and Ignore buttons
-1 Show All messages

Example: This example runs a simulation, then opens the output statistics module without prompting the user.

```
Sub SkipMessages()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetMessageMode 0
  pmObject.Simulate

  Set pmObject = Nothing
End Sub
```

SetPan

Syntax: SetPan x, y
Description: Moves the layout view so that the x-y coordinates specified are at the top left corner of the layout window (as if the user had moved the scroll bars).
Parameters:
x (Double) Horizontal shift in pixels.
y (Double) Vertical shift in pixels.
Returns: Nothing
Example: This example loads a model, then scrolls right 100 pixels and down 100 pixels.

```
Sub AutoScroll()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetPan 100, 100

  Set pmObject = Nothing
End Sub
```

SetView

Syntax: SetView ViewName
Description: Sets the layout window to display a specific view.
Parameters:
ViewName (String) The name of the view you wish to display.
Returns: Nothing
Example: This example loads a model, then switches to the defined view.

```
Sub ShowFull()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetView "Full View"

  Set pmObject = Nothing
End Sub
```

SetViewRect

Syntax: SetViewRect Left, Top, Right, Bottom
Description: Zooms and pans the layout to allow the rectangle defined by the input coordinates to fill the layout window. Note: The x,y dimensions are always scaled equally.
Parameters:

Left (Double) The left boundary.
Top (Double) The top boundary.
Right (Double) The right boundary.
Bottom (Double). The bottom boundary.

Example: This example loads a model, then sets the layout window to show the section from x-y coordinates (30, 40) to (100, 200).

```
Sub NewPosition()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetViewRect 30, 40, 100, 200

  Set pmObject = Nothing
End Sub
```

SetWindowPos

Syntax: SetWindowPos Window, xPos, yPos, xSize, ySize, Flags
Description: Defines the position, size and style of the application window or the layout window. The application window position is relative to the screen, other layout window is relative to the application window.
Parameters:

Window (Long) Identifies the window (1 = layout window, 2 = application window).
xPos, yPos (Long) Determine the position of the window.
xSize, ySize (Long) Determine the size of the window.
Flags (Long) If xSize or ySize is zero, SetWindowPos calls the Windows API function ShowWindow and passes the flags.

Window Types:
1 Layout window
2 Application window

Flags:

-1 Original Style
0 Hide
1 Normal
2 Minimize
3 Maximize
9 Restore

Returns: Nothing
Example: This example loads a model, ensures that ProModel is running maximized, then moves the layout window near the top left corner of the ProModel screen.

```
Private Sub CommandButton2_Click()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.SetWindowPos 2, 0, 0, 0, 0, 3
  pmObject.SetWindowPos 1, 10, 10, 700, 500, 0

  Set pmObject = Nothing
End Sub
```

ShowTranslationDlg

Syntax: ShowTranslationDlg Status
Description: Shows or hides the translation dialog.
Parameters:
Status (Boolean) True displays the dialog, False hides the dialog.
Returns: Nothing
Example: This example loads a model, sets the option to hide the Translation dialog, then runs the simulation.

```
Sub HideIt()
  Dim pmObject As ProModel.CProModel

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.ShowTranslationDlg False
  pmObject.Simulate

  Set pmObject = Nothing
End Sub
```

Simulate

Syntax: Simulate
Description: Starts the simulation.
Parameters: None
Returns: Nothing
Example: This example loads a model, runs it for 30 minutes (1800 seconds), then terminates the simulation.

```
Sub SimIt()
  Dim pmObject As ProModel.CProModel
  Dim CurSimTime AS Double

  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel ("mfg_cost.mod")
  pmObject.Simulate
  CurSimTime = 0
  Do While CurSimTime < 1800
  pmObject.GetSimTime CurSimTime
  DoEvents
  Loop
  pmObject.EndSimulation

  Set pmObject = Nothing
End Sub
```

Zoom

Syntax: Zoom PercentZoom
Description: Sets the zoom percentage of the model. Passing a zero (0) in the *PercentZoom* parameter shrinks or expands the model to fit the layout window.
Parameters:
PercentZoom (Double) 1 = 100%, 0.75 = 75%, etc.
Returns: Nothing
Example: This example loads a model, then shows the layout 90 % of the original size.

```
Sub ShrinkIt()
  Dim pmObject As ProModel.CProModel
  Dim ZoomPercent As Double

  ZoomPercent = 0.9
  Set pmObject = CreateObject("ProModel")
  pmObject.LoadModel "mfg_cost.mod"
  pmObject.Zoom ZoomPercent

  Set pmObject = Nothing
End Sub
```

The ProModel Events Object

The PMEvents object is found in the *ProModel Events 8.0 Type Library*. This object provides methods for trapping and responding to certain events that happen in ProModel. The PMEvents object must be initialized by the GetEventsObject method found in the ProModel Application object. In Visual Basic, you must declare this object variable using the WithEvents keyword. When this is done, Visual Basic will automatically create a new entry in the Object drop-down list box in the module where the object variable was declared (you do not need to manually declare the Event Handler function). Within the Event Handler function, you will place code to respond to events returned by ProModel to your application. You may modify the default behavior of some events by returning 'True' from your Event Handler method.

- PmEventsHandler
- AnimSpeedChange
- InputTextPrompt
- ListSelectPrompt
- RunError
- TranslationError
- UserZoom

PMEventsHandler

To use the ProModel Events Handler, add a reference to the ProModel Events 8.0 Type Library (**pmcp80.dll** typically found in your "C:\Program Files\Common Files\ProModel Corporation\Components" folder). Then, add the following to the Class Module[1] where you will be calling the GetEvents method of the ProModel Object.

Private WithEvents pmEventsObject As PMCPLib.PmEvents

That will place a **pmEventsObject** variable in the Object drop-down list, and a procedure called **PmEventsHandler** in the Procedure/Events drop-down list at the top of the Code Window.

When you select the object variable (pmEventsObject) from the Object drop-down list and PmEventsHandler from the Procedure/Events drop-down list, it will create a function called **pmEventsObject_PmEventsHandler** in your Code Window. In this function, you will create "If…Then" or "Select Case" statements to handle event codes returned to your program by ProModel.

A list of Events that the PMEventsHandler handles can be found on Locations Table (1): PMD_Table_Location on page 108.

Example:

```
Private Function pmEventsObject_PmEventsHandler(ByVal PmEventID As Long, ByVal Description As String) As
Long
  Select Case PmEventID
    Case 2: 'PmEvtSaveBeforeQuit
      pmEventsObject_PmEventsHandler = True skip save
    Case Else:
      Debug.Print PmEventID & " " & Description show event in immediate 'window
  End Select
End Function
```

To activate the event handler in your program, include the ProModel Object GetEvents method at the beginning of your code.

Example:

```
Private Sub CommandButton1_Click()
```

[1] class module: A module that contains the definition of a class, including its property and method definitions.

```
    Dim pmObject As ProModel.CProModel

    Set pmObject = CreateObject("ProModel")
  Set pmEventsObject = pmObject.GetEventsObject
  …
```

Once the Events Object has been “Set”, as above, it will work on its own. So, any event that you want to “handle” differently than the default ProModel action, you will place the code for it in the pmEventsObject_PmEventsHandler function.

AnimSpeedChange

Syntax: *ObjectVariable*_AnimSpeedChange(ByVal Speed As Long) As Long
Description: This event occurs whenever animation speed is changed by the user, by ANIMATE command encountered in model code, or via COM interface (VB program code).
Parameters:

Speed (Long) – Integer from 0 through 100 representing the relative setting of the animation speed control. This number corresponds to the values used in ANIMATE statements in models and the values set and returned by SetAnimationSpeed and GetAnimationSpeed methods of the Runtime object.

Returns: Not used yet.
Example: The code below will place descriptive text and the new animation speed into a textbox each time the animation speed is changed (e.g. – "Animation speed changed to 39").

```
Private Function pmEventsObject_AnimSpeedChange(ByVal Speed As Long) As Long
  Textbox1.Text = Textbox1.Text & "Animation speed changed to " & Speed & VBCRLF
End Function
```

InputTextPrompt

Syntax: *ObjectVariable*_InputTextPrompt(ByVal Message As String, ByVal Default As String, Value As String, ByVal Flags As Long) As Long

Description: The PROMPT statement in ProModel can be defined either as a simple prompt/response dialog box or as a list of options to choose from. This event fires whenever a simple PROMPT statement is encountered in the model logic. Use the ListSelectPrompt event to handle PROMPT statements defined as choice lists.

Parameters:

Message (String) – Message to the user, describing the meaning and asking for a value

Default (String) – Text shown in the edit field of the dialog (before user changes it)

Value (String) – Variable to hold value given by user

Flags (Long) – Not Used

Returns:

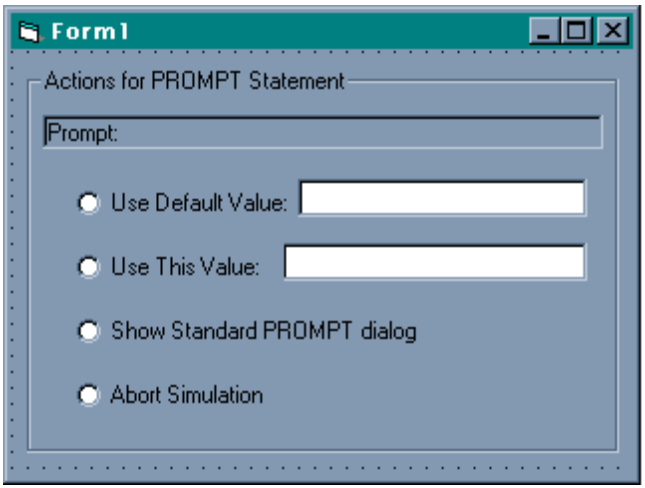
0 = Use the default value

1 = Set the variable to the value in *Value* parameter

2 = Show the prompt dialog and let the user enter a value

3 = Abort the simulation

Example: In the following example we start with the form shown below:




```
Private Function pmEventsObject_InputTextPrompt(ByVal Message As String, _
ByVal Default As String, Value As String, ByVal Flags As Long) As Long
    Label1.Caption = "Prompt: " & Message
    Textbox1.Text = Default
    If OptionButton(0).Value = True Then
      pmEventsObject_InputTextPrompt = 0 'Use default
```

```
    ElseIf OptionButton(1).Value = True Then
      Value = Textbox2.Text
      pmEventsObject_InputTextPrompt = 1 'Use value from textbox2
    ElseIf OptionButton(2).Value = True Then
      pmEventsObject_InputTextPrompt = 2 'Show standard prompt
    Else
      pmEventsObject_InputTextPrompt = 3 'Abort simulation
    End If
End Function
```

ListSelectPrompt

Syntax: *ObjectVariable*_ListSelectPrompt(ByVal Message As String, ByVal MenuItemCount As Long, psaLabels() As String, psaValues() As Double, Choice As Long, ByVal Flags As Long) As Long

Description: The PROMPT statement in ProModel can be defined either as a simple prompt/response dialog box or as a prompt dialog with a list of options to choose from. This event fires whenever a PROMPT statement defined as a choice list is encountered in the model logic. Use the InputTextPrompt event to handle simple PROMPT statements.

Parameters:

Message (String) – Message to the user, describing the meaning of parameter being prompted for, values provided, etc.

MenuItemCount (Long) – Number of items in menu

psaLabels – array of strings representing menu items

psaValues – array of real numbers (double) representing values that correspond to each menu item

Choice – zero-based index of the menu item to return to application

Flags (Long) – Not Used

Returns:

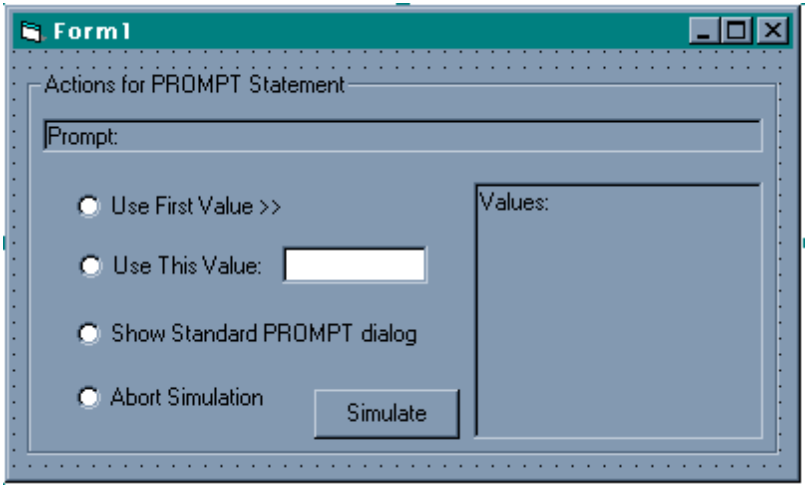
0 = Use the first value

1 = Set the variable to the value in *Choice* parameter

2 = Show the prompt dialog and let the user enter a value

3 = Abort the simulation

Example: In the following example we start with the form shown below:




```
Private Function pmEventsObject_ListSelectPrompt(ByVal Message As String, _
    ByVal MenuItemCount As Long, psaLabels() As String, _
    psaValues() As Double, Choice As Long, ByVal Flags As Long) As Long
  Dim i As Integer
```

```
  Label1.Caption = "Prompt: " & Message
  Textbox1.Text = ""
  For i = 0 To MenuItemCount - 1
    Textbox1.Text = Textbox1.Text & i & ") " & psaLabels(i) & " = " & _
      psaValues(i) & vbCrLf
  Next i

  If OptionButton(0).Value = True Then
    pmEventsObject_ListSelectPrompt = 0 'Use first value
  ElseIf OptionButton(1).Value = True Then
    Choice = Textbox2.Text
    pmEventsObject_ListSelectPrompt = 1 'Use value from textbox2
  ElseIf OptionButton(2).Value = True Then
    pmEventsObject_ListSelectPrompt = 2 'Show standard prompt
  Else
    pmEventsObject_ListSelectPrompt = 3 'Abort simulation
  End If

End Function
```

RunError

Syntax: *ObjectVariable*_RunError(ByVal ErrorID As Long, ByVal Description As String, ByVal Flags As Long) As Long
Description: This event fires whenever an error is encountered during simulation. This allows you to handle specific types of errors in different ways.
Parameters:

ErrorID (Long) – error code generated by ProModel
Description (String) – text description of error generated by ProModel
Flags (Long) – Type of error: 1 = Recoverable, 2 = Fatal

Returns:

0 = Ignore & continue with simulation, if possible
1 = Abort simulation
2 = Show error message & allow user to determine how to handle error

Example:

```
Private Function pmEventsObject_RunError(ByVal ErrorID As Long, ByVal Description As String, ByVal Flags As
Long) As Long


  Select Case Flags
    Case 1:
  pmEventsObject_RunError = 2
    Case 2:
  pmEventsObject_RunError = 1
  End Select


End Function
```

TranslationError

Syntax: *ObjectVariable*_TranslationError(ByVal ErrorID As Long, ByVal Description As String, ByVal Flags As Long) As Long
Description: This event occurs whenever an error is encountered during model translation (just before simulation begins).
Parameters:
ErrorID (Long) – error code generated by ProModel
Description (String) – text description of error generated by ProModel
Flags (Long) – Type of error: 1 = Recoverable, 2 = Fatal

Returns:
0 = Ignore & continue with translation
1 = Abort translation
2 = Show translation status dialog & allow user to determine how to handle error

Example:

```
Private Function PMEventObj_TranslationError(ByVal ErrorID As Long, ByVal
Description As String, ByVal Flags As Long) As Long

  Select Case Flags
    Case 1:
  pmEventsObject_RunError = 2
    Case 2:
  pmEventsObject_RunError = 1
  End Select

End Function
```

UserZoom

Syntax: *ObjectVariable*_UserZoom (ByVal ZoomPercent As Double) As Long
Description: Fired when zoom setting of the layout window is changed, either by explicit zoom or by changing views.
Parameters:
ZoomPercent (Double) – Factor by which the layout is magnified, 1 = 100%
Returns: Not used

Example:

```
Private Function PMEventObj_UserZoom(ByVal ZoomPct As Double) As Long
MsgBox("Zoomed to " & ZoomPct * 100 & "%")
End Function
```

The ProModel Runtime Object

The ProModel Runtime object is found in the *ProModel Type Library*. If you do not see this library in your references list, you will need to run ProModel.exe (or MedModel.exe or ServiceModel.exe) with the "/regserver" command line option.

Using the ProModel Runtime object you can access the statistics generated while a simulation is running, similar to using the Dynamic Plots from the ProModel interface. The methods for this object not only allow you to get the statistics values, but you can set new values for variables while the simulation is running.

The Runtime object has the following methods:

- GetAnimationState
- GetAnimationSpeed
- GetIntArrayValues
- GetRealArrayValues
- GetStatValue
- SetAnimationState
- SetAnimationSpeed
- SetIntArrayValues
- SetRealArrayValues
- SetStatValue

GetAnimationState

Syntax: GetAnimationState State

Description: Gets the current state of the animation, whether ON or OFF.

Parameters:

State (Long - Output Parameter) Zero (0) if animation is off, one (1) if animation is on.

Returns: Long - Zero (0) if animation is off, one (1) if animation is on.

Example:

```
Dim AnimState as Long
Dim pmRun as Promodel.CRuntime

pmObj.Simulate
pmRun.GetAnimationState AnimState
MsgBox "Animation is " & IIF(AnimState = 0, "off", "on")

Set pmObj = nothing
Set pmRun = nothing
```

GetAnimationSpeed

Syntax: GetAnimationSpeed Speed
Description: Gets the current value … When using GetAnimationSpeed and SetAnimationSpeed, the numeric value of a given speed may be slightly different for each method (because of rounding).
Parameters:
Speed (Long - Output Parameter) Numeric value representing animation speed (1 to 100).
Returns: Long - Numeric value representing animation speed (1 to 100).
Example:

```
Dim AnimSpeed as Long
Dim pmRun as Promodel.CRuntime

pmObj.Simulate
pmRun.GetAnimationSpeed AnimSpeed
MsgBox "Animation Speed is " & AnimSpeed

Set pmObj = nothing
Set pmRun = nothing
```

GetIntArrayValues

Syntax: GetIntArrayValues
Description: Gets the values on an integer array during simulation runtime
Parameters:

Array Name (String) The name of integer array.

Subset (String) A selection of elements from the integer array. The syntax is the same as the syntax used in VBA to index an array. For instance, to get elements 1 to 10 from an array set Subset to "1 to 10". For a two-dimentional array you would use "1 to 10, 1 to 10".

pvar (variable) An array of integers taken from the specified ProModel array.

Returns: An array of integers taken from a ProModel array during runtime.

Example:This example launches a simulation and displays part of the contents of an integer array after 1 hour of simulation time has elapsed.

```
Sub GetIntegerArrayValues()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim curSimTime As Double

  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("ProModel.Cruntime")
  pmObject.LoadModel "Tanker.mod"
  pmObject.Simulate
  curSimTime = 0
  Do While curSimTime < 3600
    pmObject.GetSimTime curSimTime
    DoEvents
  Loop


  Dim intArray
  Dim n As Long
  Dim arrayContents As String


  pmRuntime.GetIntArrayValues "Tank_State", "1 to 10", intArray
  arrayContents = "Array Contents: " & Chr(13)
  For n = 1 To 10
  arrayContents = arrayContents & intArray(n) & Chr(13)
  Next n
  MsgBox arrayContents
End Sub
```

GetRealArrayValues

Syntax: GetRealArrayValues
Description: Gets the values of a real array during simulation runtime
Parameters:

Array Name (String) The name of the real array.

Subse t (String) A selection of elements from the real array. The syntax is the same as the syntax used in VBA to index an array. For instance, to get elements 1 to 10 from an array set Subset to "1 to 10". For a two-dimentional array you would use "1 to 10, 1 to 10".

pvar (variable) An real array taken from the specified ProModel array.

Returns: An real array taken from a ProModel array during runtime.

Example:This example launches a simulation and displays part of the contents of a real array after 1 hour of simulation time has elapsed.

```
Sub GetRealArrayValues()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim curSimTime As Double

  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("ProModel.Cruntime")
  pmObject.LoadModel "Tanker.mod"
  pmObject.Simulate
  curSimTime = 0
  Do While curSimTime < 3600
    pmObject.GetSimTime curSimTime
    DoEvents
  Loop


  Dim realArray
  Dim n As Long
  Dim arrayContents As String

  pmRuntime.GetRealArrayValues "Tank_State", "1 to 10", realArray
  arrayContents = "Array Contents: " & Chr(13)
  For n = 1 To 10
    arrayContents = arrayContents & realArray(n) & Chr(13)
  Next n
  MsgBox arrayContents
End Sub
```

GetStatValue

Syntax: GetStatValue TableID, RecordNum, StatID, SubRec, Value
Description: Gets the current value for the specified statistics element.
Parameters:

TableID (Long) Table (Section) Number containing the data element you wish to retrieve.
RecordNum (Long) Record (Row) Number containing the data element you wish to retrieve.
StatID (Long) Statistic element you wish to retrieve.
SubRec (Long) For Resource Stats > Unit Number (0 = parent); For Node Entry Stats > Node Index; For Failed Arrivals > RecordNum = Entity Index, SubRec = Location Index
Value (Double) Variable or object in which to place the statistics value retrieved.

Returns: The numeric value of the statistic element at the moment the method is executed. The value will be placed in the variable or object specified in the *Value* parameter.
Example: This example loads Receive.mod, runs the simulation, then pauses 3 times at 30 minute intervals and displays the Current Contents of the Pallet_Storage location, then ends the simulation.

```
Sub ShowSomeStats()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim CurSimTime As Double
  Dim MyStatVal As Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("Promodel.CRuntime")
  pmObject.LoadModel ("receive.mod")
  pmObject.Simulate

  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop

  pmObject.MenuCommand 4816, 0
  pmRuntime.GetStatValue 1, 21, 5, MyStatVal
  pmObject.MsgBox MyStatVal
  pmObject.MenuCommand 4816, 0

  Do While CurSimTime < 3600
    pmObject.GetSimTime CurSimTime
    DoEvents
```

```
    Loop

    pmObject.MenuCommand 4816, 0 'Pauses Simulation
    pmRuntime.GetStatValue 1, 21, 5, MyStatVal
    pmObject.MsgBox MyStatVal
    pmObject.MenuCommand 4816, 0 'Resumes Simulation

    Do While CurSimTime < 5400
      pmObject.GetSimTime CurSimTime
      DoEvents
    Loop

    pmObject.MenuCommand 4816, 0
    pmRuntime.GetStatValue 1, 21, 5, MyStatVal
    pmObject.MsgBox MyStatVal
    pmObject.MenuCommand 4816, 0

    pmObject.EndSimulation
    Set pmObject = Nothing
End Sub
```

SetAnimationState

Syntax: SetAnimationState State

Description: Sets the state of the animation to be either on or off.

Parameters:

State (Long) Zero (0) to turn animation off, one (1) to turn animation on.

Returns: Nothing

Example:

```
Dim pmRun as Promodel.CRuntime

pmObj.Simulate
pmRun.SetAnimationState 0 'Off

Set pmObj = nothing
```

SetAnimationSpeed

Syntax: SetAnimationSpeed Speed

Description: Sets the animation speed. When using GetAnimationSpeed and SetAnimationSpeed, the numeric value of a given speed may be slightly different for each method (because of rounding).

Parameters:

Speed (Long) Numeric value representing animation speed (1 to 100).

Returns: Nothing

Example:

```
Dim pmRun as Promodel.CRuntime

pmObj.Simulate
pmRun.SetAnimationState 50

Set pmObj = nothing
Set pmRun = nothing
```

SetIntArrayValues

Syntax: SetIntArrayValues
Description: Sets the value of an integer array during simulation runtime.
Parameters:

ArrayName (String) The name of the integer array.

Subset (String) A selection of elements from the integer array. The syntax is the same as the syntax used in VBA to index an array. For instance, to get elements 1 to 10 from an array set Subset to "1 to 10". For a two-dimensional array you would use "1 to 10, 1 to 10".

psa (variable) An array of integers to be injected into the specified ProModel array.

Returns: Nothing

Example:

This example launches a simulation and displays part of the contents of an integer array after 1 hour of simulation time has elapsed. Notice that the first thre entries have been set to 1.

```
Sub SetIntegerArrayValues()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim CurSimTime As Double

  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("Promodel.CRuntime")
  pmObject.LoadModel "Tanker.mod"
  pmObject.Simulate
  curSimTime = 0
  Do While curSimTime < 3600
    pmObject.GetSimTime curSimTime
    DoEvents

  Loop
  Dim intArray
  Dim n As Long
  Dim arrayContents As String
  Dim inVals(3) As Long

  inVals(1) =1
  inVals(2) =1
  inVals(3) =1
  pmRuntime.SetIntArrayValues "Tank_State", "1 to 10", invVals
  pmRuntime.GetIntArrayValues "Tank_State", "1 to 10", intArray
  arrayContents = "Array Contents:" & Chr(13)
```

```
    Next n
    MsgBox arrayContents
End Sub
```

SetRealArrayValues

Syntax: SetRealArrayValues
Description: Sets the values of a real array during simulation runtime.
Parameters:

ArrayName (String) The name of the real array.

Subset (String) A selection of elements from the real array. The syntax is the same as the syntax used in VBA to index an array. For instance, to get elements 1 to 10 from an array set Subset to "1 to 10". For a two-dimensional array you would use "1 to 10, 1 to 10".

psa (variable) A real array to be injected into the specified ProModel array.

Returns: Nothing

Example:

This example launches a simulation and displays part of the contents of a real integer array after 1 hour of simulation time has elapsed. Notice that the first thre entries have been set to 1.

```
Sub SetRealArrayValues()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim CurSimTime As Double

  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("Promodel.CRuntime")
  pmObject.LoadModel "Tanker.mod"
  pmObject.Simulate
  curSimTime = 0
  Do While curSimTime < 3600
    pmObject.GetSimTime curSimTime
    DoEvents

  Loop
  Dim realArray
  Dim n As Long
  Dim arrayContents As String
  Dim inVals(3) As Double

  inVals(1) =0.1
  inVals(2) =0.1
  inVals(3) =0.1
  pmRuntime.SetRealArrayValues "Tank_State", "1 to 10", invVals
  pmRuntime.GetRealArrayValues "Tank_State", "1 to 10", realArray
```

```
    arrayContents = "Array Contents:" & Chr(13)
    Next n
    MsgBox arrayContents
End Sub
```

SetStatValue

Syntax: SetStatValue TableID, RecordNum, SubRec, StatID, NewVal

Description: Changes the current value for the specified statistics element. To set variable values, use TableID = 12, StatID = 5.

Parameters:

TableID (Long) Table (Section) Number containing the data element you wish to set.

RecordNum (Long) Record (Row) Number containing the data element you wish to set.

StatID (Long) Statistic element you wish to set.

SubRec (Long)

For Resource Stats: SubRec = Unit Number (parent = 0)

For Node Entry Stats: SubRec = Node Index;

For Failed Arrivals: RecordNum = Entity Index, SubRec = Location Index

For all others: SubRec = 0

NewVal (Double) New value for the specified statistics element.

Returns: Nothing.

Example:

This example loads Receive.mod, runs the simulation, sets the value variable 1 to zero after 30, 60 and 90 minutes of simulation, then ends the simulation.

```
Sub ResetStats()
  Dim pmObject As ProModel.CProModel
  Dim pmRuntime As ProModel.CRuntime
  Dim CurSimTime As Double

  CurSimTime = 0
  Set pmObject = CreateObject("ProModel")
  Set pmRuntime = CreateObject("Promodel.CRuntime")
  pmObject.LoadModel ("receive.mod")
  pmObject.Simulate

  Do While CurSimTime < 1800
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop
  pmRuntime.SetStatValue 12, 1, 5, 0, 0

  Do While CurSimTime < 3600
    pmObject.GetSimTime CurSimTime
```

```
    DoEvents
  Loop
  pmRuntime.SetStatValue 12, 1, 5, 0, 0

  Do While CurSimTime < 5400
    pmObject.GetSimTime CurSimTime
    DoEvents
  Loop
  pmRuntime.SetStatValue 12, 1, 5, 0, 0

  pmObject.EndSimulation
  Set pmObject = Nothing
End Sub
```

The ProModel Data Object

The ProModel Data object is found in the *ProModel Type Library*. If you do not see this library in your references list, you will need to run ProModel.exe (or MedModel.exe or ServiceModel.exe) with the "/regserver" command line option.

This object provides access to your model data, such as Locations, Subroutines, etc. The ProModel Data object must be initialized by the *Populate* method. With the methods listed below, you can read your model information, change it, even delete it. This is a powerful tool and must be used with caution. It is a good idea to always make a backup copy of your original model before using the ProModel Data object to make changes.

- AddBackgroundBitmap
- AppendEntitySpot
- AppendGraphicIcon
- AppendGraphicIconSize
- AppendRecord
- AppendRoutingPoint
- DeleteRecord
- GetIntFieldValue
- GetRealFieldValue
- GetRecordCount
- GetSelectedsFromType
- GetStringFieldValue
- InsertRecord
- Populate
- SelectMainRecordByIndex
- SelectMainRecordByName
- SetIntFieldValue
- SetRealFieldValue
- SetStringFieldValue

AddBackgroundBitmap

Syntax: AddBackgroundBitmap RecordType, Filename, Xpos, Ypos, Percent

Description: Adds the bitmap in the specified file to the background of the current model. When adding more than one bitmap, they are placed one on top of the other, so be sure to add the one(s) in back first. It is best to use the ProModel object's Zoom method to set the zoom to 100% before adding any graphics. You may also want to use the RedrawLayout method after adding new graphics (they may not show until you do).

Parameters:

RecordType (Long) **49 -** Background Graphics Table, is the only value allowed.

Filename (String) Path & filename of Bitmap, Windows Metafile or Enhanced Metafile.

Xpos (Double) How far (in pixels) from the left margin of the layout to place the left edge of the bitmap.

Ypos (Double) How far (in pixels) from the top margin of the layout to place the top edge of the bitmap

Percent (Double) Factor by which bitmap will be compressed or expanded from its original size. (1.0 = 100%)

Example: The following example creates a new model, ensures that the zoom is set to 100%, then adds a background bitmap.

```
Sub AddGfx()


  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmDataObject.Populate
  pmObject.Zoom

  pmDataObject.AddBackgroundBitmap 49, "pm_prac.bmp", 3, 3, 1
  pmObject.RedrawLayout

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

AppendEntitySpot

Syntax: AppendRecord RecordIndex, XPosition, YPosition
Description: Appends an Entity Spot graphic to an existing Location record.
Parameters:

RecordIndex (Long) The Record Index (Row Number) of the Location you wish to add an Entity Spot to.
XPosition (Double) Number of pixels over from the left margin of the layout to place the entity spot.
YPosition (Double) Number of pixels down from the top margin of the layout to place the entity spot.

Example: This example loads the mfg_cost model, adds a record to the Locations table, names the new location & gives it an Entity Spot graphic icon.

```
Sub AddGfx()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim recnum As Long
  Dim fieldnum As Long
  Dim subtable As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.AppendRecord 1
  pmDataObject.GetRecordCount 1, recnum
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"
  pmDataObject.AppendEntitySpot recnum, 45, 90

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

Note: An entity spot can also be added by appending a location graphic record and changing the type field to Enitity Spot (type 7). See Working with Graphics on page 7.

AppendGraphicIcon

Syntax: AppendGraphicIcon RecordType, RecordIndex, GraphicNumber, XPos, YPos
Description: Appends a graphic to an existing Location, Entity or Resource record, without changing its size.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.
RecordIndex (Long) The Record Index (Row Number) of the Location, Entity or Resource you wish to add a graphic icon to.
GraphicNumber (Long) Index number of a graphic icon in the Graphics Library specified in the General Information dialog of the loaded model. Keep in mind that the numbers are not necessarily sequential, there may be gaps.
XPos (Double) Number of pixels from the left margin to place the graphic icon.
YPos (Double) Number of pixels from the top margin to place the graphic icon.

Example:This example loads a model, adds a record to the Locations table, then appends a graphic icon to the new record.

```
Sub AddGfx()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim recnum As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmDataObject.Populate


  pmObject.LoadModel ("mfg_cost.mod")
  pmDataObject.AppendRecord 1
  pmDataObject.GetRecordCount 1, recnum
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"
  pmDataObject.AppendGraphicIcon 1, recnum, 10, 100, 50

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

AppendGraphicIconSize

Syntax: AppendGraphicIconSize RecordType, RecordIndex, GraphicNumber, Xpos, Ypos, Xsize, Ysize
Description: Appends a graphic to an existing Location, Entity or Resource record. This method lets you specify the size of the graphic as a percentage of its original size.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.
RecordIndex (Long) The Record Index (Row Number) of the Location, Entity or Resource you wish to add a graphic icon to.
GraphicNumber (Long) Index number of a graphic icon in the Graphics Library specified in the General Information dialog of the loaded model. Keep in mind that the numbers are not necessarily sequential, there may be gaps.
Xpos (Double) Number of pixels from the left margin to place the graphic icon.
Ypos (Double) Number of pixels from the top margin to place the graphic icon.
Xsize (Double) Width in pixels.
Ysize (Double) Height in pixels.

Example: This example loads a model, adds a record to the Locations table, then appends a graphic icon to the new record and makes it 50 x 50 pixels.

```
Sub AddGfx()
  Dim pmObject As ProModel.CProModel
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim recnum As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmDataObject.Populate

  pmObject.LoadModel ("mfg_cost.mod")
  pmDataObject.AppendRecord 1
  pmDataObject.GetRecordCount 1, recnum
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"
  pmDataObject.AppendGraphicIconSize 1, recnum, 10, 100, 50, 50, 50

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

AppendRecord

Syntax: AppendRecord RecordType
Description: Adds a new record to the **end** of the specified table. When using AppendRecord, you may use the *Set…FieldValue* methods without first calling the *SelectRecordBy…* method. However, if you are using a method that requires the Record Index number as a parameter, you will need to get that number with the *GetRecordCount* method (or any other method that returns that information).
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to add a record to.

Example: This example loads the mfg_cost model, adds a new record to the Locations table, and then changes the location name to My_New_Loc.

```
Sub AppRec()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel ("mfg_cost.mod")
  pmDataObject.Populate
  pmDataObject.AppendRecord 1
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

AppendRoutingPoint

Syntax: AppendRoutingPoint XPosition, Yposition
Description: Adds a new Routing Point and connects the previous end point to the new point. You must first select the Processing and Routing records you wish to append the new Routing Point to.
Parameters:

Xposition (Double) Number of pixels from the left margin of the layout to place the Routing Point.

Yposition (Double) Number of pixels from the top margin of the layout to place the Routing Point.

Example: The following example loads the mfg_cost model, selects the first processing record, then selects the second routing record for that process and adds a new point.

```
Sub AddAPoint()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  PmObject.LoadModel “mfg_cost.mod”
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByIndex 19, 1
  pmDataObject.SelectMainRecordByIndex 20, 2
  pmDataObject.AppendRoutingPoint 125, 90

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

DeleteRecord

Syntax: DeleteRecord RecordType
Description: Deletes the currently selected record from the specified table. Be very careful when using this method, because it will delete records without any data integrity checking. This means that even if the record is referenced by a record in another table, it will be deleted and make the other record invalid.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to delete a record from.

Example: The following example deletes all records from the Locations table in the mfg_cost model. Be careful with this one, or you could disable the mfg_cost demo model if you then save it.

```
Sub EmptyLocs()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim rec_count As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel ("mfg_cost.mod")
  pmDataObject.Populate

  pmDataObject.GetRecordCount 1, rec_count
  For x = 1 To rec_count
    pmDataObject.SelectMainRecordByIndex 1, 1
    pmDataObject.DeleteRecord 1
  Next x

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

GetIntFieldValue

Syntax: GetIntFieldValue RecordType, FieldNumber, FieldValue
Description: Returns the value of the specified integer field from the selected record.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.
FieldNumber (Long) The field (column) number in the specified table from which you want to return a value.
FieldValue (Long) Data value contained in the specified field for the selected record. This must be a variable.

Example: The following example gets the value of the Default Time units from the mfg_cost model, then displays a message with that information.

```
Sub GetTimeUnits()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim TimeUnit As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByIndex 18, 1
  pmDataObject.GetIntFieldValue 18, 2, TimeUnit

  Select Case TimeUnit
    Case 1:
      pmObject.MsgBox ("Default Time Units: Seconds")
    Case 2:
      pmObject.MsgBox ("Default Time Units: Minutes")
    Case 3:
      pmObject.MsgBox ("Default Time Units: Hours")
    Case 4:
      pmObject.MsgBox ("Default Time Units: Days")
    Case Else:
      pmObject.MsgBox ("Error")
  End Select


  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

GetRealFieldValue

Syntax: GetIntFieldValue RecordType, FieldNumber, FieldValue
Description: Returns the value of the specified real number field from the selected record.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.
FieldNumber (Long) The field (column) number in specified table from which you want to return a value.
FieldValue (Long) Data value contained in the specified field for the selected record. This must be a variable.

Example: The following example selects the second routing record (not necessarily the second routing block) of the first processing record, gets the Probability value and displays it in a message box.

```
Sub SetRoutingProbability()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim ProbValue As Double

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "Orders.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByIndex 19, 1
  pmDataObject.SelectMainRecordByIndex 20, 2
  pmDataObject.GetRealFieldValue 20, 13, ProbValue
  pmObject.MsgBox "Probability = " & ProbValue

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

GetRecordCount

Syntax: GetRecordCount RecordType, Count

Description: Returns the current number of records in the specified table. This is very useful in looping through each record in a table of unknown size.

Parameters:

RecordType (Long) The Record Type (Table Number) for which you want the record count.

Count (Long) Returns the number of records in the specified table.

Example: The following example loads the mfg_cost model, gets the record count from the Locations table, then displays a message telling the user how many locations there are.

```
Sub HowManyLocs()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim rec_count As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel ("mfg_cost.mod")
  pmDataObject.Populate

  pmDataObject.GetRecordCount 1, rec_count

  If rec_count = 1 Then
      pmObject.MsgBox "There is 1 location in this model"
    Else
      pmObject.MsgBox "There are " & rec_count & " locations in this model"
  End If

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

GetSelectedsFromType

Syntax: GetSelectedsFromType RecordType, MainIndex, Field, SubIndex
Description: Returns record index values for the currently selected main and sub records.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you want to work with.
MainIndex (Long) Returns Record Index (row number) of the currently selected record.
Field (Long) Although this parameter is no longer used, it is still required. It returns -1.
SubIndex (Long) Although this parameter is no longer used, it is still required. It returns -1.

Example: The following example loads the mfg_cost model, finds the "Inspect" location and displays a message telling the user what the index number is for that record.

```
Sub WhatsMyIndex()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim recnum As Long
  Dim fieldnum As Long
  Dim subrec As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByName 1, "Inspect"
  pmDataObject.GetSelectedsFromType 1, recnum, fieldnum, subrec
  pmObject.MsgBox "Location <Inspect> is Record Index #" & recnum

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

GetStringFieldValue

Syntax: GetStringFieldValue RecordType, FieldNumber, FieldValue
Description: Returns the value of the specified string field from the selected record.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you want to work with.

FieldNumber (Long) The field (column) number in specified table from which you want to return a value.

FieldValue (Long) Data value contained in the specified field for the selected record. This must be a variable.

Example: The following example steps through the Macros table, gets the name and value of each macro, then displays a message with that information.

```
Sub GetMacros()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim i As Integer
  Dim RecCount As Long
  Dim MacName As String
  Dim MacText As String

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.GetRecordCount 11, RecCount

  For i = 1 To RecCount
    pmDataObject.SelectMainRecordByIndex 11, i
    pmDataObject.GetStringFieldValue 11, 1, MacName
    pmDataObject.GetStringFieldValue 11, 2, MacText
    pmObject.MsgBox (MacName & ": " & MacText)
  Next i

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

InsertRecord

Syntax: InsertRecord RecordType
Description: Inserts a new record **before** the selected record in the table. The newly inserted record will already be selected, so there is no need to use the *SelectMainRecordBy…* methods before populating the data.
Parameters:
RecordType (Long) The Record Type (Table Number) of the table you wish to insert the record into.
Example: The following example loads the mfg_cost model, finds the "Inspect" location and inserts a new record before it, then selects the new record & gives it a name.

```
Sub AddALoc()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByName 1, "Inspect"
  pmDataObject.InsertRecord 1
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"
  pmObject.MsgBox "Done"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

Populate

Syntax: Populate

Description: Populates the ProModelData object with the current, loaded model data. Call this method before using the data object. You must call the *Populate* method to initialize the ProModelData object with the model's current information. Use *Populate* each time you open or close a model, or if you add or remove records.

Example: This example loads the mfg_cost model and populates the ProModelData object.

```
Sub AddALoc()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate
  pmObject.MsgBox "ProModelData object populated"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SelectMainRecordByIndex

Syntax: SelectMainRecordByIndex RecordType, RecordIndex

Description: Selects a record by its Record Index (row number). This method is very useful in looping through all the records in a table.

Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.

RecordIndex (Long) The Record Index (Row Number) of the record you wish to select.

Example: The following example steps through the Macros table, selecting each record by Index Number, then gets the name and value of each macro, then displays a message with that information.

```
Sub GetMacros()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim i As Integer
  Dim RecCount As Long
  Dim MacName As String
  Dim MacText As String

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.GetRecordCount 11, RecCount

  For i = 1 To RecCount
    pmDataObject.SelectMainRecordByIndex 11, i
    pmDataObject.GetStringFieldValue 11, 1, MacName
    pmDataObject.GetStringFieldValue 11, 2, MacText
    pmObject.MsgBox (MacName & ": " & MacText)
  Next i

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SelectMainRecordByName

Syntax: SelectMainRecordByName RecordType, RecordName
Description: Selects a record by its Name or ID value. This method can only be used with record types that have a name or ID field. If the table does not have a Name or ID field, you must use the *SelectMainRecordByIndex* method.
Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.

RecordName (Long) The Name or ID field value for the record (row) you wish to select.

Example: The following example loads the mfg_cost model, selects the location record by the name “Inspect”, inserts a new record before it, and gives it a name.

```
Sub AddALoc()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByName 1, "Inspect"
  pmDataObject.InsertRecord 1
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"
  pmObject.MsgBox "Done"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SetIntFieldValue

Syntax: SetIntFieldValue RecordType, FieldIndex, FieldValue

Description: Changes the value for the specified integer field of the selected record.

Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.

FieldIndex (Long) The Field Index (Column Number) of the field you wish to make changes to.

FieldValue (Long) The integer (long) value you want to place in the specified field.

Example: The following example loads the mfg_cost model, then sets its default time unit to Hours (it was Minutes).

```
Sub Time2Hours()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim TimeUnit As Long

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  TimeUnit = 3 '(Hours)
  pmDataObject.SelectMainRecordByIndex 18, 1
  pmDataObject.SetIntFieldValue 18, 2, TimeUnit

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SetRealFieldValue

Syntax: RecordType, FieldIndex, FieldValue

Description: Changes the value for the specified real field of the selected record. When setting values for real fields, it is best to place the number into a variable of the correct type, then use the variable in the method call. This will avoid incorrect numeric conversions.

Parameters:

RecordType (Long) The Record Type (Table Number) of the table you wish to work with.

FieldIndex (Long) The Field Index (Column Number) of the field for which you want to change the value.

FieldValue (Double) The real number value you wish to place in the specified field for the selected record.

Example: The following example loads the Orders model, selects the third processing record, then changes the probabilities for the routing records to 70/30. Be cautious with using this method to set probabilities or percentages that must total 100 (make sure the records still have the correct total after changes).

```
Sub GetRoutingProbability()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData
  Dim ProbVal1 As Double
  Dim ProbVal2 As Double

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")
  pmObject.LoadModel "Orders.mod"
  pmDataObject.Populate

  ProbVal1 = 0.7
  ProbVal2 = 0.3
  pmDataObject.SelectMainRecordByIndex 19, 3
  pmDataObject.SelectMainRecordByIndex 20, 1
  pmDataObject.SetRealFieldValue 20, 13, ProbVal1
  pmDataObject.SelectMainRecordByIndex 20, 2
  pmDataObject.SetRealFieldValue 20, 13, ProbVal2
  pmObject.MsgBox ("Done")

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

SetStringFieldValue

Syntax: SetStringFieldValue RecordType, FieldIndex, FieldValue

Description: Changes the value for the specified string field of the selected record.

Parameters:

RecordType(Long) The Record Type (Table Number) of the table you wish to work with.

FieldIndex (Long) The Field Index (Column Number) of the field in which you want to place the new value.

FieldValue (String) The value you wish to place in the specified field for the selected record.

Example: The following example loads the mfg_cost model, finds the "Inspect" location and inserts a new record before it, then selects the new record & gives it a name.

```
Sub AddALoc()
  Dim pmObject As ProModel.CProModel
  Dim pmDataObject As ProModel.CProModelData

  Set pmObject = CreateObject("ProModel")
  Set pmDataObject = CreateObject("ProModelData")

  pmObject.LoadModel "mfg_cost.mod"
  pmDataObject.Populate

  pmDataObject.SelectMainRecordByName 1, "Inspect"
  pmDataObject.InsertRecord 1
  pmDataObject.SetStringFieldValue 1, 2, "My_New_Loc"

  Set pmDataObject = Nothing
  Set pmObject = Nothing
End Sub
```

The RDBDataServer Object

To use the RDBDataServer object it must be registered, which should have done automatically during installation. This library is listed as RDBSrv in the references in your VB Editor.

As you work with the RDBDataServer, keep in mind that you are working with a database, not a text file or spreadsheet. This means that there is some information that you cannot get directly from the .rdb file. For example, when you have multiple replications, periods, or scenarios, you can't get the averages, standard deviations or totals using only the RDBDataServer methods. You must get the same data elements from each replication, period or scenario, then perform the calculations in your spreadsheet or other program.

When talking about tables, each section of the standard output statistics report corresponds to a table in the .rdb file. There are some tables that may or may not be available, depending upon whether that type of data is collected for the specific model (like Logs). However, the table numbers are pre-defined, so each number will always reference the same table, whether or not it is in use.

If you want to get the names of the records (i.e. – Location or Resource Names), you will need to get the data for field number 0 (zero). The zero column in each table has the Record ID from the model.

The RDBDataServer uses the following methods:

- CloseFile
- FieldName
- GetPositionInfo
- GetValue
- OpenFile
- PeriodName
- PositionIsValid
- RecordName
- ReplicationNumber
- ScenarioName
- SelectData
- TableName

CloseFile

Syntax: CloseFile
Description: Closes the previously opened RDB file.
Parameters: None
Returns: Nothing
Example: This example opens the mfg_cost.rdb, selects data, displays it, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim x

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile "mfg_cost.rdb"

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  x = RDBObj.GetPositionInfo
  MsgBox (x)

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

FieldName

Syntax: FieldName
Description: Returns the name of the current field.
Parameters: None
Returns: The name of the specified field (column), or “invalid” if the SelectData specifications are not valid.
Example: This example opens the mfg_cost.rdb, displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyField As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyField = RDBObj.FieldName
    MsgBox ("Field (Column): " & MyField)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

GetPositionInfo

Syntax: GetPositionInfo
Description: Returns the scenario, period, replication, table, field, record and data value of the current selection. This can be useful in loops.
Returns: Details of current data selection, as follows:

Scenario: *Scenario Name*
Replication: *Replication Number*
Period: *Period Name*
Table: *Table Name*
Field: *Field Name*
Record: *Record Name*
Value: *Data Value*

Parameters: None
Example: This example opens the mfg_cost.rdb, selects data, then displays the details of the selection.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim x

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile "mfg_cost.rdb"

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  x = RDBObj.GetPositionInfo
  MsgBox (x)

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

GetValue

Syntax: GetValue
Description: Returns the data value for the current data selection.
Parameters: None
Returns: The data value of the current position (or zero if the current selection is not valid).
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyDataVal As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyDataVal = RDBObj.GetValue
    MsgBox ("Data Value = " & MyDataVal)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

OpenFile

Syntax: OpenFile FileName
Description: Call this function first to open and load the proper .RDB file.
Parameters:
FileName (String) Path and filename of any valid .rdb file.
Returns: Nothing
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim x

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile "mfg_cost.rdb"

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  x = RDBObj.GetPositionInfo
  MsgBox (x)

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

PeriodName

Syntax: PeriodName
Description: Returns the name of the Period in the current data selection.
Parameters: None
Returns: The current period name, or “invalid” if the current selection is not valid.
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyPeriod As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyPeriod = RDBObj.PeriodName
    MsgBox ("Selected Period = " & MyPeriod)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

PositionIsValid

Syntax: PositionIsValid
Description: Check the data for the most recent data selection & return "True" if it is valid, "False" if it is not. This method is a little tricky, because it really doesn't do much unless it is used in an "If…Then" or "Select Case" statement (the 'x = RDBObj.PositionIsValid' format is not allowed)
Parameters: None
Returns: TRUE if the last SelectData function points to valid data. FALSE if it does not.
Example: This example opens the mfg_cost.rdb, selects data, checks to see if selection is valid, displays the data or an error message, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyPeriod As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyPeriod = RDBObj.PeriodName
    MsgBox ("Selected Period = " & MyPeriod)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

RecordName

Syntax: RecordName
Description: Returns the name of the Record in the current data selection.
Parameters: None
Returns: The current record name, or “invalid” if the current selection is not valid.
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyRecord As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyRecord = RDBObj.RecordName
    MsgBox ("Selected Record = " & MyRecord)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

ReplicationNumber

Syntax: ReplicationNumber
Description: Returns the number of the Replication in the current data selection.
Parameters: None
Returns: The current replication number, or “invalid” if the current selection is not valid.
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim RepNum As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    RepNum = RDBObj.ReplicationNumber
    MsgBox ("Selected Replication = " & RepNum)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

ScenarioName

Syntax: ScenarioName
Description: Returns the name of the Scenario in the current data selection.
Parameters: None
Returns: The current scenario name, or “invalid” if the current selection is not valid.
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyScenario As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyScenario = RDBObj.ScenarioName
    MsgBox ("Selected Scenario = " & MyScenario)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

SelectData

Syntax SelectData Scenario, Replication, Period, Table, Field, Record

Description: Retrieves the specified data element from the .rdb file. All of the parameters must be specified, even if there is only one scenario, replication or period.

Parameters:

Scenario (Long) Scenario number containing the data you wish to select.

Replication (Long) Replication number containing the data you wish to select.

Period (Long) Period number containing the data you wish to select.

Table (Long) Table (report section) number containing the data you wish to select.

Field (Long) Field (column) number containing the data you wish to select.

Record (Long) Record (row) number containing the data you wish to select.

Returns: TRUE if there is valid data corresponding to the parameters.

Example: This example opens the mfg_cost.rdb, gets and displays the Scheduled Hours for the first listed location in the Locations section of the statistic report (first scenario, replication & period), then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyScenario As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyScenario = RDBObj.ScenarioName
    MsgBox ("Selected Scenario = " & MyScenario)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

TableName

Syntax: TableName
Description: Returns the name of the Table in the current data selection.
Parameters: None
Returns: The current table name, or “invalid” if the current selection is not valid.
Example: This example opens the mfg_cost.rdb, gets and displays some of the data, then closes the file.

```
Sub Get_Data()
  Dim RDBObj As Object
  Dim MyTable As String

  Set RDBObj = CreateObject("RDBDataServer")
  RDBObj.OpenFile("C:\ProModel 2001\Output\mfg_cost.rdb")

  RDBObj.SelectData 1, 1, 1, 1, 1, 1
  If RDBObj.PositionIsValid = True Then
    MyTable = RDBObj.TableName
    MsgBox ("Selected Table = " & MyTable)
  Else
    MsgBox ("Data Selection Invalid")
  End If

  RDBObj.CloseFile
  Set RDBObj = Nothing
End Sub
```

Appendix: Table Definitions

ProModelData Table Definitions

To help you determine the RecordType values, the pmconst.bas file defines the constants used in the examples. For the complete list of ProActiveX constants, click **here**. The following is a list of RecordType values, Visual Basic record type constants, and field types for all data tables.

Table Name	Visual Basic Constant	Value	Location of Table in ProModel
Locations	PMD_Table_Locations	1	Build > Locations
Entities	PMD_Table_Entities	2	Build > Entities
Path Networks	PMD_Table_PathNetworks	3	Build > Path Networks
Resources	PMD_Table_Resources	4	Build > Resources
Arrivals	PMD_Table_Arrivals	6	Build > Arrivals
Shift Assignment	PMD_Table_Shifts	7	Build > Shifts > Assign
Attributes	PMD_Table_Attributes	8	Build > Attributes
Variables	PMD_Table_Variables	9	Build > Variables
Arrays	PMD_Table_Arrays	10	Build > Arrays
Macros	PMD_Table_Macros	11	Build > Macros
Subroutines	PMD_Table_Subroutines	12	Build > Subroutines
Arrival Cycles	PMD_Table_ArrivalCycles	13	Build > More Elements > Arrival Cycles
Table Functions	PMD_Table_TableFunc-tions	14	Build > More Elements > Table Functions
User Distributions	PMD_Table_UserDistribs	15	Build > More Elements > User Distributions
External Files	PMD_Table_ExternalFiles	16	Build > More Elements > External Files
Streams	PMD_Table_Streams	17	Build > More Elements > Streams
General Information	PMD_Table_GenInfo	18	Build > General Information
Processing	PMD_Table_Processing	19	Build > Processing
Routings	PMD_Table_Routing	20	Build > Processing
Model Parameters	PMD_Table_ModelParams	21	Simulation > Model Parameters
Scenarios	PMD_Table_Scenarios	22	Simulation > Scenarios
Simulation Options	PMD_Table_SimOptions	23	Simulation > Options
Location Clock Downtimes	PMD_Table_LocClockDTs	25	Build > Locations > DTs… > Clock
Location Entry Downtimes	PMD_Table_LocEntryDTs	26	Build > Locations > DTs… > Entry
Location Usage Downtimes	PMD_Table_LocUsageDTs	27	Build > Locations > DTs… > Usage
Location Setup Downtimes	PMD_Table_LocSetupDTs	28	Build > Locations > DTs… > Setup
Resource Clock Downtimes	PMD_Table_ResClockDTs	31	Build > Resources > DTs… > Clock
Resource Usage Downtimes	PMD_Table_ResUsageDTs	32	Build > Resources > DTs… > Usage
Resource Work Search	PMD_Table_ ResWorkSearch	33	Build > Resources > Search… > Work

Resource Park Search	PMD_Table_ ResParkSearch	34	Build > Resources > Search… > Park
Resource Node Logic	PMD_Table_ResNodeLogic	35	Build > Resources > Logic…
Resource Points	PMD_Table_ResPoints	36	Build > Resources > Pts…
Shift Assignment Locations	PMD_Table_ShiftLocations	38	Build > Shifts > Assign > Locations
Shift Assignment Resources	PMD_Table_ShiftResources	39	Build > Shifts > Assign > Resources
Shift Assignment Files	PMD_Table_ShiftFiles	40	Build > Shifts > Assign > Shift Files
Subroutine Parameters	PMD_Table_ SubroutineParams	41	Build > Subroutines > Parameters…
Arrival Cycles Values	PMD_Table_ArrivalCycData	42	Build > More Elements > Arrival Cycles > Table…
Table Functions Values	PMD_Table_Func-tionSubtable	43	Build > More Elements > Table Functions > Table…
User Distributions Values	PMD_Table_User-DistribSubtable	44	Build > More Elements > User Distributions > Table…
Location Graphics	PMD_Table_Loca-tionGraphics	45	Not Visible
Resource Graphics	PMD_Table_ ResourceGraphics	46	Not Visible
Entity Graphics	PMD_Table_EntityGraphics	47	Not Visible
Background Graphics	PMD_Table_Back-groundGraphics	49	Build > Background Graphics
Model Defaults	PMD_Table_ModelDefaults	50	Simulation > Model Parameters
Path Networks Segments	PMD_Table_PathSegments	51	Build > Path Networks > Paths…
Path Networks Interfaces	PMD_Table_PathInterfaces	52	Build > Path Networks > Interfaces…
Path Networks Mapping	PMD_Table_PathMappings	53	Build > Path Networks > Mapping…
Path Networks Nodes	PMD_Table_PathNodes	54	Build > Path Networks > Nodes…
Views	PMD_Table_Views	55	View > Views > Define
Scenario Parameters	PMD_Table_Scen-arioParams	56	Simulation > Scenarios > (Add/Edit)
Path Networks Mapping Destin-ations	PMD_Table_PathMapDest	57	Build > Path Networks > Mapping… > Dest.
Work Search Locations	PMD_Table_ WorkSearchLocs	58	Build > Resources > Search… > Work > Loca-tion List
Park Search Nodes	PMD_Table_ParkNodes	59	Build > Resources > Search… > Park > Park-ing Node List
Variable Graphics	PMD_Table_Vari-ableGraphics	60	Build > Variables
Queue/Conveyor Joints	PMD_Table_QJoints	61	Build > Locations > {Right-click on Queue} > Add Joint

Table Field Types

Key:

I = Integer
S = String
R = Real

Locations Table (1): PMD_Table_Location

Field	Type	Constant	Description or Location in ProModel
2	S	PMD_Field_LocName	Build > Locations > Name
3	S	PMD_Field_LocCapacity	Build > Locations > Capacity
4	I	PMD_Field_LocUnits	Build > Locations > Units
9	I	PMD_Field_LocStats	Build > Locations > Stats
		Stats Rule 1 = None 'PMD_Rule_StatsNone' 2 = Basic or Summary 'PMD_Rule_StatsBasic' or 'PMD_Rule_StatsSummary' 3 = Time Series or By Unit 'PMD_Rule_StatsTimeSeries' or 'PMD_Rule_StatsByUnit'	
10	I	PMD_Field_LocIncoming	Build > Locations > Rules > Select Incoming Entities
		Incoming Rule 1 = Oldest by priority pmdRuleInEntOldest 2 = Not used 3 = Least available capacity pmdRuleInEntLeastAvail 4 = Random pmdRuleInEntRandom 5 = Last selected location pmdRuleInEntLastLoc 6 = Lowest attribute value pmdRuleInEntMinAttrib 7 = Highest attribute value pmdRuleInEntMaxAttrib	
11	S	PMD_Field_LocIncAttrib	Build > Locations > Rules > Select Incoming Entities > Attribute
12	I	PMD_Field_LocQOutput	Build > Locations > Rules > Queuing For Output
		Queue Rule 1 = No queuing pmdRuleQOutNone 2 = Firsit in, First out pmdRuleQOutFIFO 3 = Last in, First out pmdRuleQOutLIFO 4 = Highest attribute value pmdRuleQOutMaxAttrib 5 = Lowest attribute value pmdRuleQOutMinAttrib 6 = By type pmdRuleQOutByType	
13	S	PMD_Field_LocQOutAttrib	Build > Locations > Rules > Queuing For Output > Attribute

14	I	PMD_Field_LocSelectUnit	Build > Locations > Rules > Selecting A Unit
		Unit Selection Rule 1 = Longest empty pmdRuleUSelLongestEmpty 2 = Random pmdRuleUSelRandom 3 = By turn pmdRuleUSelByTurn 4 = Most available capacity pmdRuleUSelMostAvail 5 = Fewest entries pmdRuleUSelLeastEntries 6 = First available pmdRuleUSelFirstAvail	
15	S	PMD_Field_LocNotes	Build > Locations > Notes
16	S	PMD_Field_LocCostRate	Operation cost per time unit in field 20. Build > Cost > Locations > Operation Rate
20	I	PMD_Field_LocCostTimeUnits	Build > Cost > Locations > Per
		Time Units Rule 1 = Seconds pmdTimeUnitSec 2 = Minutes pmdTimeUnitMin 3 = Hours pmdTimeUnitHr 4 = Days pmdTimeUnitDay	

Location Clock Downtimes Subtable (25): PMD_Table_LocClockDTs

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_DTFrequency	Frequency
2	S	PMD_Field_DTFirstTime	First Time
3	S	PMD_Field_DTPriority	Priority
4	I Bool	PMD_Field_DTScheduled	Scheduled
5	S	PMD_Field_DTLogic	Logic
6	I Bool	PMD_Field_DTDisable	Disable

Location Entry Downtimes Subtable (26): PMD_Table_LocEntryDTs

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_DTFrequency	Frequency
2	S	PMD_Field_DTFirstTime	First Time
5	S	PMD_Field_DTLogic	Logic
6	I Bool	PMD_Field_DTDisable	Disable

Location Usage Downtimes Subtable (27): PMD_Table_LocUsageDTs

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_DTFrequency	Frequency
2	S	PMD_Field_DTFirstTime	First Time
3	S	PMD_Field_DTPriority	Priority
5	S	PMD_Field_DTLogic	Logic
6	I Bool	PMD_Field_DTDisable	Disable

Location Setup Downtimes Subtable (28): PMD_Table_LocSetupDTs

Field	Type	Constant	Description or Location in ProModel
5	S	PMD_Field_DTLogic	Logic
6	I Bool	PMD_Field_DTDisable	Disable
7	S	PMD_Field_DTEntity	Entity
8	S	PMD_Field_DTPriorEnt	Prior Entity

Location Called Downtimes Subtable (63): PMD_Table_LocCalledDTs

Field	Type	Constant	Description or Location in ProModel
3	S	PMD_Field_DTPriority	Priority
4	I Bool	PMD_Field_DTScheduled	Scheduled
5	S	PMD_Field_DTLogic	Logic
11	S	PMD_Field_DTName	Name

Location Graphics (45): PMD_Table_LocGraphics

Field	Graphic Type Table	Constant	Description or Location in ProModel
1	**1**	PMD_GfxType_Library	Library
1	**2**	PMD_GfxType_Q	Queue / Conveyor
1	**3**	PMD_GfxType_Tank	Gauge / Tank
1	**4**	PMD_GfxType_Counter	Counter
1	**5**	PMD_GfxType_Text	Text
1	**6**	PMD_GfxType_Status	Status Light
1	**7**	PMD_GfxType_EntSpot	Entity Spot
1	**8**	PMD_GfxType_Region	Region

Queue/Conveyor Joints Table (61): pmdTblQJoint

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldQJointXPos	(Distance, in pixels, from Left margin)
2	I	pmdFldQJointYPos	(Distance, in pixels, from Top margin)

Entities Table (2): PMD_Table_Entity

Field	Type	Constant	Description or Location in ProModel
2	S	PMD_Field_EntName	Build > Entities > Name
3	S	PMD_Field_EntSpeed	Build > Entities > Speed
4	I	PMD_Field_EntStats	Build > Entities > Stats
		Stats Rule 1 = None pmdStatsTypeNone 2 = Basic or Summary pmdStatsTypeBasicSum 3 = Time Series or By Unit pmdStatsTypeTSUnit	
5	S	PMD_Field_EntNotes	Build > Entities > Notes
8	S	PMD_Field_EntInitCost	Build > Cost > Entities > Initial Cost

Entity Graphics (47): PMD_Table_EntGraphics

Field	Type	Constant	Description or Location in ProModel
1	**1**	PMD_GfxType_Library	Library

Path Networks (3): PMD_Table_PathNetworks

Field	Type	Constant	Description or Location in ProModel
1	I	PMD_Field_PathColor	Build > Path Networks > Graphics > Color
2	I Bool	PMD_Field_PathVisible	Build > Path Networks > Graphics > Visible
3	S	PMD_Field_PathName	Build > Path Networks > Name
4	I	PMD_Field_PathType	Build > Path Networks > Type
		Type Rule 0 = No Passing pmdPathTypeNoPass 1 = Speed pmdPathTypePass 2 = Crane pmdPathTypeCrane	
5	I	PMD_Field_PathBasis	Build > Path Networks > T/S
		Basis Rule 0 = Time pmdPathBasisTime 1 = Speed & Distance pmdPathBasisSpeedDist	
20	S	PMD_Field_CraneBridgeSeparation	Build>Path Networks>Cranes>Separation
21	I Bool	PMD_Field_CraneHideBridge	Build>Path Networks>Cranes>HideBridge
22	I Bool	PMD_Field_CraneHideRail	Build>Path Networks>Cranes>HideRail
23	I	PMD_Field_CraneRailWidth	Build>Path Networks>Cranes>RailWidth

24	I	PMD_Field_CraneRailBorder-Color	Build>Path Networks>Cranes>RailBorderColor
25	I	PMD_Field_CraneRailFillColor	Build>Path Networks>Cranes>RailFillColor
26	I	PMD_Field_CraneBridgeWidth	Build>Path Networks>Cranes>BridgeWidth
27	I	PMD_Field_ CraneBridgeBorderColor	Build>Path Networks>Cranes>BridgeBorderColor
28	I	PMD_Field_CraneBridgeFillCo-lor	Build>Path Networks>Cranes>BridgeFillColor

Path Segments Subtable (51): PMD_Table_PathSegments

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_PathSegFrom	Build > Path Networks > Paths… > From
2	S	PMD_Field_PathSegTo	Build > Path Networks > Paths… > To
3	I Bool	PMD_Field_PathSegBiDi	Build > Path Networks > Paths… > BI
4	S	PMD_Field_PathSegSpeedFactor	Build > Path Networks > Paths… >
5	S	PMD_Field_PathSegDistance	Build > Path Networks > Paths… >Time/Distance

Interfaces Subtable (52): PMD_Table_PathInterfaces

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_PathXfaceNode	Build > Path Networks > Interfaces… > Node
2	S	PMD_Field_PathXfaceLoc	Build > Path Networks > Interfaces… > Location

Mappings Subtable (53): PMD_Table_PathMappings

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_PathMapFromNode	Build > Path Networks > Mapping… > From
2	I	PMD_Field_PathMapSegNum	Not Visible (references Path Networks Segment record number)

Mapping Destinations Subtable (57): PMD_Table_PathMapDest

Field	Type	Constant	Description or Location in ProModel
3	S	PMD_Field_PathMapDestNode	pmMapDestSNode

Nodes Subtable (54): PMD_Table_PathNodes

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_PathNodeName	Build > Path Networks > Nodes> Node
2	S	PMD_Field_PathNodeCapacity	Build > Path Networks > Nodes> Limit
3	I	PMD_Field_PathNodeXPos	Not Visible (number of pixels from left)
4	I	PMD_Field_PathNodeYPos	Not Visible (number of pixels from top)

Resources Table (4): PMD_Table_Resources

Field	Type	Constant	Description or Location in ProModel
2	S	PMD_Field_ResName	Build > Resources > Name
3	S	PMD_Field_ResUnits	Build > Resources > Units
6	I	PMD_Field_ResStats	Build > Resources > Stats…
		Stats Rule 1 = None pmdStatsTypeNone 2 = Summary pmdStatsTypeBasicSum 3 = By Unit pmdStatsTypeTSUni	
7	S	PMD_Field_ResNetwork	Build > Resources > Specs… > Path Network
8	I	PMD_Field_ResSearch	Build > Resources > Specs… > Resource Search
		Resource Search Rule 1 = Closest Resource pmdResSrchClosest 2 = Longest Idle pmdResSrchLongestIdle 3 = Least Utilized pmdResSrchLeastUsed	
9	I	PMD_Field_ResEntSearch	Build > Resources > Specs… > Entity Search
		Entity Search Rule 1 = Longest Waiting pmdEntSrchOldest 2 = Closest Entity pmdEntSrchClosest 3 = Minimum Attribute Value pmdEntSrchMinAttrib 4 = Maximum Attribute Value pmdEntSrchMaxAttrib	
10	S	PMD_Field_ResEntSrchMinAttrib	Build > Resources > Specs… > Entity Search > Min Attribute
11	S	PMD_Field_ResEntSrchMaxAttrib	Build > Resources > Specs… > Entity Search > Max Attribute
12	S	PMD_Field_ResSpeedEmpty	Build > Resources > Specs… > Motion > Speed (Empty)
13	S	PMD_Field_ResSpeedFull	Build > Resources > Specs… > Motion > Speed (Full)
14	S	PMD_Field_ResAccel	Build > Resources > Specs… > Motion > Accelerate
15	S	PMD_Field_ResDecel	Build > Resources > Specs… > Motion > Decelerate
16	S	PMD_Field_ResPickupTime	Build > Resources > Specs… > Motion > Pick-up Time
17	S	PMD_Field_ResDepositTime	Build > Resources > Specs… > Motion > Deposit Time
18	S	PMD_Field_ResHomeNode	Build > Resources > Specs… > Nodes > Home
19	I Bool	PMD_Field_ResHomeIfIdle	Build > Resources > Specs… > Nodes > Return Home If Idle
20	S	PMD_Field_ResOffShiftNode	Build > Resources > Specs… > Nodes > Off-Shift
21	S	PMD_Field_ResBreakNode	Build > Resources > Specs… > Nodes > Break
26	S	PMD_Field_ResNotes	Build > Resources > Notes
27	S	PMD_Field_ResCost	Build > Cost > Resources > Regular Rate

29	S	PMD_Field_ResCostPerUse	Build > Cost > Resources > Cost Per Use
30	I	PMD_Field_ResCostTimeUnit	Build > Cost > Resources > Per
31	R		Build > Resources > Resource Graphics > Graphic Size

Clock Downtimes Subtable (31): PMD_Table_ResClockDTs

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_DTFrequency	Frequency
2	S	PMD_Field_DTFirstTime	First Time
3	S	PMD_Field_DTPriority	Priority
4	I	PMD_Field_DTScheduled	Scheduled
5	S	PMD_Field_DTLogic	Logic
6	I	PMD_Field_DTDisable	Disable
9	S	PMD_Field_DTList	List
10	S	PMD_Field_DTNode	Node

Usage Downtimes Subtable (32): PMD_Table_ResUsageDTs

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_DTFrequency	Frequency
2	S	PMD_Field_DTFirstTime	First Time
3	S	PMD_Field_DTPriority	Priority
5	S	PMD_Field_DTLogic	Logic
6	I	PMD_Field_DTDisable	Disable
9	S	PMD_Field_DTList	List
10	S	PMD_Field_DTNode	Node

Work Search Subtable (33): PMD_Table_ResWorkSearch

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_ResWorkSrchNode	Build > Resources > Search… > Work > Node
2	I	PMD_Field_ResWorkSrchType	Build > Resources > Search… > Work > Type
		Resource Work Search Type Rule 1 = Exclusive pmdWSrchTypeExclusive 2 = Non-Exclusive pmdWSrchTypeNonExclusv	

Search Locations Subtable (58): PMD_Table_WorkSearchLocs

Field	Type	Constant	Description or Location in ProModel
3	S	PMD_Field_WorkSrchLocName	Build > Resources > Search… > Work > Location List

Resource Graphics (46): PMD_Table_ResourceGraphics

Field	Type	Constant	Description or Location in ProModel
1	1	PMD_GfxType_Library	Library

Park Search Subtable (34): PMD_Table_ResParkSearch

Field	Type	Constant	Description or Location in ProModel
1	S	PMD_Field_ResParkSrchNode	Build > Resources > Search… > Park > Node

Park Nodes Subtable (59):PMD_Table_ParkNodes

Field	Type	Constant	Description or Location in ProModel
2	S	PMD_Field_ResParkNodeName	Build > Resources > Search… > Park > Parking Node List

Node Logic Subtable (35): PMD_Table_ResNodeLogic

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldResNLogicNode	Build > Resources > Logic… > Node
2	S	pmdFldResNLogicEntry	Build > Resources > Logic… > Entry Logic
3	S	pmdFldResNLogicExit	Build > Resources > Logic… > Exit Logic

Resource Points Subtable (36): PMD_Table_ResPoints

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldPointsNode	Build > Resources > Pts… > Node
2	I	pmdFldPointsXPos	Build > Resources > Pts… > Points (before comma)
3	I	pmdFldPointsYPos	Build > Resources > Pts… > Points (after comma)

Arrivals Table (6): PMD_Table_Arrivals

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldArrivalEntName	Build > Arrivals > Entity…
2	S	pmdFldArrivalLocName	Build > Arrivals > Location…
3	S	pmdFldArrivalQtyEach	Build > Arrivals > Qty Each… > Quantity
4	S	pmdFldArrivalCycle	Build > Arrivals > Qty Each… > Arrival Cycle
5	S	pmdFldArrivalFirstTime	Build > Arrivals > First Time
6	S	pmdFldArrivalOccur	Build > Arrivals > Occurrences
7	S	pmdFldArrivalFrequency	Build > Arrivals > Frequency
8	S	pmdFldArrivalLogic	Build > Arrivals > Logic…
9	I Bool	pmdFldArrivalDisable	Build > Arrivals > Disable
10	I	pmdFldArrivalTimeBasis	Build > Arrivals > First Time > Define Arrival By

		Arrival Time Basis Rule: 0 = Time Only 1 = Calendar 2 = Weekly	
11	I	pmdFldArrivalMinute	Build > Arrivals > First Time > Edit Arrival Time > Min
12	I	pmdFldArrivalHour	Build > Arrivals > First Time > Edit Arrival Time > Hr
13	I	pmdFldArrivalWeekDay	Build > Arrivals > First Time > Edit Arrival Time > Day
14	I	pmdFldArrivalMonthDay	Build > Arrivals > First Time > Edit Arrival Time > Day
15	I	pmdFldArrivalWeek	Build > Arrivals > First Time > Edit Arrival Time > Week
16	I	pmdFldArrivalMonth	Build > Arrivals > First Time > Edit Arrival Time > Month
17	I	pmdFldArrivalYear	Build > Arrivals > First Time > Edit Arrival Time > Year
18	S	pmdFldArrivalVariation	Build > Arrivals > First Time > Scheduling Options > Variation
19	S	pmdFldArrivalOffset	Build > Arrivals > First Time > Scheduling Options > Offset
20	I	pmdFldArrivalRepeatType	Build > Arrivals > First Time > Scheduling Options > Repeat (Daily/Weekly)
		Arrival Repeat Type Rule 1 = No Selection pmdRepeatNone 2 = Repeat Daily pmdRepeatDaily 3 = Repeat Weekly pmdRepeatWeekly	

Shift Assignment Table (7): PMD_Table_Shifts

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldShiftPriority	Build > Shifts > Assign > Priorities… > Priority for ending shift
2	S	pmdFldShiftOffShiftPri	Build > Shifts > Assign > Priorities… > Off-shift priority
3	S	pmdFldShiftStartBreakPri	Build > Shifts > Assign > Priorities… > Priority for starting break
4	S	pmdFldShiftBreakPri	Build > Shifts > Assign > Priorities… > Break priority
5	S	pmdFldShiftPreOffShiftLogic	Build > Shifts > Assign > Logic> Pre-Off Shift
6	S	pmdFldShiftOffShiftLogic	Build > Shifts > Assign > Logic> Off Shift
7	S	pmdFldShiftPreBreakLogic	Build > Shifts > Assign > Logic> Pre-Break
8	S	pmdFldShiftBreakLogic	Build > Shifts > Assign > Logic> Break
9	I Bool	pmdFldShiftDisable	Build > Shifts > Assign > Disable

Location Index Subtable (38): PMD_Table_Shift_Locations

Field	Type	Constant	Description or Location in ProModel
10	I	pmdFldShiftLocNum	Build > Shifts > Assign > Locations

Resource Index Subtable (39): PMD_Table_Shift_Resources

Field	Type	Constant	Description or Location in ProModel
11	I	pmdFldShiftResNum	Build > Shifts > Assign > Resources
12	S	pmdFldShiftResUnits	Build > Shifts > Assign > Resources > Units

Shift File Index Subtable (40): PMD_Table_ShiftFiles

Field	Type	Constant	Description or Location in ProModel
13	I	pmdFldShiftFileNum	Build > Shifts > Assign > Shift Files > Selected Files
14	S	pmdFldShiftStartTime	Build > Shifts > Assign > Shift Files > Start

Attributes Table (8): PMD_Table_Attributes

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldAttribID	Build > Attributes > ID
2	I	pmdFldAttribType	Build > Attributes > Type…
		Type Rule 2 = Integer pmdDataTypeInteger 4 = Real pmdDataTypeReal	
3	I	pmdFldAttribClass	Build > Attributes > Classification…
		Classification Rule 1 = Entity pmdAttribClassEnt 2 = Location pmdAttribClassLoc	
4	S	pmdFldAttribNotes	Build > Attributes > Notes…

Variables Table (9): PMD_Table_Variables

Field	Type	Constant	Description or Location in ProModel
2	S	pmdFldVarID	Build > Variables > ID
3	I	pmdFldVarType	Build > Variables > Type…
		Type Rule 2 = Integer pmdDataTypeInteger 4 = Real pmdDataTypeReal	
4	S	pmdFldVarInitValue	Build > Variables > Initial Value
5	I	pmdFldVarStatsType	Build > Variables > Stats… > (None/Basic/Time Series)
		Stats Type 1 = None pmdStatsTypeNone 2 = Basic or Summary pmdStatsTypeBasicSum 3 = Time Series or By Unit pmdStatsTypeTSUni	

6	I	pmdFldVarStatsBasis	Build > Variables > Stats… > (Time-weighted/Observation-based)
		Statistics Basis Rule 1 = Time Weighted pmdStatsBasisTime 2 = Observation Based pmdStatsBasisObserv	
7	S	pmdFldVarNotes	Build > Variables > Notes…

Variable Graphics (60): PMD_Table_VariableGraphics

Field	Type	Constant	Description or Location in ProModel
1	**4**	pmdGfxTypeCounter	Counter

Arrays Table (10): PMD_Table_Arrays

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldArrayID	Build > Arrays > ID
2	S	pmdFldArrayDimens	Build > Arrays > Dimensions
3	I	pmdFldArrayType	Build > Arrays > Type…
		Array Type 2 = Integer 4 = Real 6 = String 7 = Expression	
4	S	pmdFldArrayNotes	Build > Arrays > Notes…
5	S	pmdFldArrayImportPath	Build > Arrays > Import File… > Import File
6	S	pmdFldArraySheetName	Build > Arrays > (Import) File… > Sheet Name
8	S	pmdFldArrayStartCell	Build > Arrays > (Import) File… > (Import) Start Cell
9	S	pmdFldArrayEndCell	Build > Arrays > (Import) File… > (Import) End Cell
10	S	pmdFldArrayExportPath	Build > Arrays > Export File… > Export File
11	S	pmdFldArrayExSheetName	Build > Arrays > (Export) File… > Sheet Name
12	S	pmdFldArrayExStartCell	Build > Arrays > (Export) File… > (Export) Start Cell
13	S	pmdFldArrayExEndCell	Build > Arrays > (Export) File… > (Export) End Cell
14	S	pmdFldArrayImDBConnect	Build > Arrays > (Import) File… > (Import) DB Connection String
15	S	pmdFldArrayImDBQueryProc	Build > Arrays > (Import) File… > (Import) DB Query or Stored Procedure
17	I Bool	pmdFldArrayImportType	Build > Arrays > (Import) File… > (Import) Type (0=Excel, 1=Database)
20	I Bool	pmdFldArrayPersistData	Build > Arrays > Persist (0=Clear, 1=Keep)
21	I Bool	pmdFldArrayExportLastRep	Build > Arrays > (Export) File… > Export after final replication only (0=no, 1=yes)
22	I Bool	pmdFldArrayDisImport	Build > Arrays > Disable (Import) (0=no, 1=yes)
23	I Bool	pmdFldArrayDisExport	Build > Arrays > Disable (Export) (0=no, 1=yes)

Macros Table (11): PMD_Table_Macros

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldMacroID	Build > Macros > ID
2	S	pmdFldMacroText	Build > Macros > Text…
4	I Bool	pmdFldMacroGroup	Build > Macros > Options… > Resource Group
5	S	pmdFldMacroRTIName	Build > Macros > Options… > RTI > Define > Parameter Name
6	S	pmdFldMacroRTIPrompt	Build > Macros > Options… > RTI > Define > Prompt
7	S	pmdFldMacroRTIMinValue	Build > Macros > Options… > RTI > Define > From
8	S	pmdFldMacroRTIMaxValue	Build > Macros > Options… > RTI > Define > To
9	I	pmdFldMacroRTIType	Build > Macros > Options… > RTI > Define > (type selections)
		Type Rule 2 = Range 3 = Unrestricted	

Subroutines Table (12):PMD_Table_Subroutines

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldSubRtnID	Build > Subroutines > ID
2	S	pmdFldSubRtnLogic	Build > Subroutines > Logic…
4	I	pmdFldSubRtnType	Build > Subroutines > Type…
		Type Rule 0 = None pmdDataTypeNone 2 = Integer pmdDataTypeInteger 4 = Real pmdDataTypeReal 8 = Interactive pmdDataTypeInteract	

Parameters Subtable (41): PMD_Table_SubRoutineParams

Field	Type	Constant	Description or Location in ProModel
5	S	pmdFldSubRtnParamName	Build > Subroutines > Parameters… > ID
6	I	pmdFldSubRtnParamType	Build > Subroutines > Parameters… > Type…
		Parameter Type Rule 2 = Integer pmdDataTypeInteger 4 = Real pmdDataTypeReal	

Arrival Cycles Table (13): PMD_Table_ArrivalCycles

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldArrCycID	Build > More Elements > Arrival Cycles > ID
2	I	pmdFldArrCycType	Build > More Elements > Arrival Cycles > Qty/%

		Quantity Rule 0 = Percent pmdArvCycPercent 1 = Quantity pmdArvCycQuant	
3	I Bool	pmdFldArrCycCumulative	Build > More Elements > Arrival Cycles > Cumulative…

Arrival Cycle Values Subtable (42): PMD_Table_ArrivalCycleSubtable

Field	Type	Constant	Description or Location in ProModel
5	S	pmdFldArrCycTime	Build > More Elements > Arrival Cycles > Table… > Time
6	S	pmdFldArrCycQtyPercent	Build > More Elements > Arrival Cycles > Table… > Qty/%

Table Functions Table (14): PMD_Table_TableFunctions

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldTableFunID	Build > More Elements > Table Functions > ID

Function Values Subtable (43): PMD_Table_FunctionSubtable

Field	Type	Constant	Description or Location in ProModel
3	S	pmdFldTableFunIndepVal	Build > More Elements > Table Functions > Table… > Independent Value
4	S	pmdFldTableFunDepVal	Build > More Elements > Table Functions > Table… > Dependent Value

User Distributions Table (15): PMD_Table_UserDistribs

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldUserDistID	Build > More Elements > User Distributions > ID
2	I	pmdFldUserDistType	Build > More Elements > User Distributions > Type…
		Type Rule 1 = Discrete pmdUDistDiscrete 2 = Continuous pmdUDistContinuous	
3	I	pmdFldUserDistCumulative	Build > More Elements > User Distributions > Cumulative…

User Distributions Values Subtable (44): PMD_Table_UserDistribSubtable

Field	Type	Constant	Description or Location in ProModel
4	S	pmdFldUserDistPercentage	Build > More Elements > User Distributions > Table… > Percentage
5	S	pmdFldUserDistValue	Build > More Elements > User Distributions > Table… > Value

External Files Table (16): PMD_Table_ExternalFiles

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldXfilesID	Build > More Elements > External Files > ID
2	I	pmdFldXfilesType	Build > More Elements > External Files > Type…
		Type Rule 1 = General Read pmdFileTypeGenRead 2 = General Write pmdFileTypeGenWrite 3 = Entity Location pmdFileTypeEntLoc 4 = Arrival pmdFileTypeArrival 5 = Shift pmdFileTypeShift 6 = DLL pmdFileTypeDLL 7 = Excel pmdFileTypeExcel	
3	S	pmdFldXfilesPath	Build > More Elements > External Files > File Name…
4	S	pmdFldXfilesPrompt	Build > More Elements > External Files > Prompt
5	S	pmdFldXfilesNotes	Build > More Elements > External Files > Notes…

Streams Table (17): PMD_Table_Streams

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldStreamNum	Build > More Elements > Streams > Stream #
2	I	pmdFldStreamSeedNum	Build > More Elements > Streams > Seed #
3	I	pmdFldStreamReset	Build > More Elements > Streams > Reset…
		Reset Rule 0 = No 1 = Yes	

General Information Table (18): PMD_Table_GenInfo

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldGenInfoTitle	Build > General Information > Title
2	I	pmdFldGenInfoDefTimeUnits	Build > General Information > Time Units
		Default Time Units Rule 1 = Seconds pmdTimeUnitSec 2 = Minutes pmdTimeUnitMin 3 = Hours pmdTimeUnitHr 4 = Days pmdTimeUnitDay	

3	I	pmdFldGenInfoDefDistUnits	Build > General Information > Distance Units
		Default Distance Units Rule 1 = Feet pmdDistUnitFeet 2 = Meters pmdDistUnitMeters	
4	S	pmdFldGenInfoGLibFile	Build > General Information > Graphic Library File…
5	S	pmdFldGenInfoInitLogic	Build > General Information > Initialization Logic…
6	S	pmdFldGenInfoTermLogic	Build > General Information > Termination Logic…
7	S	pmdFldGenInfoNotes	Build > General Information > Model Notes…
8	I		Has the model been modified?
		Modified Rule 0 = No 1 = Yes	
9	S	pmdFldGenInfoModFile	Path of the .MOD file.
10	R	pmdFldGenInfoZoomPct	Layout zoom percentage.

Process Table (19): PMD_Table_Processing

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldProcEntName	Build > Processing > Entity…
2	I Bool	pmdFldProcPreempt	Build > Processing > Entity… > Preemption Process
3	S	pmdFldProcLocName	Build > Processing > Location…
4	S	pmdFldProcOpLogic	Build > Processing > Operation…

Routing Subtable (20): PMD_Table_Routing

Field	Type	Constant	Description or Location in ProModel
6	S	pmdFldRtgEntName	Build > Processing > Output…
7	S	pmdFldRtgLocName	Build > Processing > Destination…
8	S	pmdFldRtgPriority	Build > Processing > Destination… > Priority
9	I Bool	pmdFldRtgNewBlock	Build > Processing > Rule… > New Block
10	I Bool	pmdFldRtgNewEntity	Build > Processing > Rule… > New Entity
11	S	pmdFldRtgQuantity	Build > Processing > Rule… > Quantity
12	I	pmdFldRtgRoutingRule	Build > Processing > Rule… > (choice list)

		Routing Rules 1 = First Available pmdRtgFirstAvail 2 = Most Available pmdRtgMostAvailCap 3 = By Turn pmdRtgByTurn 4 = If Join Request pmdRtgJoinRequest 5 = If Load Request pmdRtgLoadRequest 6 = If Send pmdRtgIfSend 7 = Until Full pmdRtgUntilFull 8 = As Alternate pmdRtgAsAlt 9 = Probability pmdRtgProb 10 = User Condition pmdRtgCond 11 = Random pmdRtgRandom 12 = Longest Unoccupied pmdRtgLongestEmpty 13 = If Empty pmdRtgIfEmpty 14 = Continue pmdRtgContinue 15 = As Backup pmdRtgAsBackup 16 = Dependent pmdRtgDependent	
13	R	pmdFldRtgProbability	Build > Processing > Rule… > Probability
14	S	pmdFldRtgCondition	Build > Processing > Rule… > User Condition
15	S	pmdFldRtgMoveLogic	Build > Processing > Move Logic…

Routing Points Table (62):

Field	Type	Constant	Description or Location in ProModel
1	I		(Distance, in pixels, from Left margin)
2	I		(Distance, in pixels, from Top margin)

Model Parameters Table (21): PMD_Table_Model_Params

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldModParamValue	Simulation > Model Parameters > Change

Scenarios Table (22): PMD_Table_Scenarios

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldScenName	Simulation > Scenarios
2	I Bool	pmdFldScenEnable	Simulation > Scenarios > Enable (toggles on/off for selected record)

Scenario Parameter Subtable (56): PMD_Table_ScenarioParams

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldScenParamValue	Simulation > Scenarios > (Add/Edit) > Change

Simulation Options (23): PMD_Table_SimOptions

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldSimOptOutPath	Simulation > Options > Output Path
2	I	pmdFldSimOptTimeBasis	Simulation > Options > Define run length by
		Time Basis Rule 0 = Time Only pmdTimeBasisTimeOnly 1 = Calendar Date pmdTimeBasisCalDate 2 = Weekly Time pmdTimeBasisWeeklyTime **Note:** Values 3 & 4 are used only if Time Basis = 0.	
3	S	pmdFldSimOptRunHours	Simulation > Options > Run Hours (Time Only)
4	S	pmdFldSimOptWarmPeriod	Simulation > Options > Warmup Hours (Time Only)
		Sunday = 0 Monday = 1 Tuesday = 2 Wednesday = 3 Thursday = 4 Friday = 5 Saturday = 6 **Note:** The returned year is the current year minus 1900 The month of the year is zero based (January is month 0, not month 1).	
5	I	pmdFldSimOptWarmMonCal	Simulation > Options > Warmup Start > Month (Calendar)
6	I	pmdFldSimOptWarmDayCal	Simulation > Options > Warmup Start > Day (Calendar)
7	I	pmdFldSimOptWarmYrCal	Simulation > Options > Warmup Start > Year (Calendar)
8	I	pmdFldSimOptWarmHr	Simulation > Options > Warmup Start > Hour (Weekly, Calendar)
9	I	pmdFldSimOptWarmMin	Simulation > Options > Warmup Start > Min (Weekly, Calendar)
10	I	pmdFldSimOptBegMonCal	Simulation > Options > Sim. Begin > Month (Calendar)
11	I	pmdFldSimOptBegDayCal	Simulation > Options > Sim. Begin > Day (Calendar)
12	I	pmdFldSimOptBegYrCal	Simulation > Options > Sim. Begin > Year (Calendar)
13	I	pmdFldSimOptBegHr	Simulation > Options > Sim. Begin > Hour (Weekly, Calendar)
14	I	pmdFldSimOptBegMin	Simulation > Options > Sim. Begin > Min (Weekly, Calendar)
15	I	pmdFldSimOptEndMonCal	Simulation > Options > Sim. End > Month (Calendar)
16	I	pmdFldSimOptEndDayCal	Simulation > Options > Sim. End > Day (Calendar)

17	I	pmdFldSimOptEndYrCal	Simulation > Options > Sim. End > Year (Calendar)
18	I	pmdFldSimOptEndHr	Simulation > Options > Sim. End > Hour (Weekly, Calendar)
19	I	pmdFldSimOptEndMin	Simulation > Options > Sim. End > Min (Weekly, Calendar)
20	I	pmdFldSimOptRptMethod	Simulation > Options > Output Reporting
		Output Report Method Rule 0 = Standard pmdSimRptStandard 1 = Batch Mean pmdSimRptBatchMean 2 = Periodic pmdSimRptPeriodic	
21	S	pmdFldSimOptRptIntLength	Simulation > Options > Output Reporting > Interval Length
22	S	pmdFldSimOptNumReps	Simulation > Options > Output Reporting > Number of Replications
23	I	pmdFldSimOptClockPrecision	Simulation > Options > Clock Precision > (drop-down list)
		Clock Precision Rule 0 = No Clock Precision Chosen pmdSimClockPrecNone 1 = 1 pmdSimClockPrec1 2 = .1 pmdSimClockPrec10 3 = .01 pmdSimClockPrec100 4 = .001 pmdSimClockPrec1000 5 = .0001 pmdSimClockPrec10000 6 = .00001 pmdSimClockPrec100000	
24	I	pmdFldSimOptClockPrecUnits	Simulation > Options > Clock Precision > (selection list)
		Clock Precision Units Rule 1 = Seconds pmdTimeUnitSec 2 = Minutes pmdTimeUnitMin 3 = Hours pmdTimeUnitHr 4 = Days pmdTimeUnitDay	
25	I Bool	pmdFldSimOptDisTimeSeries	Simulation > Options > Disable Time Series
26	I Bool	pmdFldSimOptDisAnimation	Simulation > Options > Disable Animation
27	I Bool	pmdFldSimOptDisCost	Simulation > Options > Disable Cost
28	I Bool	pmdFldSimOptPause	Simulation > Options > Pause at Start
29	I Bool	pmdFldSimOptDisplayNotes	Simulation > Options > Display Notes
30	I Bool	pmdFldSimOptEnableWarmup	Simulation > Options > Warmup Period
31	I	pmdFldSimOptWarmDayWk	Simulation > Options > Warmup Start > Day (Weekly)
32	I	pmdFldSimOptWarmWeekWk	Simulation > Options > Warmup Start > Week (Weekly)
33	I	pmdFldSimOptBegDayWk	Simulation > Options > Sim. Begin > Day (Weekly)
34	I	pmdFldSimOptBegWeekWk	Simulation > Options > Sim. Begin > Week (Weekly)

35	I	pmdFldSimOptEndDayWk	Simulation > Options > Sim. End > Day (Weekly)
36	I	pmdFldSimOptEndWeekWk	Simulation > Options > Sim. End > Week (Weekly)
37	I Bool	pmdFldSimOptDaySaveAdjust	Simulation > Options > Adjust for Daylight Savings
38	I Bool	pmdFldSimOptGenAniScript	Simulation > Options > Generate Animation Script
39	I Bool	pmdFldSimOptComRand	Simulation > Options > Common Random Numbers
40	I Bool	pmdFldSimOptSkipResDT	Simulation > Options > Skip Resource DTs if Off-shift
41	I Bool	pmdFldSimOptDisArrayExp	Simulation > Options > Disable Array Export
42	I	pmFldSimOptInitAnimSpeed	
43	I Bool	pmdFldSimOptTrace	Simulation>Options>Trace
44	I Bool	pmdFldSimOptRecompileMap	Simulation>Options>Recompile Mappings
45	S	PMD_Field_SimOptRunName	Simulation>Options>RunName
46	I Bool	PMD_Field_SimOptMinitab	Simulation>Options>MiniTab
47	I Bool	PMD_Field_SimOptOut-putViewer	Simulation>Options>OutputViewer
48	I	PMD_Field_SimOptOVver-sionNum	Simulation>Options>OutputViewerVersionNumber
49	I Bool	PMD_Field_SimOptSim-ulateBaseline	Simulation>Options>SimulateBaseline

Graphic Type Tables

Library Graphics (Graphic Type Table 1)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	Graphic Type
2	I	pmdFldGfxLibWidth	Width
3	I	pmdFldGfxLibHeight	Height
4	R	pmdFldGfxLibWidR	Real Width (feet or meters)
5	R	pmdFldGfxLibHtR	Real Height (feet or meters)
6	I	pmdFldGfxLibRotat	Rotation
		Enum Rule 0 = none 900 = 90 degrees 1800 = 180 degrees 2700 = 270 degrees	
7	I	pmdFldGfxLibID	ID
8	I	pmdFldGfxLibSpotX	Hotspot X value
9	I	pmdFldGfxLibSpotY	Hotspot Y value
10	I	pmdFldGfxLibXpos	Postion X value
11	I	pmdFldGfxLibYpos	Postion Y value

12	I	pmdFldGfxLibColor	Color
20	S	pmdFldGfxLibConvWidth	Width on conveyor (Entities only)
21	S	pmdFldGfxLibConvLength	Length on conveyor (Entities only)

Queue/Conveyor Graphics (Graphics Type Table 2)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	Graphic Type
2	I Bool	pmdFldGfxQIsQ	Conveyor?
		Conveyor Rule 0 = No 1 = Yes	
3	S	pmdFldGfxQSpeed	Speed (Conveyor Only)
4	S	pmdFldGfxQLength	Length (Conveyor Only)
5	I Bool	pmdFldGfxQAccum	Accumulating?
		Accumulating Rule 0 = No 1= Yes	
6	I	pmdFldGfxQEntDir	Entity Orientation
		Entity Orientation Rule 0 = Lengthwise 1 = Widthwise	
7	I	pmdFldGfxQBrdrClr	Border Color
8	I	pmdFldGfxQFillClr	Fill Color
9	I	pmdFldGfxQStyle	Graphic Style
		Graphic Style Rule 1 = Solid 2 = Line 3 = Roller	
10	I	pmdFldGfxQWidth	Width
11	I Bool	pmdFldGfxQIsVis	Invisible?
		Invisible Rule 0 = No 1 = Yes	

Gauge/Tank Graphics (Graphics Type Table 3)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	GraphicType
2	I	pmdFldGfxTnkLeft	Left
3	I	pmdFldGfxTnkTop	Top
4	I	pmdFldGfxTnkRight	Right
5	I	pmdFldGfxTnkBotm	Bottom
6	I	pmdFldGfxTnkBrdrClr	Border Color
7	I	pmdFldGfxTnkEmptyClr	Empty Color
8	I	pmdFldGfxTnkFillClr	Fill Color
9	R	pmdFldGfxTnkMinVal	Minimum Value
10	R	pmdFldGfxTnkMaxVal	Maximum Value
11	I	pmdFldGfxTnkDir	Direction
		Direction Rule 1 = Up 2 = Down 3 = Left 4 = Right	
12	I	pmdFldGfxTnkScale	Show Scale?
		Scale Rule 0 = No 1 = Yes	
13	I	pmdFldGfxTnkBrdr	Show Border?
		Border Rule 0 = No 1 = Yes	
14	I	pmdFldGfxTnkIsTank	Tank?
		Tank Rule 0 = No 1 = Yes	

Counter Graphics (Graphics Type Table 4)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	GraphicType
10	I	pmdFldGfxCtrLeft	Left
11	I	pmdFldGfxCtrTop	Top
12	I	pmdFldGfxCtrRight	Right

13	I	pmdFldGfxCtrBotm	Bottom
14	I	pmdFldGfxCtrFrType	Frame Type
		Frame Type Rule 1 = Invisible 2 = Plain 3 = Raised 4 = Recessed 5 = Shadow 6 = None	
15	I	pmdFldGfxCtrFrShape	Frame Shape
		Frame Shape Rule 1 = Rectangle 2 = Round Rectangle 3 = Ellipse 4 = Diamond	
16	I	pmdFldGfxCtrFrClr	Frame Color
17	I	pmdFldGfxCtrBrdrClr	Frame Border Color
20	I	pmdFldGfxCtrFontSize	Font Size
21	S	pmdFldGfxCtrFont	Font Name
22	I	pmdFldGfxCtrFontClr	Font Color

Text Graphics (Graphic Type Table 5)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	Graphic Type
2	S	pmdFldGfxTxtLabel	Label Text
10	I	pmdFldGfxTxtLeft	Left
11	I	pmdFldGfxTxtTop	Top
12	I	pmdFldGfxTxtRight	Right
13	I	pmdFldGfxTxtBotm	Bottom
14	I	pmdFldGfxTxtFrType	Frame Type

		Frame Type Rule 1 = Invisible 2 = Plain 3 = Raised 4 = Recessed 5 = Shadow 6 = None	
15	I	pmdFldGfxTxtFrShape	Frame Shape
		Frame Shape Rule 1 = Rectangle 2 = Round Rectangle 3 = Ellipse 4 = Diamond	
16	I	pmdFldGfxTxtFrClr	Frame Color
17	I	pmdFldGfxTxtBrdrClr	Frame Border Color
20	I	pmdFldGfxTxtFontSize	Font Size
21	S	pmdFldGfxTxtFont	Font Name
22	I	pmdFldGfxTxtFontClr	Font Color
23	I		Alignment
		Allignment Rule 1 = Left 2 = Center 3 = Right	
24	I		Rotation
		Rotation Rule 0 = None 1 = 90 degrees 2 = 180 degrees 3 = 270 degrees	

Status Light Graphics (Graphic Type Table 6)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	Graphic Type
2	I	pmdFldGfxLightX	X Position
3	I	pmdFldGfxLightY	Y Position

Entity Spot Graphics (Graphic Type Table 7)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	Graphic Type
2	I	pmdFldGfxSpotX	X Position
3	I	pmdFldGfxSpotY	Y Position

Region Graphics (Graphic Type Table 8)

Field	Type	Constant	Description or Location in ProModel
1	I	pmdFldGfxType	GraphicType
2	I	pmdFldGfxRgnLeft	Left
3	I	pmdFldGfxRgnTop	Top
4	I	pmdFldGfxRgnRight	Right
5	I	pmdFldGfxRgnBotm	Bottom

Background Graphics (49): pmdTblBackGraphic

Field	Type	Constant	Description or Location in ProModel
1	I		1=Library, 6=Text
4	I		Font color (RGB)
6	I		Position Locked? (0 = no, 1 = yes)
8	I		X-coordinate of upper left corner of graphic
9	I		Y-coordinate of upper left corner of graphic
10	I		X-coordinate of lower right corner of graphic
11	I		Y-coordinate of lower right corner of graphic
100	S		Text shown on graphic
101	I		Frame type (1=invisible, 2=plain, 3=raised, 4=recessed, 5=shadow, 6=no border)
102	I		Frame shape (1=rectangle, 2=rounded rectangle, 3=ellipse, 4=di-iamond)
103	I		Frame border color (RGB)
120	I		Font size (use negative number to specify point size)
121	S		Font name
122	I		Alignment (1=left, 2=center, 3=right)
123	I		Rotation (0=0 degrees, 1=90 degrees, 2=180 degrees, 3=270 degrees

Model Defaults Table (50): pmdTblModelDefaults

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldModDfltModPath	Tools> Options> Default Folders > Models
2	S	pmdFldModDfltGfxPath	Tools> Options> Default Folders > Graphics Library
3	S	pmdFldModDfltOutPath	Tools> Options> Default Folders > Output Results
4	I Bool	pmdFldModDfltSaveSet	Save defaults to .INI file?
5	I Bool	pmdFldModDfltShowPath	Views> Show Hidden Networks
6	I Bool	pmdFldModDfltShowRtg	Views> Show Routings
7	I Bool	pmdFldModDfltShowGrid	Views> Show Grid
8	I Bool	pmdFldModDfltLongMenu	Tools> Options> Long Build Menu checkbox

Views Table (55): pmdTblView

Field	Type	Constant	Description or Location in ProModel
1	S	pmdFldViewName	View > Views > Define > View List
2	I	pmdFldViewXOffset	Relative horizontal position of view
3	I	pmdFldViewYOffset	Relative vertical position of view
4	I	pmdFldViewWidth	View width in pixels
5	I	pmdFldViewHeight	View height in pixels

Runtime Table.

Table Name	Visual Basic Constant	Value
Locations	pmrTblLocation	1
Single Capacity Locations	pmrTblLocSingle	2
Multi Capacity Locations	pmrTblLocMulti	3
Resources	pmrTblResource	5
Resource States (by Percentage)	pmrTblResState	6
Node Entries	pmrTblNodeEntry	7
Failed Arrivals	pmrTblFailArrival	8
Entity Activity	pmrTblEntAct	9
Entity States (by Percentage)	pmrTblEntState	10
Variables	pmrTblVariable	12
Logs	pmrTblLog	13
Location Costing	pmrTblLocCost	14
Resource Costing	pmrTblResCost	16
Entity Activity Costing	pmrTblEntCost	17

Locations Runtime Fields (1): pmrTblLocation

Field	Constant	Field
1	pmrFldLocEntries	Total Entries
2	pmrFldLocAvgMins	Avg Min per Entry
3	pmrFldLocAvgCnt	Avg Contents
4	pmrFldLocMaxCnt	Max Contents
5	pmrFldLocCurCnt	Cur Contents
6	pmrFldLocUtilPct	Utilization %

Single Capacity Locations Runtime Fields (2): pmrTblLocSingle

Field	Constant	Field
1	pmrFldLocSOperPct	Operation %
2	pmrFldLocSSetupPct	Setup %
3	pmrFldLocSIdlePct	Idle %
4	pmrFldLocSWaitPct	Waiting %
5	pmrFldLocSBlkdPct	Blocked %
6	pmrFldLocSDownPct	Down %

Multi Capacity Locations Runtime Fields (3): pmrTblLocMulti

Field	Constant	Field
1	pmrFldLocMEmptyPct	Empty %
2	pmrFldLocMPartFullPct	Part Occupy %
3	pmrFldLocMFullPct	Full %
4	pmrFldLocMDownPct	Down %

Resource Runtime Fields (5): pmrTblResource

Field	Constant	Field
1	pmrFldResTimesUsed	# Times Used
2	pmrFldResAvgMins	Avg Min per Usage
3	pmrFldResTrvl2Use	Travel to Use
4	pmrFldResTrvl2Park	Travel to Park
5	pmrFldResBlkdTrvlPct	Blocked in Travel %
6	pmrFldResUtilPct	Utilization %

Resource States (By Percentage) Runtime Fields (6): pmrTblResState

Field	Constant	Field
1	pmrFldResStPctUse	In Use %
2	pmrFldResStPctTrvl2Use	Travel to Use %
3	pmrFldResStPctTrvl2Prk	Travel to Park %
4	pmrFldResStPctIdle	Idle %
5	pmrFldResStPctDown	Down %

Node Entries Runtime Fields (7): pmrTblNodeEntry

Field	Constant	Field
1	pmrFldNodeTotEntry	Total Entries
2	pmrFldNodeBlkdEntry	Blocked Entries

Failed Arrivals Runtime Fields (8): pmrTblFailArrival

Field	Constant	Field
1	pmrFldNodeTotFail	Total Failed

Entity Activity Runtime Fields (9): pmrTblEntAct

Field	Constant	Field
1	pmrFldEntTotExit	Total Exits
2	pmrFldEntCurQty	Cur Qty Sys

3	pmrFldEntAvgMinSys	Avg Min System
4	pmrFldEntAvgMinMove	Avg Min Move Logic
5	pmrFldEntAvgWaitRes	Avg Wait Res
6	pmrFldEntAvgMinOper	Avg Min Operation
7	pmrFldEntAvgMinBlkd	Avg Min Blocked

Entity States (By Percentage) Runtime Fields (10): pmrTblEntState

Field	Constant	Field
1	pmrFldEntStPctMove	In Move Logic %
2	pmrFldEntStPctWait	Wait for Res %
3	pmrFldEntStPctOper	In Operation %
4	pmrFldEntStPctBlkd	Blocked %

Variables Runtime Fields (12): pmrTblVariable

Field	Constant	Field
1	pmrFldVarTotChgs	Total Changes
2	pmrFldVarAvgMinPer	Avg Min Per Change
3	pmrFldVarMinVal	Min Value
4	pmrFldVarMaxVal	Max Value
5	pmrFldVarCurVal	Current Value
6	pmrFldVarAvgVal	Avg Value

Logs Runtime Fields (13): pmrTblLog

Field	Constant	Field
1	pmrFldLogNumObsrv	# Observations
2	pmrFldLogMinVal	Min Value
3	pmrFldLogMaxVal	Max Value
4	pmrFldLogAvgVal	Avg Value

Location Costing Runtime Fields (14): pmrTblLocCost

Field	Constant	Field
1	pmrFldLCostOperDlr	$ Operation Cost
2	pmrFldLCostOperPct	% Operation Cost
3	pmrFldLCostResDlr	$ Resource Cost
4	pmrFldLCostResPct	% Resource Cost
5	pmrFldLCostTotDlr	$ Total Cost
6	pmrFldLCostTotPct	% Total Cost

Resource Costing Runtime Fields (16): pmrTblResCost

Field	Constant	Field
1	pmrFldRCostNonUseDlr	$ Non-use Cost
2	pmrFldRCostNonUsePct	% Non-Use Cost
3	pmrFldRCostUseDlr	$ Usage Cost
4	pmrFldRCostUsePct	% Usage Cost
5	pmrFldRCostTotDlr	$ Total Cost
6	pmrFldRCostTotPct	% Total Cost

Entity Activity Costing Runtime Fields (17): pmrTblEntCost

Field	Constant	Field
1	pmrFldECostTotDlr	$ Total Cost
2	pmrFldECostTotPct	% Total Cost

Result Codes (Errors)

Field	Constant	Description of Error
0	pmdErrNoError	No Error.
1	pmdErrInvalidTableNum	There is no table by that number in ProModel.
2	pmdErrMethodNotApplicable	Function does not apply to the selected record (e.g. - SelectMainRecordByName () on the Processing Table)
3	pmdErrInvalidRecordNum	Record by that name does not exist, or record has not been selected (when using Set… or Get…)
4	pmdErrInvalidFieldNum	There is no field (column) by that number in the selected table.
5	pmdErrDataTypeMismatch	Set… or Get… function does not match data type (e.g. - SetStringFieldValue() used on an integer field
6	pmdErrFieldNotApplicable	Field does not exist for the selected record (e.g. - trying to access the RTI fields for non-RTI macros)
7	pmdErrNotImplemented	The selected element has not been ActiveX enabled yet.
8	pmdErrGfxNotImplemented	Graphic element has not been ActiveX enabled yet.
9	pmdErrValueNotApplicable	Value does not apply for selected field (time units = 6. Arrivals > ent name set to non-existing rec.)
11	pmdErrParentNotSelected	The parent table has not been selected
12	pmdErrObsolete	The specified ActiveX element is no longer used
13	pmdErrDataProtected	The model is protected, so data can't be accessed
14	pmdErrKeyNotFound	No hardware key, data can't be accessed
15	pmdErrReadOnlyField	Attempting to write to a read-only field
16	pmdErrJointNotFound	Attempting to delete queue/conveyor joint, when none exist (i.e. - has only start & end points)
99	pmdErrOther	Any other kind of error

Events

Field	Constant	When does the event fire?	Return Values
0	pmeUndefined		
1	pmeQuit	Just after "Save Current Model?" dialog	True = Cancel Quit
2	pmeSaveBeforeQuit	"Save Current Model?" dialog	True = Skip Save dialog
3	pmeSimComplete	Simulation Ends Normally	True = Skip Collect Stats dialog
4	pmeSimUserEnd	User Selects "Simulation > End Simulation"	True = Skip Collect Stats dialog
5	pmeSimPauseOn	User Selects "Simulation > Pause Simulation"	
6	pmeSimPauseOff	User Selects "Simulation > Resume"	
7	pmeSimAbEnd	Abnormal Termination	True = Skip Collect Stats dialog
8	pmeSimStart	Simulation Start (before translation/simulation)	
9	pmeTranStart	Translation Start (at start of translation for each scenario)	
10	pmeTranEnd	Translation End (between translation & simulation)	
11	pmeEditModeOn	Edit Mode Entered (between simulation end and return to build environment)	
12	pmeModLoaded	Model Loaded (after successful load)	
15	pmeAnimOn	Animation On (when animation turned on)	
16	pmeAnimOff	Animation Off (when animation turned off)	
20	pmeMsgInfo		
21	pmeMsgOK	"GLB Missing" message box, other message box	
22	pmeMsgOkCancel		
23	pmeMsgRetryCancel		
24	pmeMsgYesNo	"Couldn't Get Exclusive…" message box	
25	pmeMsgYesNoCancel	"Save Current Model?" dialog	
26	pmeMsgAbortRetryIgnore		
30	pmeMsgPrompt		

Path Colors

Field	Constant	Name
0	pmdColorBlack	Black
128	pmdColorMaroon	Maroon
255	pmdColorRed	Red
32768	pmdColorDk_Green	Dark Green
32896	pmdColorGold	Gold
65280	pmdColorLt_Green	Light Green
65535	pmdColorYellow	Yellow
8388608	pmdColorDk_Blue	Dark Blue
8388736	pmdColorPurple	Purple
8421376	pmdColorTeal	Teal
8421504	pmdColorDk_Gray	Dark Gray
12632256	pmdColorLt_Gray	Light Gray
16711680	pmdColorMed_Blue	Medium Blue
16711935	pmdColorPink	Pink
16776960	pmdColorCyan	Cyan (Light Blue)
16777215	pmdColorWhite	White

Menu Ids

This method is defined as: ***MenuCommand***<*MenuId*>,<*Parameter*>
For most commands, Parameter has no meaning and a zero should be passed. The following is a list of menu IDs that have no equivalent method in the CProModel interface.

Menu ID	Descriptionn	Window	Parameter
1000	Freezes the positions of all layout graphics	Main	
1006	Open Resources Module	Main	
1080	Open Entities Module	Main	
1090	Open Arrivals Module	Main	
1100	Open Variables Module	Main	
1110	Open Attributes Module	Main	
1113	Open Arrays Module	Main	
1116	Open Subroutines Module	Main	
1120	Open Table Functions Module	Main	
1123	Open User Distributions Module	Main	
1126	Open Arrival Cycles Module	Main	
1155	Open Cost Dialog	Main	
1160	Background Graphics (Behind Grid)	Main	
1162	Background Graphics (Front of Grid)	Main	
1164	Background Graphics Right-Click menu	Main	
1165	Open Path Networks Module	Main	
1180	Open Locations Module	Main	
1181	Open Macros Module	Main	
1182	Open External Files Module	Main	
1183	Open Streams Module	Main	
1185	Open General Information Module	Main	
1190	Open Processing Module	Main	
1192	Open Shift Assignments Module	Main	
1500	Find Dialog	Main	
1505	Replace Dialog	Main	
1700	Simulation Options Dialog	Main	
1701	Start Simulation	Main	
1710	Launch SimRunner	Main	
2500	Launch 3DR	Main	1 = Open Hidden
2505	View Trace File	Main	
2600	Launch Graphic Editor	Main	
2605	Launch Shift Editor	Main	
2645	Launch License Manager	Main	

Menu ID	Descriptionn	Window	Parameter
2650	Launch Stat::Fit	Main	
2653	Launch 3D Animator	Main	
2654	Launch Model Collaborator	Main	
2655	Launch Tutorial	Main	
2660	Customized Tools Menu	Main	
2754	View Shortcut Panel	Main	
2755	Refresh Layout	Main	
2850	Open Views Dialog	Main	
2861	Set Layout to 1st View	Main	
2862	Set Layout to 2nd view	Main	
2889	Set Layout to 29th view	Main	
2906	Toggle "Snap to Grid"	Main	
2908	Toggle Grid On/Off	Main	
2910	Open Edit Table Font Selection Dialog	Main	
2920	Open Tools>Options Dialog	Main	
2942	Open Grid Settings Dialog	Main	
2944	Open Background Color Selection Dialog	Main	
2946	Open Routing Path Color Dialog	Main	
2948	Run Scenarios	Main	
2951	Open Scenarios Dialog	Main	
2952	Open Model Parameters Dialog	Main	
2980	Reset Window Positions	Main	
3000	Open About ProModel Window	Main	
3005	Go To Support Page on the Web	Main	
3007	Check for ProModel Version	Main	
3010	Toggle "Show Hidden Networks"	Main	
3015	Toggle "Show Routing Paths"	Main	
3020	Open Color Selection Dialog	Main	
3200	Pop Up Layout Right-Click Menu	Layout	
3601	View>Toolbars>File	Main	
3602	View>Toolbars>Layout	Main	
3603	View>Toolbars>View	Main	
3604	View>Toolbars>Build Basic	Main	
3605	View>Toolbars>Build Advanced	Main	
3606	View>Toolbars>Simulation	Main	
3607	View>Toolbars>Simulation Information	Main	
3608	View>Toolbars>Debug	Main	
3609	View>Toolbars>Tools	Main	
4000	Quit ProModel	Main	

Menu ID	Descriptionn	Window	Parameter
4100	Create New Model	Main	
4200	Open Load Model Dialog	Main	
4237	Open File to Merge Dialog	Main	
4410	Open Save As Dialog	Main	
4415	Open Create Model Package Dialog	Main	
4416	Open Install Model Package Dialog	Main	
4420	Open Print Text to File Dialog	Main	
4500	View Text	Main	
4600	Open Print Layout Dialog	Main	
4601	Open Print Setup Dialog	Main	
4602	Open Print Text Dialog	Main	
4650	Open Help Index	Main	
4655	Open Context Help	Main	
4710	Open File #1 in History List	Main	
4711	Open File #2 in History List	Main	
4712	Open File #3 in History List	Main	
4713	Open File #4 in History List	Main	
4714	Open File #5 in History List	Main	
4801	Simulation Abort	Main	
4804	Trace Off	Simulation	
4805	Close Trace Window	Simulation	
4806	Trace Step	Simulation	
4807	Trace Continuous	Simulation	
4808	Trace to Window	Simulation	
4810	Trace to File	Simulation	
4811	Toggle Animation On/Off	Simulation	
4814	Open User Pause Dialog	Simulation	
4815	Open Location Status Legend	Simulation	
4816	Open Pause	Simulation	
4817	Open Location Info Dialog	Simulation	
4818	Open All Locations Info Dialog	Simulation	
4821	Open Variable Info Dialog	Simulation	
4822	Open Array Info Dialog	Simulation	
4824	Open Debug Options Dialog	Simulation	
4825	Open Interactive Subroutines Dialog	Simulation	
4826	New Dynamic Plots Dialog	Simulation	
4827	Open Dynamic Plots Configuration	Simulation	
4830	Open Custom Trace Options	Simulation	
4850	Resource Work Search	Resources	
4851	Resource Park Search	Resources	

ProActiveX Constants

PM = ProModel Application Object Constants

PMD = ProModel Data Object Constants

PME = ProModel Event Object Constants

RDB = RDB Data Server Object Constants

RTI = RunTime Object Constants

Status Codes

```
  Public Const PM_Status_Unknown = 0
  Public Const PM_Status_Empty = 1   No Model Loaded
  Public Const PM_Status_Loading = 2
  Public Const PM_Status_Loaded = 3
  Public Const PM_Status_LoadError = 4
  Public Const PM_Status_Translating = 5
  Public Const PM_Status_Running = 6
  Public Const PM_Status_RunAborted = 7  Sim or Translation ended before completion
  Public Const PM_Status_RunCompleted = 8
  Public Const PM_Status_RunFrozen = 9
```

Error Codes

```
 Data
  Public Const PMD_Error_NoError = 0
  Public Const PMD_Error_InvalidTableNumber = 1
  Public Const PMD_Error_MethodNotApplicable = 2
  Public Const PMD_Error_InvalidRecNumber = 3
  Public Const PMD_Error_InvalidFieldNumber = 4
  Public Const PMD_Error_DataTypeMismatch = 5
```

```
  Public Const PMD_Error_FieldNotApplicable = 6
  Public Const PMD_Error_NotImplemented = 7
  Public Const PMD_Error_GfxNotImplemented = 8
  Public Const PMD_Error_ValueNotApplicable = 9
  Public Const PMD_Error_ParentNotSelected = 11
  Public Const PMD_Error_Obsolete = 12
  Public Const PMD_Error_DataProtected = 13
  Public Const PMD_Error_KeyNotFound = 14
  Public Const PMD_Error_ReadOnly = 15
  Public Const PMD_Error_JointNotFound = 16
  Public Const PMD_Error_Other = 999

  Run-Time Interface
  Public Const RTI_Error_SimNotRunning = 1
  Public Const RTI_Error_TableNotFound = 2
  Public Const RTI_Error_RecordNotFound = 3
  Public Const RTI_Error_FieldNotFound = 4
  Public Const RTI_Error_SubRecNotFound = 5
  Public Const RTI_Error_NoStatsForRec = 6
  Public Const RTI_Error_TableNotApplicable = 7
  Public Const RTI_Error_StatNotApplicable = 8

-------------------------------------------------------------------------------
Module Codes
-------------------------------------------------------------------------------
  Public Const PM_Module_None = 0  No Modules open
  Public Const PM_Module_Locations = 1
  Public Const PM_Module_Arrivals = 2
  Public Const PM_Module_Entities = 5
  Public Const PM_Module_Resources = 6
  Public Const PM_Module_Variables = 10
  Public Const PM_Module_Attributes = 11
  Public Const PM_Module_Arrays = 12
```

```
  Public Const PM_Module_TableFunctions = 15
  Public Const PM_Module_UserDistribs = 16
  Public Const PM_Module_ArrivalCycles = 17
  Public Const PM_Module_Subroutines = 18
  Public Const PM_Module_Processing = 19
  Public Const PM_Module_PathNetworks = 24
  Public Const PM_Module_Macros = 25
  Public Const PM_Module_Streams = 26
  Public Const PM_Module_ExternalFiles = 27
  Public Const PM_Module_Shifts = 34
  Public Const PM_Module_BackGfxBehind = 36
  Public Const PM_Module_BackGfxInFront = 37

---------------------------------------------------------------------------------------
Event Codes
---------------------------------------------------------------------------------------
  Public Const PME_Event_Undefined = 0
  Public Const PME_Event_Quit = 1       True to cancel Quit
  Public Const PME_Event_SaveBeforeQuit = 2  True to skip Save dialog
  Public Const PME_Event_SimComplete = 3  True to skip Stats dialog
  Public Const PME_Event_SimUserEnd = 4   True to skip Collect Data dialog
  Public Const PME_Event_SimPauseOn = 5
  Public Const PME_Event_SimPauseOff = 6
  Public Const PME_Event_SimAbEnd = 7    (not used?)
  Public Const PME_Event_SimStart = 8
  Public Const PME_Event_TranStart = 9
  Public Const PME_Event_TranEnd = 10
  Public Const PME_Event_EditModeOn = 11
  Public Const PME_Event_ModLoaded = 12
  Public Const PME_Event_AnimOn = 15
  Public Const PME_Event_AnimOff = 16
  Public Const PME_Event_MsgInfo = 20
  Public Const PME_Event_MsgOk = 21
```

```
  Public Const PME_Event_MsgOkCancel = 22
  Public Const PME_Event_MsgRetryCancel = 23
  Public Const PME_Event_MsgYesNo = 24
  Public Const PME_Event_MsgYesNoCancel = 25
  Public Const PME_Event_MsgAbortRetryCancel = 26
  Public Const PME_Event_MsgPrompt = 30

---------------------------------------------------------------------------------------
RunTime Tables
---------------------------------------------------------------------------------------
  Public Const RTI_Table_Locations = 1
  Public Const RTI_Table_LocSingle = 2
  Public Const RTI_Table_LocMulti = 3
  Public Const RTI_Table_Resources = 5
  Public Const RTI_Table_ResStates = 6
  Public Const RTI_Table_NodeEntry = 7
  Public Const RTI_Table_ArrivalFail = 8
  Public Const RTI_Table_EntActivity = 9
  Public Const RTI_Table_EntStates = 10
  Public Const RTI_Table_Variables = 12
  Public Const RTI_Table_Logs = 13
  Public Const RTI_Table_LocCost = 14
  Public Const RTI_Table_ResCost = 16
  Public Const RTI_Table_EntActCost = 17

---------------------------------------------------------------------------------------
RunTime Fields
---------------------------------------------------------------------------------------
  Locations (1)
  Public Const RTI_Field_LocTotEntry = 1
  Public Const RTI_Field_LocAvgMinPer = 2
  Public Const RTI_Field_LocAvgContent = 3
  Public Const RTI_Field_LocMaxContent = 4
```

Public Const RTI_Field_LocCurContent = 5
Public Const RTI_Field_LocPctUtil = 6

Single Capacity Locations (2)
Public Const RTI_Field_SLocPctOper = 1
Public Const RTI_Field_SLocPctSetup = 2
Public Const RTI_Field_SLocPctIdle = 3
Public Const RTI_Field_SLocPctWait = 4
Public Const RTI_Field_SLocPctBlock = 5
Public Const RTI_Field_SLocPctDown = 6

Multiple Capacity Locations (3)
Public Const RTI_Field_MLocPctEmpty = 1
Public Const RTI_Field_MLocPctPartFull = 2
Public Const RTI_Field_MLocPctFull = 3
Public Const RTI_Field_MLocPctDown = 4

Resources (5)
Public Const RTI_Field_ResTimesUsed = 1
Public Const RTI_Field_ResAvgMinPer = 2
Public Const RTI_Field_ResTrvl2Use = 3
Public Const RTI_Field_ResTrvl2Park = 4
Public Const RTI_Field_ResPctBlockTrvl = 5
Public Const RTI_Field_ResPctUtil = 6

Resource States (6)
Public Const RTI_Field_RStPctUse = 1
Public Const RTI_Field_RStPctTrvl2Use = 2
Public Const RTI_Field_RStPctTrvl2Pk = 3
Public Const RTI_Field_RStPctIdle = 4
Public Const RTI_Field_RStPctDown = 5

Node Entries (7)

Public Const RTI_Field_NodTotEntry = 1
Public Const RTI_Field_NodBlockEntry = 2

Failed Arrivals (8)
Public Const RTI_Field_FailArrivlTot = 1

Entity Activity (9)
Public Const RTI_Field_EntActTotExit = 1
Public Const RTI_Field_EntActCurQty = 2
Public Const RTI_Field_EntActAvgMinTot = 3
Public Const RTI_Field_EntActAvgMinMove = 4
Public Const RTI_Field_EntActAvgWait4Res = 5
Public Const RTI_Field_EntActAvgMinOper = 6
Public Const RTI_Field_EntActAvgMinBlock = 7

Entity States (10)
Public Const RTI_Field_EntStPctMove = 1
Public Const RTI_Field_EntStPctWaitRes = 2
Public Const RTI_Field_EntStPctOper = 3
Public Const RTI_Field_EntStPctBlock = 4

Variables (12)
Public Const RTI_Field_VarTotChange = 1
Public Const RTI_Field_VarAvgMinPer = 2
Public Const RTI_Field_VarMinValue = 3
Public Const RTI_Field_VarMaxValue = 4
Public Const RTI_Field_VarCurValue = 5
Public Const RTI_Field_VarAvgValue = 6

Logs (13)
Public Const RTI_Field_LogNumObserv = 1
Public Const RTI_Field_LogMinValue = 2
Public Const RTI_Field_LogMaxValue = 3

```
  Public Const RTI_Field_LogAvgValue = 4

  Location Costing (14)
  Public Const RTI_Field_LCostOperCost = 1
  Public Const RTI_Field_LCostOperPct = 2
  Public Const RTI_Field_LCostResCost = 3
  Public Const RTI_Field_LCostResPct = 4
  Public Const RTI_Field_LCostTotCost = 5
  Public Const RTI_Field_LCostTotPct = 6

  Resource Costing (16)
  Public Const RTI_Field_RCostNonUseCost = 1
  Public Const RTI_Field_RCostNonUsePct = 2
  Public Const RTI_Field_RCostUsageCost = 3
  Public Const RTI_Field_RCostUsagePct = 4
  Public Const RTI_Field_RCostTotCost = 5
  Public Const RTI_Field_RCostTotPct = 6

  Entity Activity Costing (17)
  Public Const RTI_Field_ECostTotCost = 1
  Public Const RTI_Field_ECostTotPct = 2

-------------------------------------------------------------------------------
Data Tables
-------------------------------------------------------------------------------
  Public Const PMD_Table_Locations = 1
  Public Const PMD_Table_Entities = 2
  Public Const PMD_Table_PathNetworks = 3
  Public Const PMD_Table_Resources = 4
  Public Const PMD_Table_Arrivals = 6
  Public Const PMD_Table_Shifts = 7
  Public Const PMD_Table_Attributes = 8
  Public Const PMD_Table_Variables = 9
```

```
Public Const PMD_Table_Arrays = 10
Public Const PMD_Table_Macros = 11
Public Const PMD_Table_Subroutines = 12
Public Const PMD_Table_ArrivalCycles = 13
Public Const PMD_Table_TableFunctions = 14
Public Const PMD_Table_UserDistribs = 15
Public Const PMD_Table_ExternalFiles = 16
Public Const PMD_Table_Streams = 17
Public Const PMD_Table_GenInfo = 18
Public Const PMD_Table_Processing = 19
Public Const PMD_Table_Routing = 20
Public Const PMD_Table_ModelParams = 21
Public Const PMD_Table_Scenarios = 22
Public Const PMD_Table_SimOptions = 23
Public Const PMD_Table_LocClockDTs = 25
Public Const PMD_Table_LocEntryDTs = 26
Public Const PMD_Table_LocUsageDTs = 27
Public Const PMD_Table_LocSetupDTs = 28
Public Const PMD_Table_LocCalledDTs = 63
Public Const PMD_Table_ResClockDTs = 31
Public Const PMD_Table_ResUsageDTs = 32
Public Const PMD_Table_ResWorkSearch = 33
Public Const PMD_Table_ResParkSearch = 34
Public Const PMD_Table_ResNodeLogic = 35
Public Const PMD_Table_ResPoints = 36
Public Const PMD_Table_ShiftLocations = 38
Public Const PMD_Table_ShiftResources = 39
Public Const PMD_Table_ShiftFiles = 40
Public Const PMD_Table_SubRoutineParams = 41
Public Const PMD_Table_ArrivalCycleSubtable = 42
Public Const PMD_Table_FunctionSubtable = 43
Public Const PMD_Table_UserDistribSubtable = 44
Public Const PMD_Table_LocationGraphics = 45
```

Public Const PMD_Table_ResourceGraphics = 46
Public Const PMD_Table_EntityGraphics = 47
Public Const PMD_Table_BackgroundGraphics = 49
Public Const PMD_Table_ModelDefaults = 50
Public Const PMD_Table_PathSegments = 51
Public Const PMD_Table_PathInterfaces = 52
Public Const PMD_Table_PathMappings = 53
Public Const PMD_Table_PathNodes = 54
Public Const PMD_Table_Views = 55
Public Const PMD_Table_ScenarioParams = 56
Public Const PMD_Table_PathMapDest = 57
Public Const PMD_Table_WorkSearchLocs = 58
Public Const PMD_Table_ParkNodes = 59
Public Const PMD_Table_VariableGraphics = 60
Public Const PMD_Table_QJoints = 61

Data Fields

Downtimes (25 thru 32)
Locations & Resources Tables:
Public Const PMD_Field_DTFrequency = 1 Applies To:(Clock, Entry, Usage)
Public Const PMD_Field_DTFirstTime = 2 Applies To:(Clock, Entry, Usage)
Public Const PMD_Field_DTPriority = 3 Applies To:(Clock, Usage, Called)
Public Const PMD_Field_DTScheduled = 4 Applies To:(Clock, Called)
Public Const PMD_Field_DTLogic = 5 Applies To:(Clock, Entry, Usage, Setup, Called)
Public Const PMD_Field_DTDisable = 6 Applies To:(Clock, Entry, Usage, Setup)
Public Const PMD_Field_DTName = 11 Applies To:(Called)
Locations Table Only:
Public Const PMD_Field_DTEntity = 7 Applies To:(Setup)
Public Const PMD_Field_DTPriorEnt = 8 Applies To:(Setup)
Resources Table Only:
Public Const PMD_Field_DTList = 9 Applies To:(Clock, Usage)

Public Const PMD_Field_DTNode = 10 Applies To:(Clock, Usage)

Locations (1)

Public Const PMD_Field_LocName = 2

Public Const PMD_Field_LocCapacity = 3

Public Const PMD_Field_LocUnits = 4

Public Const PMD_Field_LocStats = 9 (see Rules constants)

Public Const PMD_Field_LocIncoming = 10 (see Rules constants)

Public Const PMD_Field_LocIncAttrib = 11 (see Rules constants)

Public Const PMD_Field_LocQueOutput = 12 (see Rules constants)

Public Const PMD_Field_LocQueOutAttrib = 13 (see Rules constants)

Public Const PMD_Field_LocSelectUnit = 14 (see Rules constants)

Public Const PMD_Field_LocNotes = 15

Public Const PMD_Field_LocCostRate = 16

Public Const PMD_Field_LocCostTimeUnits = 20 (see Rules constants)

Entities (2)

Public Const PMD_Field_EntGfxNum = 1

Public Const PMD_Field_EntName = 2

Public Const PMD_Field_EntSpeed = 3

Public Const PMD_Field_EntStats = 4 (see Rules constants)

Public Const PMD_Field_EntNotes = 5

Public Const PMD_Field_EntInitCost = 8

Path Network (3)

Public Const PMD_Field_PathColor = 1

Public Const PMD_Field_PathVisible = 2

Public Const PMD_Field_PathName = 3

Public Const PMD_Field_PathType = 4 (see Rules constants)

Public Const PMD_Field_PathBasis = 5 (see Rules constants)

Public Const PMD_Field_CraneBridgeSeparation = 20

Public Const PMD_Field_CraneHideBridge = 21

Public Const PMD_Field_CraneHideRail = 22

Public Const PMD_Field_CraneRailWidth = 23
Public Const PMD_Field_CraneRailBorderColor = 24
Public Const PMD_Field_CraneRailFillColor = 25
Public Const PMD_Field_CraneBridgeWidth = 26
Public Const PMD_Field_CraneBridgeBorderColor = 27

Path Segments (51)
Public Const PMD_Field_PathSegFrom = 1
Public Const PMD_Field_PathSegTo = 2
Public Const PMD_Field_PathSegBiDi = 3 BiDi = BiDirectional
Public Const PMD_Field_PathSegSpeedFactor = 4
Public Const PMD_Field_PathSegDistance = 5
Public Const PMD_Field_PathSegTime = 6

Path Interfaces (52)
Public Const PMD_Field_PathXfaceNode = 1
Public Const PMD_Field_PathXfaceLocation = 2

Path Mapping (53)
Public Const PMD_Field_PathMapFromNode = 1
Public Const PMD_Field_PathMapSegmentNum = 2

Path Mapping Destinations (57)
Public Const PMD_Field_PathMapDestNode = 3

Path Nodes (54)
Public Const PMD_Field_PathNodeName = 1
Public Const PMD_Field_PathNodeCapacity = 2
Public Const PMD_Field_PathNodeXPosition = 3
Public Const PMD_Field_PathNodeYPosition = 4
Public Const PMD_Field_CraneNodeRailPosition = 5
Public Const PMD_Field_CraneNodeBridgePosition = 6

Resources (4)

Public Const PMD_Field_ResName = 2

Public Const PMD_Field_ResUnits = 3

Public Const PMD_Field_ResStats = 6 (see Rules constants)

Public Const PMD_Field_ResNetwork = 7

Public Const PMD_Field_ResSearch = 8 (see Rules constants)

Public Const PMD_Field_ResEntSearch = 9 (see Rules constants)

Public Const PMD_Field_ResEntSrchMinAttrib = 10

Public Const PMD_Field_ResEntSrchMaxAttrib = 11

Public Const PMD_Field_ResSpeedEmpty = 12

Public Const PMD_Field_ResSpeedFull = 13

Public Const PMD_Field_ResAccel = 14

Public Const PMD_Field_ResDecel = 15

Public Const PMD_Field_ResPickupTime = 16

Public Const PMD_Field_ResDepositTime = 17

Public Const PMD_Field_ResHomeNode = 18

Public Const PMD_Field_ResHomeIfIdle = 19 (see Rules constants)

Public Const PMD_Field_ResOffShiftNode = 20

Public Const PMD_Field_ResBreakNode = 21

Public Const PMD_Field_ResNotes = 26

Public Const PMD_Field_ResCost = 27 (Cost per Cost Time Unit)

Public Const PMD_Field_ResOTCost = 28

Public Const PMD_Field_ResCostPerUse = 29

Public Const PMD_Field_ResCostTimeUnit = 30

Resource Work Search (33)

Public Const PMD_Field_ResWorkSrchNode = 1

Public Const PMD_Field_ResWorkSrchType = 2

Work Search Location List (58)

Public Const PMD_Field_WorkSrchLocName = 3

Resource Park Search (34)

Public Const PMD_Field_ResParkSrchNode = 1

Park Search Node List (59)

Public Const PMD_Field_ResParkNodeName = 2

Resource Node Logic (35)

Public Const PMD_Field_ResNLogicNode = 1

Public Const PMD_Field_ResNLogicEntry = 2

Public Const PMD_Field_ResNLogicExit = 3

Resource Points (36)

Public Const PMD_Field_PointsNode = 1

Public Const PMD_Field_PointsXPosition = 2

Public Const PMD_Field_PointsYPosition = 3

Arrivals (6)

Public Const PMD_Field_ArrivalEntName = 1

Public Const PMD_Field_ArrivalLocName = 2

Public Const PMD_Field_ArrivalQtyEach = 3

Public Const PMD_Field_ArrivalCycles = 4 Subtable: ArrivalCycles

Public Const PMD_Field_ArrivalFirstTime = 5

Public Const PMD_Field_ArrivalOccurrences = 6

Public Const PMD_Field_ArrivalFrequency = 7

Public Const PMD_Field_ArrivalLogic = 8

Public Const PMD_Field_ArrivalDisable = 9 (see Rules constants)

Public Const PMD_Field_ArrivalTimeBasis = 10

Public Const PMD_Field_ArrivalMinutes = 11

Public Const PMD_Field_ArrivalHour = 12

Public Const PMD_Field_ArrivalWeekDay = 13

Public Const PMD_Field_ArrivalMonthDay = 14

Public Const PMD_Field_ArrivalWeek = 15

Public Const PMD_Field_ArrivalMonth = 16

Public Const PMD_Field_ArrivalYear = 17

Public Const PMD_Field_ArrivalVariation = 18
Public Const PMD_Field_ArrivalOffset = 19
Public Const PMD_Field_ArrivalRepeatType = 20 (see Rules constants)

Arrival Cycles (13)
Public Const PMD_Field_ArrCycID = 1
Public Const PMD_Field_ArrCycQty = 2 (see Rules constants)
Public Const PMD_Field_ArrCycCumulative = 3 (see Rules constants)

Arrival Cycle SubTable (42)
Public Const PMD_Field_ArrCycTime = 5
Public Const PMD_Field_ArrCycQtyPercent = 6

Shift Assignments (7)
Public Const PMD_Field_ShiftPriority = 1
Public Const PMD_Field_ShiftOffShiftPriority = 2
Public Const PMD_Field_ShiftStartBreakPriority = 3
Public Const PMD_Field_ShiftBreakPriority = 4
Public Const PMD_Field_ShiftPreOffShiftLogic = 5
Public Const PMD_Field_ShiftOffShiftLogic = 6
Public Const PMD_Field_ShiftPreBreakLogic = 7
Public Const PMD_Field_ShiftBreakLogic = 8
Public Const PMD_Field_ShiftDisable = 9 (see Rules constants)

Shift Locations (38)
Public Const PMD_Field_ShiftLocationNum = 10

Shift Resources (39)
Public Const PMD_Field_ShiftResourceNum = 11
Public Const PMD_Field_ShiftResourceUnits = 12

Shift Files (40)
Public Const PMD_Field_ShiftFileNum = 13

Public Const PMD_Field_ShiftStartTime = 14

Attributes (8)

Public Const PMD_Field_AttribID = 1

Public Const PMD_Field_AttribType = 2 (see Rules constants)

Public Const PMD_Field_AttribClass = 3 (see Rules constants)

Public Const PMD_Field_AttribNotes = 4

Variables (9)

Public Const PMD_Field_VarGraphicNum = 1

Public Const PMD_Field_VarID = 2

Public Const PMD_Field_VarType = 3 (see Rules constants)

Public Const PMD_Field_VarInitValue = 4 (see Rules constants)

Public Const PMD_Field_VarStatsLevel = 5

Public Const PMD_Field_VarStatsBasis = 6 (see Rules constants)

Public Const PMD_Field_VarNotes = 7

Arrays (10)

Public Const PMD_Field_ArrayID = 1

Public Const PMD_Field_ArrayDimensions = 2

Public Const PMD_Field_ArrayType = 3 (see Rules constants)

Public Const PMD_Field_ArrayNotes = 4

Public Const PMD_Field_ArrayImportPath = 5

Public Const PMD_Field_ArraySheetName = 6

Public Const PMD_Field_ArrayFirstDataCell = 7

Public Const PMD_Field_ArrayStartCell = 8

Public Const PMD_Field_ArrayEndCell = 9

Public Const PMD_Field_ArrayExportPath = 10

Public Const PMD_Field_ArrayExSheetName = 11

Public Const PMD_Field_ArrayExStartCell = 12

Public Const PMD_Field_ArrayExEndCell = 13

Public Const PMD_Field_ArrayImDBConnectString = 14

Public Const PMD_Field_ArrayImDBQuery_ProcString = 15

Public Const PMD_Field_ArrayImportType = 17
Public Const PMD_Field_ArrayExportType = 18 Not yet implemented
Public Const PMD_Field_ArrayImDBQueryOrProc = 19
Public Const PMD_Field_ArrayPersist = 20
Public Const PMD_Field_ArrayExportLastRep = 21
Public Const PMD_Field_ArrayDisableImport = 22
Public Const PMD_Field_ArrayDisableExport = 23

Macros (11)
Public Const PMD_Field_MacroID = 1
Public Const PMD_Field_MacroText = 2
Public Const PMD_Field_MacroGroup = 4
Public Const PMD_Field_MacroRTIName = 5
Public Const PMD_Field_MacroRTIPrompt = 6
Public Const PMD_Field_MacroRTIMinValue = 7
Public Const PMD_Field_MacroRTIMaxValue = 8
Public Const PMD_Field_MacroRTIType = 9 (see Rules constants)

SubRoutines (12)
Public Const PMD_Field_SubRtnID = 1
Public Const PMD_Field_SubRtnOperation = 2
Public Const PMD_Field_SubRtnType = 4 (see Rules constants)

SubRoutine Parameters (41)
Public Const PMD_Field_SubRtnParamName = 5
Public Const PMD_Field_SubRtnParamType = 6 (see Rules constants)

Table Functions (14)
Public Const PMD_Field_TableFunID = 1

Table Functions Subtable (43)
Public Const PMD_Field_TableFunIndepVal = 3
Public Const PMD_Field_TableFunDepVal = 4

User Distributions (15)

Public Const PMD_Field_UserDistID = 1

Public Const PMD_Field_UserDistType = 2 (see Rules constants)

Public Const PMD_Field_UserDistCumulative = 3 (see Rules constants)

User Distributions Subtable (44)

Public Const PMD_Field_UserDistPercentage = 4

Public Const PMD_Field_UserDistValue = 5

External Files (16)

Public Const PMD_Field_XfilesID = 1

Public Const PMD_Field_XfilesType = 2 (see Rules constants)

Public Const PMD_Field_XfilesPath = 3

Public Const PMD_Field_XfilesPrompt = 4

Public Const PMD_Field_XfilesNotes = 5

Streams (17)

Public Const PMD_Field_StreamNumber = 1

Public Const PMD_Field_StreamSeedNum = 2

Public Const PMD_Field_StreamReset = 3 (see Rules constants)

General Information (18)

Public Const PMD_Field_GenInfoTitle = 1

Public Const PMD_Field_GenInfoDfltTimeUnits = 2 (see Rules constants)

Public Const PMD_Field_GenInfoDfltDistUnits = 3 (see Rules constants)

Public Const PMD_Field_GenInfoGLibFile = 4

Public Const PMD_Field_GenInfoInitLogic = 5

Public Const PMD_Field_GenInfoTerminateLogic = 6

Public Const PMD_Field_GenInfoNotes = 7

Public Const PMD_Field_GenInfoModFile = 9 (Read Only)

Public Const PMD_Field_GenInfoZoomPct = 10

Processing (19)

Public Const PMD_Field_ProcEntName = 1

Public Const PMD_Field_ProcPreempt = 2 (see Rules constants)

Public Const PMD_Field_ProcLocName = 3

Public Const PMD_Field_ProcOpLogic = 4

Public Const PMD_Field_ProcRouting = 20 Subtable: Routing

Routing (20)

Public Const PMD_Field_RoutEntName = 6

Public Const PMD_Field_RoutLocName = 7

Public Const PMD_Field_RoutPriority = 8

Public Const PMD_Field_RoutNewBlock = 9 (see Rules constants)

Public Const PMD_Field_RoutNewEntity = 10 (see Rules constants)

Public Const PMD_Field_RoutQuantity = 11

Public Const PMD_Field_RoutSelection = 12 (see Rules constants)

Public Const PMD_Field_RoutProbability = 13

Public Const PMD_Field_RoutCondition = 14

Public Const PMD_Field_RoutMoveLogic = 15

Simulation Options (23)

Public Const PMD_Field_SimOptOutputPath = 1

Public Const PMD_Field_SimOptTimeBasis = 2 (see Rules constants)

Public Const PMD_Field_SimOptRunHours = 3 Time Only

Public Const PMD_Field_SimOptWarmupPeriod = 4 Time Only

Public Const PMD_Field_SimOptWarmupMonth = 5 Calendar Date

Public Const PMD_Field_SimOptWarmupDay = 6 Calendar Date

Public Const PMD_Field_SimOptWarmupYear = 7 Calendar Date

Public Const PMD_Field_SimOptWarmupHour = 8 Calendar Date

Public Const PMD_Field_SimOptWarmupMin = 9 Calendar Date

Public Const PMD_Field_SimOptBeginMonth = 10 Calendar Date

Public Const PMD_Field_SimOptBeginDay = 11 Calendar Date

Public Const PMD_Field_SimOptBeginYear = 12 Calendar Date

Public Const PMD_Field_SimOptBeginHour = 13 Calendar Date

Public Const PMD_Field_SimOptBeginMin = 14 Calendar Date

Public Const PMD_Field_SimOptEndMonth = 15 Calendar Date

Public Const PMD_Field_SimOptEndDay = 16 Calendar Date

Public Const PMD_Field_SimOptEndYear = 17 Calendar Date

Public Const PMD_Field_SimOptEndHour = 18 Calendar Date

Public Const PMD_Field_SimOptEndMin = 19 Calendar Date

Public Const PMD_Field_SimOptRptMethod = 20 (see Rules constants)

Public Const PMD_Field_SimOptRptIntLength = 21

Public Const PMD_Field_SimOptNumReps = 22

Public Const PMD_Field_SimOptClockPrecision = 23 (see Rules constants)

Public Const PMD_Field_SimOptClockPrecUnits = 24 (see Rules constants)

Public Const PMD_Field_SimOptDisTimeSeries = 25 (see Rules constants)

Public Const PMD_Field_SimOptDisAnimation = 26 (see Rules constants)

Public Const PMD_Field_SimOptDisCosts = 27 (see Rules constants)

Public Const PMD_Field_SimOptPause = 28 (see Rules constants)

Public Const PMD_Field_SimOptDisplayNotes = 29 (see Rules constants)

Public Const PMD_Field_SimOptEnableWarmup = 30 (see Rules constants)

Public Const PMD_Field_WarmDay = 31 Weekly Time

Public Const PMD_Field_WarmWeek = 32 Weekly Time

Public Const PMD_Field_BeginDay = 33 Weekly Time

Public Const PMD_Field_BeginWeek = 34 Weekly Time

Public Const PMD_Field_EndDay = 35 Weekly Time

Public Const PMD_Field_EndWeek = 36 Weekly Time

Public Const PMD_Field_SimOptDaylightSav = 37

Public Const PMD_Field_SimOptGenAnimScript = 38

Public Const PMD_Field_SimOptCommonRandom = 39

Public Const PMD_Field_SimOptSkipResDTs = 40

Public Const PMD_Field_SimOptDisArrays = 41

Public Const PMD_Field_SimOptInitAnimSpeed = 42

Public Const PMD_Field_SimOptTrace = 43

Public Const PMD_Field_SimOptRecompileMap = 44

Public Const PMD_Field_SimOptRunName = 45

Public Const PMD_Field_SimOptMinitab = 46

Public Const PMD_Field_SimOptOutputViewer = 47

Public Const PMD_Field_SimOptOVversionNum = 48

Scenarios (22)

Public Const PMD_Field_ScenName = 1

Public Const PMD_Field_ScenDisable = 2

Scenario Parameters (56)

Public Const PMD_Field_ScenParamValue = 1

Views (55)

Public Const PMD_Field_ViewName = 1

Public Const PMD_Field_ViewXOffset = 2

Public Const PMD_Field_ViewYOffset = 3

Public Const PMD_Field_ViewWidth = 4

Public Const PMD_Field_ViewHeight = 5

Graphics - Library (Type = 1)

Public Const PMD_Field_LGfxType = 1

Public Const PMD_Field_LGfxWidthI = 2

Public Const PMD_Field_LGfxHeightI = 3

Public Const PMD_Field_LGfxWidthR = 4

Public Const PMD_Field_LGfxHeightR = 5

Public Const PMD_Field_LGfxRotation = 6

Public Const PMD_Field_LGfxID = 7

Public Const PMD_Field_LGfxHotXpos = 8

Public Const PMD_Field_LGfxHotYpos = 9

Public Const PMD_Field_LGfxXpos = 10

Public Const PMD_Field_LGfxYpos = 11

Public Const PMD_Field_LGfxColor = 12 (see Rules constants)

Public Const PMD_Field_LGfxEntConvWidth = 20

Public Const PMD_Field_LGfxEntConvLength = 21

Graphics - Queue/Conveyor (Type = 2)

Public Const PMD_Field_QGfxType = 1

Public Const PMD_Field_QGfxConvey = 2 Boolean (see Rules constants)

Public Const PMD_Field_QGfxSpeed = 3 Conveyor Only

Public Const PMD_Field_QGfxLength = 4 Conveyor Only

Public Const PMD_Field_QGfxAccum = 5 Boolean (see Rules constants)

Public Const PMD_Field_QGfxOrient = 6

Public Const PMD_Field_QGfxBColor = 7 (see Rules constants)

Public Const PMD_Field_QGfxFColor = 8 (see Rules constants)

Public Const PMD_Field_QGfxStyle = 9 (see Rules constants)

Public Const PMD_Field_QGfxWidth = 10

Public Const PMD_Field_QGfxInvis = 11

Queue/Conveyor Joints (61)

Public Const PMD_Field_QJointXPos = 1

Public Const PMD_Field_QJointYPos = 2

Graphics - Gauge/Tank (Type = 3)

Public Const PMD_Field_GGfxType = 1

Public Const PMD_Field_GGfxLeft = 2

Public Const PMD_Field_GGfxTop = 3

Public Const PMD_Field_GGfxRight = 4

Public Const PMD_Field_GGfxBottom = 5

Public Const PMD_Field_GGfxBColor = 6 (see Rules constants)

Public Const PMD_Field_GGfxEColor = 7 (see Rules constants)

Public Const PMD_Field_GGfxFColor = 8 (see Rules constants)

Public Const PMD_Field_GGfxMin = 9

Public Const PMD_Field_GGfxMax = 10

Public Const PMD_Field_GGfxDirxn = 11 (see Rules constants)

Public Const PMD_Field_GGfxShowScale = 12 Boolean (see Rules constants)

Public Const PMD_Field_GGfxShowBrdr = 13 Boolean (see Rules constants)

Public Const PMD_Field_GGfxIsTank = 14 Boolean (see Rules constants)

Graphics - Counter (Type = 4)

Public Const PMD_Field_CGfxType = 1

Public Const PMD_Field_CGfxLeft = 10

Public Const PMD_Field_CGfxTop = 11

Public Const PMD_Field_CGfxRight = 12

Public Const PMD_Field_CGfxBottom = 13

Public Const PMD_Field_CGfxFrmType = 14 (see Rules constants)

Public Const PMD_Field_CGfxFrmShape = 15 (see Rules constants)

Public Const PMD_Field_CGfxFrmColor = 16 (see Rules constants)

Public Const PMD_Field_CGfxFrmBColor = 17 (see Rules constants)

Public Const PMD_Field_CGfxFontSize = 20

Public Const PMD_Field_CGfxFontName = 21

Public Const PMD_Field_CGfxFontColor = 22 (see Rules constants)

Graphics - Text (Type = 5)

Public Const PMD_Field_TGfxType = 1

Public Const PMD_Field_TGfxLblTxt = 2

Public Const PMD_Field_TGfxLeft = 10

Public Const PMD_Field_TGfxTop = 11

Public Const PMD_Field_TGfxRight = 12

Public Const PMD_Field_TGfxBottom = 13

Public Const PMD_Field_TGfxFrmType = 14 (see Rules constants)

Public Const PMD_Field_TGfxFrmShape = 15 (see Rules constants)

Public Const PMD_Field_TGfxFrmColor = 16 (see Rules constants)

Public Const PMD_Field_TGfxFrmBColor = 17 (see Rules constants)

Public Const PMD_Field_TGfxFontSize = 20

Public Const PMD_Field_TGfxFontName = 21

Public Const PMD_Field_TGfxFontColor = 22 (see Rules constants)

Graphics - Status Light (Type = 6)

Public Const PMD_Field_SGfxType = 1

Public Const PMD_Field_SGfxXpos = 2

Public Const PMD_Field_SGfxYpos = 3

Graphics - Part Spot (Type = 7)

Public Const PMD_Field_PGfxType = 1

Public Const PMD_Field_PGfxXpos = 2

Public Const PMD_Field_PGfxYpos = 3

Graphics - Region (Type = 8)

Public Const PMD_Field_RGfxType = 1

Public Const PMD_Field_RGfxLeft = 2

Public Const PMD_Field_RGfxTop = 3

Public Const PMD_Field_RGfxRight = 4

Public Const PMD_Field_RGfxBottom = 5

Rules

Data Types

Public Const PMD_Rule_TypeNone = 0

Public Const PMD_Rule_TypeInteger = 2

Public Const PMD_Rule_TypeReal = 4

Public Const PMD_Rule_TypeString = 6

Public Const PMD_Rule_TypeExpression = 7

Public Const PMD_Rule_TypeInteractive = 8

Boolean

Public Const PMD_Rule_NoFalse = 0

Public Const PMD_Rule_YesTrue = 1

Stats Type: Locations, Entities, Resources, Variables

Public Const PMD_Rule_StatsNone = 1 (Loc, Ent, Res, Var)

Public Const PMD_Rule_StatsBasic = 2 (Loc, Ent, Var)

Public Const PMD_Rule_StatsSummary = 2 (Res)

Public Const PMD_Rule_StatsTimeSeries = 3 (Loc, Ent, Var)

Public Const PMD_Rule_StatsByUnit = 3 (Res)

Stats Basis: Variables

Public Const PMD_Rule_StatsBasisNone = 0

Public Const PMD_Rule_StatsBasisTime = 1 Time Weighted

Public Const PMD_Rule_StatsBasisObserv = 2 Observation Based

Incoming Entities: Locations

Public Const PMD_Rule_IncomEntOldestByPriority = 1

Public Const PMD_Rule_IncomEntNotUsed = 2 was "most contents"

Public Const PMD_Rule_IncomEntLeastAvailCapacity = 3

Public Const PMD_Rule_IncomEntRandom = 4

Public Const PMD_Rule_IncomEntLastLoc = 5

Public Const PMD_Rule_IncomEntMinAttrib = 6

Public Const PMD_Rule_IncomEntMaxAttrib = 7

Incoming Entity Attribute Options

Public Const PMD_Rule_AttribQty = 1

Public Const PMD_Rule_AttribCust = 2

Queue Output: Locations

Public Const PMD_Rule_QOutNone = 1

Public Const PMD_Rule_QOutFIFO = 2

Public Const PMD_Rule_QOutLIFO = 3

Public Const PMD_Rule_QOutMaxAttrib = 4

Public Const PMD_Rule_QOutMinAttrib = 5

Public Const PMD_Rule_QOutByType = 6

Unit Selection: Locations

Public Const PMD_Rule_USelNone = 0

Public Const PMD_Rule_USelLongestEmpty = 1

Public Const PMD_Rule_USelRandom = 2

Public Const PMD_Rule_USelByTurn = 3

Public Const PMD_Rule_USelMostAvailCapacity = 4

Public Const PMD_Rule_USelFewestEntries = 5

Public Const PMD_Rule_USelFirstAvail = 6

Time Units: Locations

Public Const PMD_Rule_TimeUSeconds = 1

Public Const PMD_Rule_TimeUMinutes = 2

Public Const PMD_Rule_TimeUHours = 3

Public Const PMD_Rule_TimeUDays = 4

RTI Types: Macros

Public Const PMD_Rule_rtiTypeRange = 2

Public Const PMD_Rule_rtiTypeText = 3

Colors: Path Networks

Public Const PMD_Color_White = 16777215

Public Const PMD_Color_Black = 0

Public Const PMD_Color_Lt_Grey = 12632256

Public Const PMD_Color_Dk_Grey = 8421504

Public Const PMD_Color_Red = 255

Public Const PMD_Color_Maroon = 128

Public Const PMD_Color_Lt_Green = 65280

Public Const PMD_Color_Dk_Green = 32768

Public Const PMD_Color_Med_Blue = 16711680

Public Const PMD_Color_Dk_Blue = 8388608

Public Const PMD_Color_Cyan = 16776960

Public Const PMD_Color_Teal = 8421376

Public Const PMD_Color_Yellow = 65535

Public Const PMD_Color_Gold = 32896

Public Const PMD_Color_Pink = 16711935

Public Const PMD_Color_Purple = 8388736

Path Types :Path Networks

Public Const PMD_Rule_PathTypeNoPassing = 0

Public Const PMD_Rule_PathTypePassing = 1

Public Const PMD_Rule_PathTypeCrane = 2

Basis: Path Networks

Public Const PMD_Rule_PathBasisTime = 0

Public Const PMD_Rule_PathBasisSpeedDist = 1

BiDirectional: Path Networks

Public Const PMD_Rule_PathUniDi = 0 One Way

Public Const PMD_Rule_PathBiDi = 1 Two Way

Resource Search: Resources

Public Const PMD_Rule_ResSrchClosest = 1 (Closest Resource)

Public Const PMD_Rule_ResSrchLongestIdle = 2

Public Const PMD_Rule_ResSrchLeastUtilized = 3

Entity Search: Resources

Public Const PMD_Rule_EntSrchOldest = 1

Public Const PMD_Rule_EntSrchClosest = 2

Public Const PMD_Rule_EntSrchMinAttrib = 3

Public Const PMD_Rule_EntSrchMaxAttrib = 4

Scheduling Options: Arrivals

Public Const PMD_Rule_RepeatNone = 0

Public Const PMD_Rule_RepeatDaily = 1

Public Const PMD_Rule_RepeatWeekly = 2

Type: Attributes

Public Const PMD_Rule_AttribTypeEnt = 1

Public Const PMD_Rule_AttribTypeLoc = 2

Quantity: Arrival Cycles

```
  Public Const PMD_Rule_ArrCycPercent = 0  Percentage
  Public Const PMD_Rule_ArrCycQuant = 1   Quantity

File Type: External Files
  Public Const PMD_Rule_FileTypeGenRead = 1
  Public Const PMD_Rule_FileTypeGenWrite = 2
  Public Const PMD_Rule_FileTypeEntLocMatrix = 3
  Public Const PMD_Rule_FileTypeArrival = 4
  Public Const PMD_Rule_FileTypeShift = 5
  Public Const PMD_Rule_FileTypeDLL = 6
  Public Const PMD_Rule_FileTypeExcel = 7

Distance/Size Units: General Info
  Public Const PMD_Rule_DistUnitFeet = 1
  Public Const PMD_Rule_DistUnitMeters = 2

Type: User Distributions
  Public Const PMD_Rule_UDistribDiscrete = 1
  Public Const PMD_Rule_UDistribContinuous = 2

Routing Rules
  Public Const PMD_Rule_RoutFirstAvail = 1
  Public Const PMD_Rule_RoutMostAvailCap = 2  Most Available Capacity
  Public Const PMD_Rule_RoutByTurn = 3
  Public Const PMD_Rule_RoutJoinRequest = 4
  Public Const PMD_Rule_RoutLoadRequest = 5
  Public Const PMD_Rule_RoutIfSend = 6
  Public Const PMD_Rule_RoutUntilFull = 7
  Public Const PMD_Rule_RoutAsAlternate = 8
  Public Const PMD_Rule_RoutProbability = 9
  Public Const PMD_Rule_RoutUserCondition = 10
  Public Const PMD_Rule_RoutRandom = 11
  Public Const PMD_Rule_RoutLongestEmpty = 12
```

Public Const PMD_Rule_RoutIfEmpty = 13
Public Const PMD_Rule_RoutContinue = 14
Public Const PMD_Rule_RoutAsBackup = 15
Public Const PMD_Rule_RoutDependent = 16

Sim Options - Time Basis
Public Const PMD_Rule_SimOptTimeOnly = 0
Public Const PMD_Rule_SimOptCalDate = 1
Public Const PMD_Rule_SimOptWeeklyTime = 2

Sim Options Reporting
Public Const PMD_Rule_SimRptStandard = 0
Public Const PMD_Rule_SimRptBatchMean = 1
Public Const PMD_Rule_SimRptPeriodic = 2

Sim Options Clock Precision
Public Const PMD_Rule_SimClockPrecNone = 0
Public Const PMD_Rule_SimClockPrec1 = 1
Public Const PMD_Rule_SimClockPrec10 = 2
Public Const PMD_Rule_SimClockPrec100 = 3
Public Const PMD_Rule_SimClockPrec1000 = 4
Public Const PMD_Rule_SimClockPrec10000 = 5
Public Const PMD_Rule_SimClockPrec100000 = 6
Public Const PMD_Rule_SimClockPrec1000000 = 7

Months - When used in ProModel, list starts on zero
Public Const PMD_Month_Jan = 0
Public Const PMD_Month_Feb = 1
Public Const PMD_Month_Mar = 2
Public Const PMD_Month_Apr = 3
Public Const PMD_Month_May = 4
Public Const PMD_Month_Jun = 5
Public Const PMD_Month_Jul = 6

```
  Public Const PMD_Month_Aug = 7
  Public Const PMD_Month_Sep = 8
  Public Const PMD_Month_Oct = 9
  Public Const PMD_Month_Nov = 10
  Public Const PMD_Month_Dec = 11

Days - When used in ProModel, list starts on zero
  Public Const PMD_Day_Sun = 0
  Public Const PMD_Day_Mon = 1
  Public Const PMD_Day_Tue = 2
  Public Const PMD_Day_Wed = 3
  Public Const PMD_Day_Thu = 4
  Public Const PMD_Day_Fri = 5
  Public Const PMD_Day_Sat = 6

Graphics Types
  Public Const PMD_GfxType_Lib = 1
  Public Const PMD_GfxType_Q = 2
  Public Const PMD_GfxType_Tank = 3
  Public Const PMD_GfxType_Count = 4
  Public Const PMD_GfxType_Text = 5
  Public Const PMD_GfxType_Light = 6
  Public Const PMD_GfxType_Spot = 7
  Public Const PMD_GfxType_Region = 8

Graphics - Frame Shapes
  Public Const PMD_GfxShape_Rect = 1
  Public Const PMD_GfxShape_RndRect = 2
  Public Const PMD_GfxShape_Ellipse = 3
  Public Const PMD_GfxShape_Diamond = 4

Graphics - Frame Type
  Public Const PMD_GfxFrame_None = 1
```

Public Const PMD_GfxFrame_Plain = 2

Public Const PMD_GfxFrame_Raised = 3

Public Const PMD_GfxFrame_Sunken = 4

Public Const PMD_GfxFrame_Shadow = 5

Public Const PMD_GfxFrame_NoBorder = 6

Graphics - Direction

Public Const PMD_GfxDir_Up = 1

Public Const PMD_GfxDir_Down = 2

Public Const PMD_GfxDir_Left = 3

Public Const PMD_GfxDir_Right = 4

Index

T

U

V

Z