Autodesk® Moldflow® Insight 2012

AMI Application Programming Interface (API)

Revision 1, 18 March 2012.

Contents

Application Programming Interface (API) 1

The Application Programming Interface (API) in Autodesk Moldflow Insight is an Object Linking and Embedding (OLE) programming interface that enables Autodesk Moldflow Insight functionality to be automated.

TIP: Access the API reference documentation from the application **Help** menu. Click **Help > API Reference** to open the **Application Programming Interface Documentation**.

You can manipulate Autodesk Moldflow Insight from scripts and third-party software.

To control Autodesk Moldflow Insight through the API, you must have access to an OLE automation client. Examples of such clients include:

- Windows Script Host that can process Visual Basic Script (VBS), JScript, and other programming languages
- Visual Basic for Applications (VBA), a fully featured client that is part of the Microsoft Office application suite
- Visual Basic (VB)
- Internet Explorer
- Perl
- Python

Autodesk Moldflow Insight can be automated by using the API in the following ways:

- Macros can be recorded and played from within the interface or from the Autodesk Moldflow Insight command line.
- VB scripts can be run from the Autodesk Moldflow Insight command line with or without command line arguments, or they can be run from within Autodesk Moldflow Insight as macros.

The OLE Automation Interface

The Object Linking and Embedding (OLE) automation interface to Synergy (the Autodesk Moldflow user interface) within the Autodesk Moldflow Application Programming Interface (API) is provided by an OLE automation client.

The following are examples of OLE automation clients:

- Windows Script Host that can process Visual Basic Script (VBS), JScript, and other programming languages
- Visual Basic for Applications (VBA), a fully featured client that is part of the Microsoft Office application suite

- Visual Basic (VB)
- Internet Explorer
- Perl
- Python

You can access Autodesk Moldflow Insight directly through the OLE interface by programming in a scripting language such as Visual Basic Script, or a programming language such as Visual Basic. While this is the most powerful and comprehensive way to access API functions, Autodesk Moldflow Insight also provides you with macro recording and playback functionality. This can ease your transition to using the API functionality and provide a more gradual introduction.

VBS is the common form of scripting language used in creating Autodesk Moldflow Insight API scripts. You can use the Autodesk Moldflow Insight command line to run scripts that take parameters.

If you opened an earlier version of Autodesk Moldflow Insight on your last access, and you try to run a macro or script with unsupported features, the OLE interface will automatically open that earlier version of Autodesk Moldflow Insight, and your macro or script will fail to execute.

NOTE: You can only have one version of Synergy open at any time.

Macros

Macro recording and playback enables you to repeat user-interface actions to automate common or repetitive tasks.

Macro recording and playback is built on top of the basic OLE automation interface and uses Visual Basic script as the recording and playback language.

By default, recorded macros are saved with a `.vbs` extension in the following folders (where **xxxx** is the software release):

- On Windows XP systems, **My Documents\My AMI xxxx Projects\scripts**
- On Windows Vista systems, **Documents\My AMI xxxx Projects\scripts**

Some user-interface functions are not suitable for macro recording, and some are not included in the recording architecture in the present release of Autodesk Moldflow Insight. In addition, certain API functionality that is available through the OLE automation interface is not available from the Synergy user interface.

Scripts that do not have command line arguments (input on the Autodesk Moldflow Insight command line) can be run as macros. If you need to create a script that uses parameters, you must write a script that takes command line arguments.

Macros

Macro recording and playback allows you to repeat user-interface actions to automate common or repetitive tasks.

Creating macros

Creating a macro allows you to automate tasks and repeat them.

NOTE: Not all Autodesk Moldflow Insight functionality is able to be recorded using macros.

---

1 Select **Tools tab > Automation panel > Record Macro** .

  The functionality you use will be recorded from now until you stop the macro recording.

2 Complete the task(s) that you want to perform.

3 Select **Tools tab > Automation panel > Stop Recording** .

  The **Save Macro** dialog appears.

4 Enter a name for the macro in the **File name** box and select **Save**.

  Your new script is saved with a (*.vbs) file extension, and can now be used as a Visual Basic script.

Playing macros or scripts using the menu

You can play a macro or a Visual Basic script in Autodesk Moldflow Insight using the menu, or using the Autodesk Moldflow Insight command line.

NOTE: You must use the Autodesk Moldflow Insight command line if the script requires command line arguments unless the script prompts for user input.

---

1 Select **Tools tab > Automation panel > Play Macro** .

  The **Open Macro** dialog appears.

  By default, macros are located in Windows XP, **My Documents\My AMI xxxx Projects\scripts**, or in Windows Vista, **Documents\My AMI xxxx Projects\scripts** , (where **xxxx** is the software release).

  By default, command line scripts are located in Windows XP, **My Documents\My AMI xxxx Projects\commands**, and in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release) and the **data\commands** folder of the Autodesk Moldflow Insight installation directory.

2 Select a macro/script, then select **Open**.

  The macro/script plays.

Running a macro or script from Windows Explorer

Because macros are just VBScripts, they can be invoked from Windows even if Autodesk Moldflow Insight is not running.

Scripts which assume that a particular study is already open, or that a particular result is already displayed, will probably not work.

NOTE: If your script uses parameters that are entered as command line arguments, you must run the script from the Autodesk Moldflow Insight command line.

1 Locate the macro or script using Windows Explorer.

By default, macros are located in Windows XP, **My Documents\My AMI xxxx Projects\scripts**, or in Windows Vista, **Documents\My AMI xxxx Projects\scripts** (where **xxxx** is the software release).

Command-line scripts are located in Windows XP, **My Documents\My AMI xxxx Projects\commands**, or in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release) and in the **data\commands** folder of the Autodesk Moldflow Insight installation directory.

2 Double-clicking on the required script.

Autodesk Moldflow Insight will start if it is not currently running and the script will open and run.

Assigning macros or Visual Basic scripts to toolbar buttons

Autodesk Moldflow Insight allows you to assign a macro or a Visual Basic script to a button on the **Tools tab > Assigned Macros panel**. This panel provides quick access to up to 10 different scripts.

NOTE: Scripts which use parameters which are input as command line arguments must be run from the Autodesk Moldflow Insight command line.

1 Click **Tools tab > Assigned Macros panel > Assign Macro**.

2 Select the button, 1 to 10, to which you want to assign a macro or Visual Basic script.

3 Specify the macro or script to be assigned:

- To assign a macro, select **Macro**, click ... at the right of the text box, then navigate to and select the ***.vbs** file you want.
- To assign a Visual Basic script, select **Command Line** and enter the Visual Basic script name in the box provided.

4 In the **Button tip** text box, enter a description for this macro/command button. This description will be displayed as a tooltip on the button.
5 Select **Save** to store the button assignment.
6 You can assign additional buttons by repeating steps 2 to 5 above.
7 Select **Close** to close the dialog.
8 Click on one of the numbered toolbar buttons to run the associated macro or script.

Macros

You can assign a command or script file to a button so it can be quickly accessed.

Assign Macro/Command Button dialog

This dialog is used to assign a command or script file, based on the Application Programming Interface (API) capabilities of Autodesk Moldflow Insight, to one of the quick access buttons on the **User Macro/Command Buttons** toolbar.

To access this dialog, click **Tools > Assign Macro/Command Button** .

TIP: To view a list of the standard commands supplied with Autodesk Moldflow Insight, click **View > Command Line,** enter **help** on the command line and press **Go**.

Limitations in API functionality

The functionality of the Application Programming Interface (API) has certain limitations.

The following table lists the features that are not supported through the API. Command line scripts written in Visual Basic (VB) can access more functionality than macros recorded in Autodesk Moldflow Insight.

All features not listed are available for inclusion in both macros and VB scripts.

Feature	Macros	VB Scripts
File Menu Functions: ■ Organize Project ■ Project Properties ■ Options (All options)	Not available	Not available
Print Functions: ■ Print ■ Print Preview ■ Print Setup	Not available	Available

Feature	Macros	VB Scripts
Selection Functions: ■ Select By Properties ■ Select By Layers ■ Select All ■ Deselect All	Not available	Available
Selection Functions: ■ Expand Selection ■ Banding Selection options	Not available	Not available
Edit Menu Functions: ■ Study Notes ■ Copy Image to Clipboard	Not available	Not available
View Menu Functions: ■ Toolbars ■ Project ■ Notes ■ Layers ■ All Panels ■ Model Display on/off ■ Default Display ■ Lock/Unlock Views/Animations/Plots	Not available	Not available
Modeling Menu Functions: ■ Create Inserts ■ Query Entities ■ Surface Boundary Diagnostics ■ Surface Connectivity Diagnostics ■ Surface Repair Tools ■ Simplify elements to beam	Not available	Not available
Mesh Menu Functions: ■ Mesh Repair Wizard ■ Show Diagnostics on/off	Not available	Not available
Analysis Menu Functions:	Not available	Not available

Feature	Macros	VB Scripts
■ Import Data From MPX		
Report Menu Functions:	Not available	Not available
■ All entries		
Tools Menu Functions:	Not available	Not available
■ New Personal Database ■ Edit Personal Database ■ Search Databases ■ Edit Default Properties Database ■ Import Legacy Moldflow/C-Mold Materials ■ Assign Macro/Command Button ■ Criteria Editor ■ Workspace		
Windows Menu Functions:	Not available	Not available
■ New Window ■ Cascade, Tile, Split ■ Arrange Icons		
Help Menu Functions:	Not available	Not available
■ All entries		
Project Panel-right click menu:	Not available	Not available
■ Compare Studies ■ New Report		
Study Tasks Pane-right click menu:	Not available	Not available
■ Hide Report Images		
Standard Toolbar:	Not available	Not available
■ Search Help ■ What's This		
Select Toolbar:	Not available	Not available

Feature	Macros	VB Scripts
■ All items except Select by properties		
Viewer Toolbar: ■ Banding Zoom ■ Move and Edit Cutting Plane ■ Add XY curve ■ Examine results ■ Split windows	Not available	Not available
Analysis Toolbar:	Not available	Available
Results Toolbar: ■ Plot Notes ■ New Report ■ Comparison Criteria Editor	Not available	Not available
Animation Toolbar: ■ All items	Not available	Not available
Modeling Toolbar: ■ Create Mold Insert ■ Runner System Wizard ■ Cooling Circuit Wizard ■ Mold Surface Wizard ■ Locate/Edit/Delete Surface Ties	Not available	Not available
Mesh Manipulation Toolbar: ■ Define Local Mesh Densities ■ Mesh Repair Wizard ■ Fix Aspect Ratios ■ All Diagnostics	Not available	Not available
Diagnostic Navigator Toolbar: ■ All items	Not available	Not available
Report Toolbar: ■ All items	Not available	Not available

Feature	Macros	VB Scripts
Macro Toolbar:	Not available	Not available
User Macro/Command Buttons Toolbar:	Not available	Not available
Scaling Toolbar:	Not available	Available

Autodesk Moldflow Insight command line and VB scripts

2

The Autodesk Moldflow Insight command line enables you to invoke Visual Basic (VB) scripts and macros from a command line user interface within Autodesk Moldflow Insight.

This capability, which is dynamic and extensible, allows you to run scripts that require you to provide parameters, and you can perform the following tasks:

- Modify existing scripts to better suit your needs, or extend them to add new capabilities
- Add new scripts containing multiple commands, which can be used just like the built-in commands that come with the Autodesk Moldflow Insight software.

A list of commands is displayed when you type **help** at the command line.

Scripts or commands in Autodesk Moldflow Insight can take multiple command line arguments to provide parameters to be used when the script is running.

The name of a script is the prefix of the corresponding script file; for example, the command **HCP** corresponds to the Visual Basic script file `HCP.vbs` in the commands folder, which prints the active display window.

You can create new scripts using one of the following techniques:

- Record a macro and save it in a commands folder to create a script that takes no command line arguments. You can then modify or extend the recorded macro to provide new capabilities or take command line arguments.

  NOTE: Macros that have been modified to take command line arguments can be launched only from the command line unless the script prompts for user input.

- Write scripts from scratch, which may or may not need parameters entered as command line arguments.

Autodesk Moldflow Insight searches for scripts first in the default project folder (where **xxxx** is the software release):

- On Windows XP systems, typically **My Documents\My AMI xxxx Projects \commands**
- On Windows Vista systems, typically**Documents\My AMI xxxx Projects \commands**

If Autodesk Moldflow Insight cannot find the script in the default project folder, it searches the Autodesk Moldflow Insight commands folder, which is typically

C:\Program Files\Autodesk\Moldflow Insight xxxx \data\commands (where **xxxx** is the software release).

NOTE: Not all functionality in Autodesk Moldflow Insight is accessible through the API. More functionality is accessible to scripts that are written by hand and run from the command line than to macros that are recorded and run from within the user interface.

Autodesk Moldflow Insight command line and VB scripts

The Autodesk Moldflow Insight command line allows you to invoke Visual Basic (VB) scripts and macros from a command line user interface within Autodesk Moldflow Insight.

Creating a Visual Basic script

Autodesk Moldflow Insight's command line interface provides access to Visual Basic scripts (VBScripts) from within the user interface.

User created scripts must be stored in the folder in Windows XP, **My Documents\My AMI xxxx Projects\commands**, or in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release).

You can use any text editor to create scripts. Some VBScript editors such as Microsoft Visual Basic Editor (VBE) or Adersoft VBSEdit, have built-in debuggers to highlight incorrect code.

You can write scripts that uses parameters which are typically entered by the user as command line arguments in the Autodesk Moldflow Insight command line. Alternatively, you can include a prompt for user input. This allows users to enter parameters if the script is run from within Autodesk Moldflow Insight.

NOTE: Not all functionality in Autodesk Moldflow Insight is accessible using the API. More functionality is accessible to scripts written by hand and run from the command line, than to macros that are recorded and run from within the interface.

---

1. Record a macro or type your script into a text editor or Visual Basic debugger.
2. Save the macro or script to a file with a *.vbs extension in Windows XP, **My Documents\My AMI xxxx Projects\commands**, or in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release).

The script can be played from the Command Line dialog **View tab > Windows panel > User Interface > Command Line**.

Debugging a Visual Basic script

The Microsoft Script Debugger can help you to locate and fix bugs in a Visual Basic script.

Download the debugger from Microsoft.

You can invoke the debugger in several ways.

- If the script crashes, the debugger starts at the line which caused the crash.

  To enable this behavior, turn off **Disable Script Debugging** in the Advanced tab of the Internet Properties item in the Control Panel.
- Insert the STOP statement in the script to launch the debugger when the statement is reached.

  Assistance for the STOP function can be found by searching the Microsoft support page. *http://support.microsoft.com*
- Invoke the script from the MS-DOS command line:

```
wscript //d //x scriptname.vbs
```

Playing macros or scripts using the menu

You can play a macro or a Visual Basic script in Autodesk Moldflow Insight using the menu, or using the Autodesk Moldflow Insight command line.

NOTE: You must use the Autodesk Moldflow Insight command line if the script requires command line arguments unless the script prompts for user input.

1 Select **Tools tab > Automation panel > Play Macro** .
The **Open Macro** dialog appears.

By default, macros are located in Windows XP, **My Documents\My AMI xxxx Projects\scripts**, or in Windows Vista, **Documents\My AMI xxxx Projects\scripts** , (where **xxxx** is the software release).

By default, command line scripts are located in Windows XP, **My Documents\My AMI xxxx Projects\commands**, and in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release) and the **data\commands** folder of the Autodesk Moldflow Insight installation directory.

2 Select a macro/script, then select **Open**.
The macro/script plays.

Running a macro or script from Windows Explorer

Because macros are just VBScripts, they can be invoked from Windows even if Autodesk Moldflow Insight is not running.

Scripts which assume that a particular study is already open, or that a particular result is already displayed, will probably not work.

NOTE: If your script uses parameters that are entered as command line arguments, you must run the script from the Autodesk Moldflow Insight command line.

---

1. Locate the macro or script using Windows Explorer.

   By default, macros are located in Windows XP, **My Documents\My AMI xxxx Projects\scripts**, or in Windows Vista, **Documents\My AMI xxxx Projects\scripts** (where **xxxx** is the software release).

   Command-line scripts are located in Windows XP, **My Documents\My AMI xxxx Projects\commands**, or in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release) and in the **data\commands** folder of the Autodesk Moldflow Insight installation directory.
2. Double-clicking on the required script.
   Autodesk Moldflow Insight will start if it is not currently running and the script will open and run.

Running a Visual Basic script using the command line

Autodesk Moldflow Insight's command line interface provides access to VBScript scripts from within the user interface.

The command line searches for scripts in the folder in Windows XP, **My Documents\My AMI xxxx Projects\commands**, or in Windows Vista, **Documents\My AMI xxxx Projects\commands**, (where **xxxx** is the software release), and in the **data\commands** folder of the Autodesk Moldflow Insight installation directory.

1. Select **View tab > Windows panel > User Interface > Command Line**.
   The **Command Line** dialog appears.
2. Enter the name of the script without its file extension followed by any command line arguments the script requires.

   For example, you may have a script named `test_script.vbs` which uses two numerical parameters (5 and 4 in this instance) that need to be input as command line parameters. Enter **test_script 5 4** at the command line.
3. Select **Go**.

The script plays.

Assigning macros or Visual Basic scripts to toolbar buttons

Autodesk Moldflow Insight allows you to assign a macro or a Visual Basic script to a button on the **Tools tab > Assigned Macros panel**. This panel provides quick access to up to 10 different scripts.

NOTE: Scripts which use parameters which are input as command line arguments must be run from the Autodesk Moldflow Insight command line.

---

1. Click **Tools tab > Assigned Macros panel > Assign Macro**.
2. Select the button, 1 to 10, to which you want to assign a macro or Visual Basic script.
3. Specify the macro or script to be assigned:
   - To assign a macro, select **Macro**, click ... at the right of the text box, then navigate to and select the ***.vbs** file you want.
   - To assign a Visual Basic script, select **Command Line** and enter the Visual Basic script name in the box provided.
4. In the **Button tip** text box, enter a description for this macro/command button. This description will be displayed as a tooltip on the button.
5. Select **Save** to store the button assignment.
6. You can assign additional buttons by repeating steps 2 to 5 above.
7. Select **Close** to close the dialog.
8. Click on one of the numbered toolbar buttons to run the associated macro or script.

Autodesk Moldflow Insight command line and VB scripts

Use the Command Line dialog to run a command or script.

Command Line dialog

This dialog is used to run a command or script file based on the Application Programming Interface (API) capabilities of Autodesk Moldflow Insight.

To access this dialog, click (**View tab > Windows panel > User Interface**), then click the drop-down arrow and select **Command Line**.

When you enter a command on the command line, the program looks for a file of the same name with the extension ".vbs" in the following folders:

- The folder *data\commands* where Autodesk Moldflow Insight has been installed. This folder contains the standard command scripts supplied by Autodesk.
- A *commands* folder under your default project folder (the User Folder that was set when the program was installed).

TIP: To view a list of the standard commands supplied with Autodesk Moldflow Insight, enter **help** on the command line.

VBScript references

More information about VBScript is available from these references.

Books

- *Learning VBScript*, Paul Lomax, O'Reilly & Associates, 1997
- *VBScript in a Nutshell* (2nd edition). Lomax, Childs, Petrusha. O'Reilly & Associates, 2003
- VBScript for Dummies, John Walkenbach. IDG Books, 1996.

Websites

- *VBScript at MSDN*
- *VBScript entry at Wikipedia*
- *Visual Basic Getting Started*, Wikibook

Editors and debuggers

- Microsoft Visual Basic Editor (VBE) is part of the Visual Studio suite
- *Adersoft VBSEdit*
- Microsoft Script Debugger is available at *www.microsoft.com*

Examples 3

The following are examples of API scripts that allow you to automate Autodesk Moldflow Insight.

API example: The first lines of a script

The following template is a suggested starting point for using the Synergy API with VBScript bindings.

It is good practice to put explanatory comments at the top of your scripts.

The following five lines and their function is outlined.

```
Option Explicit
```

The Option Explicit expression is useful to reduce programming errors. Variables that are used before they are declared will cause an error when this line is included in the script.

```
SetLocale("en-us")
```

The SetLocale option forces the non-English systems to interpret numerical values as they are in the US. If this setting is not included, then numerical values will be interpreted in the system's native language. This is a problem where commas are used instead of full stops (such as in Germany).

```
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
```

These two lines create an OLE automation object which starts the version of Synergy (the Autodesk Moldflow user interface) that was most recently executed.

NOTE: only one version of Synergy is able to be run at any time.

```
Synergy.SetUnits "METRIC"
```

“ENGLISH” can be used as an alternative to “METRIC” to use US units by default.

```
'@
'@ DESCRIPTION
'@
'@
'@ SYNTAX
'@ TheFirstLines
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@ none
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"
'
' Put remainder of code here.
'
MsgBox "Script Complete"
Wscript.Quit
```

API example: Customized aspect ratio plot

This example creates a custom contour plot using aspect ratio data from mesh diagnostics. The script uses the DiagnosisManager class for access to mesh diagnostics data, and the PlotManager class to create the custom user plot.

```
'@
'@ DESCRIPTION
'@ Take the Standard Aspect Ratio Plot and convert it into a contour plot
'@
'@ SYNTAX
'@ CustomAspect
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@ Assumes a study file is open within synergy
'@ none
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"


Dim DiagnosisManager, PlotManager
Dim Elems, AR, ARPlot


' Get aspect ratio diagnostics
Set Elems = Synergy.CreateIntegerArray()
Set AR = Synergy.CreateDoubleArray()
Set DiagnosisManager = Synergy.DiagnosisManager()
DiagnosisManager.GetAspectRatioDiagnosis 0.0, 0.0, True, Elems, AR
Set DiagnosisManager = Nothing
```

```
' Create user plot
Set PlotManager = Synergy.PlotManager()
Set ARPlot = PlotManager.CreateUserPlot()
ARPlot.SetDataType "ELDT"
ARPlot.SetName "Aspect ratio by contours"
ARPlot.AddScalarData 0.0, Elems, AR
ARPlot.Build

MsgBox "Script Complete"
Wscript.Quit
```

API example: Showing thicknesses within a range

You can use any standard function to interact with the user, such as VBScript's InputBox() function. This example displays the Thickness diagnostic with thicknesses between two user-specified values.

This script can be run as either a command or a macro. If parameters were not supplied on the command line when the script was run, you will be prompted to input parameters. The bold section below implements the user input prompts after checking for two command line argument values.

```
'@
'@ DESCRIPTION
'@
'@
'@ SYNTAX
'@ ShowThicknessInRange [Min] [Max]
'@
'@ PARAMETERS
'@ Min   Minimum Thickness value
'@ Max   Maximum Thickness value
'@
'@ DEPENDENCIES/LIMITATIONS
'@ none
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim DiagnosisManager
Dim MinimumThickness, MaximumThickness
Dim Args

Set Args = Wscript.Arguments
If Args.Count <> 2 Then
        MinimumThickness = InputBox("Enter Minimum Thickness")
        MaximumThickness = InputBox("Enter Maximum Thickness")
Else
        MinimumThickness = Args(0)
        MaximumThickness = Args(1)
End If

Set DiagnosisManager = Synergy.DiagnosisManager()
DiagnosisManager.ShowThickness MinimumThickness, MaximumThickness, False
```

```
MsgBox "Script Complete"
WScript.Quit
```

API example: Reading pressure data

Time-series data are available through the API as the independent values of a plot. This is how animations are represented.

This example performs the following tasks:

- Extracts the pressure result and computes the average pressure of each set of five time steps
- Reconstructs a custom pressure plot

You can use this example when you want to read a specific result set, or put the data into a new result set.

```
'@
'@ DESCRIPTION
'@ Split Presssure Time Series Result into Individual Results.
'@ Recreate the Pressure Time Series Result
'@
'@ SYNTAX
'@ ReadingPressureData
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@ none
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim PlotManager
Dim ResultID, Header, UserPlot, Indp, nodeID, Value, IndpValues, i, j
,Ave, Count, Str

ResultID = 1180

' Split Presssure Result into Individual Results.
Set PlotManager = Synergy.PlotManager()
Set IndpValues = Synergy.CreateDoubleArray()
PlotManager.GetIndpValues ResultID, IndpValues

' Display to Screen Average Pressure at interval through results
For i = 0 to IndpValues.size()-1 Step 5
        Set PlotManager = Synergy.PlotManager()
        Set Indp = Synergy.CreateDoubleArray()
        Indp.AddDouble(IndpValues.val(i))
        Set nodeID  = Synergy.CreateIntegerArray()
        Set Value = Synergy.CreateDoubleArray()
        PlotManager.GetScalarData ResultID, Indp, nodeID, Value
        Ave = 0.0
        Count = 0.0
        For j = 0 To nodeID.size - 1
                Count = Count +1
                Ave = Ave + Value.val(j)
        Next
        If Count > 0 Then
                Ave = Ave / cdbl(count)
        End If
    Str = Str + "Time=" & CStr(IndpValues.val(i)) & " Average P =" & Ave
 &" MPa" & vbCRLF
Next
MsgBox Str

'Recreate current Time Series Pressure Result
Set UserPlot = PlotManager.CreateUserPlot()
Header = "New Pressure Time Series"
PlotManager.DeletePlotByName(Header)
UserPlot.SetName(Header)
UserPlot.SetDataType("NDDT")
UserPlot.SetDeptUnitName("MPa")
UserPlot.SetDeptName("Time")
UserPlot.SetVectorAsDisplacement(False)
For i = 0 to IndpValues.size-1
  Set Indp = Synergy.CreateDoubleArray()
  Indp.AddDouble(IndpValues.Val(i))
  Set nodeID  = Synergy.CreateIntegerArray()
  Set Value = Synergy.CreateDoubleArray()
  PlotManager.GetScalarData ResultID, Indp, nodeID, Value
  UserPlot.AddScalarData IndpValues.Val(i), nodeID, Value
Next
UserPlot.Build()


MsgBox "Script Complete"
Wscript.Quit
```

API example: Looping through entities

Entities in studies are available though linked lists, one for each entity type, such as node, triangle, tetrahedron (tetra), and so forth. This example implements several loops for looking at each entity of each type.

This script performs the following tasks:

- Finds the maximum node number
- Counts the number of beams
- Counts the number of tetras
- Counts the number of triangles

You can use this script when you want to read nodal data.

```
'@
'@ DESCRIPTION
'@ Example of how to loop through entities using the API
'@
'@ SYNTAX
'@ LoopThroughEntities
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@ none
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim StudyDoc
Dim MaxNumber, Node, NodeNumber, Count, Tri, Tet, Beam

Set StudyDoc = Synergy.StudyDoc

' Loop through all nodes and find the highest Node Number

MaxNumber = 0
Set Node = StudyDoc.GetFirstNode()
While Not Node Is Nothing
        NodeNumber = StudyDoc.GetEntityID(Node)
        If NodeNumber > MaxNumber Then
                MaxNumber = NodeNumber
        End if
        Set Node = StudyDoc.GetNextNode(Node)
Wend
MsgBox "Maximum Node Number in Model is: " & CStr(MaxNumber)

' Count all Triangular Elements in the Model
Count = 0
Set Tri = StudyDoc.GetFirstTri()
While Not Tri Is Nothing
        Count = Count + 1
        Set Tri = StudyDoc.GetNextTri(Tri)
Wend
MsgBox "Model Contains " & CStr(Count) & " Triangular Elements"

' Count all Tet Elements in the Model
Count = 0
Set Tet = StudyDoc.GetFirstTet()
While Not Tet Is Nothing
        Count = Count + 1
        Set Tet = StudyDoc.GetNextTet(Tet)
Wend
MsgBox "Model Contains " & CStr(Count) & " Tet Elements"

' Count all Beam Elements in the Model
Count = 0
Set Beam = StudyDoc.GetFirstBeam()
While Not Beam Is Nothing
        Count = Count + 1
        Set Beam = StudyDoc.GetNextBeam(Beam)
Wend
MsgBox "Model Contains " & CStr(Count) & " Beam Elements"


MsgBox "Script Complete"
Wscript.Quit
```

API example: The minimum, maximum, average of an entity list

When you select a set of entities, the selection is made available through the Application Programming Interface (API) as an entity list. This list is unsorted, like the list of elemental results in a result.

The following example finds the minimum, maximum and average values of the result that is currently displayed, for the selected entities.

NOTE: For large models or large selections, it may be beneficial to sort the lists first to reduce the time spent matching the selected entities to results.

Before running this script, you must complete the following steps:

1 Select entities in the model.
2 Select a plot.

In the following example, some error checking has been implemented. The script checks that entities have been selected, and that there is a valid, active plot.

```
'@
'@ DESCRIPTION
'@ This command will calculate the result Minimum Maximum and Average 
Value
'@ of the entities selected
'@
'@ SYNTAX
'@ MinimumMaximumAverage
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@  1. User must select a group of entities before using this command
'@  2. User must select a result before using this command
'@  3. Only works with nodal/elemental plots  ie no vector/tensor plots
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("Synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim StudyDoc, Viewer, Plot, PlotManager
Dim EntList, ResultID, ResultType, ResultName, ResultData
Dim IndpValues, IndpValues1
Dim EntityIndex, Result, ElementNumber, NodeNumber
Dim Min, Max, Ave, Count, Value, I, J, Ent, Name

' Check that the user has selected some Entities
Set StudyDoc = Synergy.StudyDoc
Set EntList = StudyDoc.Selection
If EntList.Size <= 0 Then
   MsgBox "No Entities Selected",,"Error"
   WScript.Quit
End If

' Read/Check the Plot Information is correct
' Get Plot
Set Viewer = Synergy.Viewer
Set Plot = Viewer.ActivePlot()
If Plot Is Nothing Then
   MsgBox "Please Select a result",,"Error"
   WScript.Quit
End If

' Read Result ID Type
```

```
ResultID=Plot.GetDataID

' Read the Result Name
ResultName = Plot.GetName

' Check for Invalid Plot Data
ResultData = Plot.GetDataType
If ResultData <> "ELDT" and ResultData <> "NDDT"  Then
   MsgBox "Data Type Not Supported",,"Error"
   WScript.Quit
End If

' Check for Invalid Plot Types
ResultType = Plot.GetPlotType
If ResultType <> "Contour Plot"  Then
   MsgBox "Plot Type Not Supported",,"Error"
   WScript.Quit
End If

' Read the Result Data set
' Ensure we read the last data set
Set PlotManager = Synergy.PlotManager
Set IndpValues = Synergy.CreateDoubleArray()
PlotManager.GetIndpValues ResultID, IndpValues
Set IndpValues1 = Synergy.CreateDoubleArray()
IndpValues1.AddDouble(IndpValues.Val(IndpValues.Size()-1))
' Read Result Data from last set

Set EntityIndex = Synergy.CreateIntegerArray()
Set Result = Synergy.CreateDoubleArray()
PlotManager.GetScalarData ResultID, IndpValues1, EntityIndex, Result


' Calcuate the required Values
Min = 1.0E20   ' Set extremely Large Value
Max = -1.0E20  ' Set extremely Small Value
Ave = 0.0
Count= 0

' If Result in Elemental then loop through selected elements
If ResultData = "ELDT" Then
  For I = 0 To EntList.Size()-1
    Set Ent = EntList.Entity(I)
    Name = Ent.ConvertToString
    If (Left(Name,1) = "T" and Left(Name,2) <> "TE" ) Then
          ElementNumber = StudyDoc.GetEntityID(Ent)
      For J = 0 To EntityIndex.Size()-1
        If EntityIndex.Val(J) = ElementNumber Then
          Count = Count + 1
          Value = Result.Val(J)
          If (Value > Max) Then
            Max = Value
          End If
          If (Value < Min) Then
            Min = Value
          End If
          Ave = Ave + Value
        End If
      Next
    End If
  Next
' If Result in Nodal then loop through selected  nodes
ElseIf ResultData = "NDDT" Then
  For I = 0 To EntList.Size()-1
    Set Ent = EntList.Entity(I)
    Name = Ent.ConvertToString
    If (Left(Name,1) = "N" ) Then
          NodeNumber = StudyDoc.GetEntityID(Ent)
      For J = 0 To EntityIndex.Size()-1
        If EntityIndex.Val(J) = NodeNumber Then
```

```
          Count = Count + 1
          Value = Result.Val(J)
          If (Value > Max) Then
            Max = Value
          End If
          If (Value < Min) Then
            Min = Value
          End If
          Ave = Ave + Value
        End If
      Next
    End If
  Next
End If

' Display Results
If Count > 0 Then
        Ave = Ave / CDbl(Count)
  MsgBox "Analysing result :" & ResultName & vbcrLf & _
         "Number of Selected Entities is: " & Cstr(Count) & vbcrLF & _
         "Minimum is: " & Cstr(Min)  & vbcrLF & _
         "Maximum is: " & Cstr(Max)  & vbcrLF & _
         "Average is: " & Cstr(Ave)
Else
  MsgBox "Analysing result :" & ResultName & vbcrLf & _
         "Number of Selected Entities is: " & Cstr(Count) & vbcrLF
End If


MsgBox "Script Completed"
WScript.Quit
```

API example: Writing nodal data to a file

The standard file facilities provided in the programming language you are using (File.Write for VBScript) are used to write to external files and launch external programs.

This example creates a new comma delimited file (***.csv**), and writes the following information to the file for each node in the model:

Node number, x coordinate, y coordinate, z coordinate

When the file has been written, an external application (Notepad) is launched to display the file.

```
'@
'@ DESCRIPTION
'@ Extract all Nodal Coordinates to a comma separated text file
'@
'@
'@ SYNTAX
'@ WriteNodalData
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@
'@
'@ History
'@ Created DRA 9/8/2006
'@@
```

```
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim StudyDoc, App
Dim FS, TemporaryFolder, TempFolder, Name, lFile
Dim Str, Node, NodeNumber, Coord

Set StudyDoc = Synergy.StudyDoc()

'Open a File in the users temporary Directory
Set FS = CreateObject("Scripting.FileSystemObject")
TemporaryFolder = 2
Set TempFolder = FS.GetSpecialFolder(TemporaryFolder)
Name = "data.txt"
Set lFile = TempFolder.CreateTextFile(Name, True)

' Write File Header
Str = "Node" & "," & "X" & "," & "Y" & "," & "Z" &  vbCrLf
lFile.Write Str

' Loop through all nodal
Set Node = StudyDoc.GetFirstNode()
While Not Node Is Nothing
        NodeNumber = StudyDoc.GetEntityID(Node)
        Set Coord = StudyDoc.GetNodeCoord(Node)
       Str = NodeNumber & "," & Coord.X & "," & Coord.Y  & "," & Coord.Z
 & vbCRLF
        lFile.Write Str
        Set Node = StudyDoc.GetNextNode(Node)
Wend

' Close File
lFile.Close

' Notify user where the file is located
MsgBox "Nodal Data Recorded In File" & vbCRLF & TempFolder.Path & "\" &
 Name

' Open the File in Notepad
Set App = WScript.CreateObject("WScript.Shell")
Dim Command
Command = "notepad.exe " & TempFolder.Path & "\" & Name
App.Run Command

MsgBox "Script Complete"
Wscript.Quit
```

API example: Creating multiple drops

New entities can be created through the Application Programming Interface (API). This example creates entities for a hot runner system gated to selected nodes, and then meshes the new entities. The numbers used in properties (five-digit constants in the example) can be found in the **tcodes.dat** and **tcodeset.dat** files in the **data/dat** directory of the installed Autodesk Moldflow Insight software.

NOTE: The tcodeset reference can help you understand how to use properties in scripts.

```
'@
'@ DESCRIPTION
'@ This command will run all Models in the current directory
'@
'@
'@ SYNTAX
'@ CreateDrops
'@
'@ PARAMETERS
'@ none
'@
'@ DEPENDENCIES/LIMITATIONS
'@
'@ History
'@ Created DRA 9/8/2006
'@@
Option Explicit
SetLocale("en-us")
Dim Synergy
Set Synergy = CreateObject("synergy.Synergy")
Synergy.SetUnits "METRIC"

Dim  StudyDoc, PropEd, Modeler, MeshGenerator, Viewer
Dim SelectList, Prop, DVec, DVec1, HotGateCount, HotRunCount
Dim I, nodeCoord, BaseX, BaseY, BaseZ, Vector, Vector_1, EntList

Set StudyDoc = Synergy.StudyDoc()
If StudyDoc is Nothing Then
        MsgBox "No Active Study",,"Error"
        WScript.Quit
End If

Set SelectList = StudyDoc.Selection
If SelectList.Size = 0 Then
        MsgBox "No Entities have been selected",,"Error"
        WScript.Quit
End If

Set PropEd = Synergy.PropertyEditor()
Set Modeler = Synergy.Modeler()

' Create Properties for Gate
HotGateCount = GetLastTsetID(40434) + 1
Set Prop = PropEd.CreateProperty(40434, HotGateCount, True)
Set DVec = Synergy.CreateDoubleArray()
DVec.AddDouble 2
Prop.FieldValues 30212, DVec
Set DVec1 = Synergy.CreateDoubleArray()
DVec1.AddDouble 1.5
DVec1.AddDouble 5
Prop.FieldValues 30262, DVec1
PropEd.CommitChanges ""

' Create Properties for Drop
HotRunCount = GetLastTsetID(40430) + 1
Set Prop = PropEd.CreateProperty(40430, HotRunCount, True)
Set DVec = Synergy.CreateDoubleArray()
DVec.AddDouble 10
Prop.FieldValues 30260, DVec
PropEd.CommitChanges ""

' Loop Through Slected Nodes and create Gates and Drops
For I = 0 To SelectList.Size()-1
        Dim Ent
        Set Ent = SelectList.Entity(I)
        Dim EntName
        EntName = Ent.ConvertToString
        If ( Left(EntName, 1) = "N") Then
                Set  nodeCoord = StudyDoc.GetNodeCoord(Ent)
                BaseX = nodeCoord.X
                BaseY = nodeCoord.Y
                BaseZ = nodeCoord.Z
                Set Vector = Synergy.CreateVector()
                Set Vector_1 = Synergy.CreateVector()
                Set Modeler = Synergy.Modeler()
                ' Create Gate
                Set Prop = Modeler.FindProperty(40434, HotGateCount)
                Vector.SetXYZ BaseX, BaseY, BaseZ
                Vector_1.SetXYZ 0, 0, 5
                Set EntList = Modeler.CreateLine(Vector, Vector_1, True,
 Prop, True)
                PropEd.SetProperty EntList, Prop
                ' Create Drop
                Set Prop = Modeler.FindProperty(40430, HotRunCount)
                Vector.SetXYZ BaseX, BaseY, BaseZ+5
                Vector_1.SetXYZ 0, 0, 100
                Set EntList = Modeler.CreateLine(Vector, Vector_1, True,
 Prop, True)
                PropEd.SetProperty EntList, Prop
        End If
Next

' Save All Proerty Data
PropEd.CommitChanges "Assign"

' Mesh the new drops
Set MeshGenerator = Synergy.MeshGenerator()
MeshGenerator.Generate

' Display Updated Model
Set Viewer = Synergy.Viewer()
Viewer.Fit

' Exist Script
MsgBox "Script Complete"
WScript.Quit

' Function to Find Last Instance of a Tset in the Study File
Private Function GetLastTsetID(Tset)

        GetLastTsetID = 0
        Dim Project, PropEd
        Set Project = Synergy.Project()
        Set PropEd = Synergy.PropertyEditor()
        Dim TestPropertyID
        Set TestPropertyID = PropEd.GetFirstProperty(Tset)
        While Not TestPropertyID Is Nothing
                GetLastTsetID = TestPropertyID.ID
                if TestPropertyID.ID > GetLastTsetID Then
                        GetLastTsetID = TestPropertyID.ID
                End if
                Set TestPropertyID =
PropEd.GetNextPropertyOfType(TestPropertyID)
        Wend

End Function
```