VB Script Reference

Summary

This reference manual describes the VB Script language used in Altium Designer.

Technical Reference
TR0125 (v1.4) Sept 1, 2006

This reference covers the following topics:

- Exploring the VB Script Language
- VB Script source files
- Creating new scripts and adding scripts to a project
- Executing a script in Altium Designer
- Assigning a script to a process launcher
- About VB Script examples
- Writing VB Script scripts
- VB Script keywords, statements and functions
- Forms and Components.

Exploring the VB Script Language

This Reference details each of the VisualBasic Scripting statements, functions and extensions that are supported in the scripting system. The Visual Basic Scripting or VB Script for short also can deal with Altium Designer Object Models and Visual Components. It is assumed that you are familiar with basic programming concepts and as well as the basic operation of your Altium Designer-based software.

The scripting system supports the VB Script language (along with other scripting languages) which is derived from the Microsoft ActiveX Scripting system, for instance you should be able to use CScripts or WScripts which are based on the same ActiveX scripting engine that Altium Designer uses.

All scripting languages supported in Altium Designer are typeless or untyped which means you cannot define records or classes and pass pointers as parameters to functions for example.

VB Script script example

```
Sub DisplayName (sName)
    MsgBox "My Name is " & sName
End Sub
```

For detailed information on VB Script and its keywords, operators and statements, please refer to Microsoft Developers Network website, **http://msdn.microsoft.com/library/en-us/script56/html/vtoriVB Script.asp**.

Altium Designer and Borland Delphi Run Time Libraries

The Scripting system also supports a subset of Borland Delphi Run Time Library (RTL) and a subset of Altium Designer RTL which is covered in the RTL Reference in the Scripting Online Help in Altium Designer.

There are several Object Models in Altium Designer; for example you can use the PCB Object Model in your VB Scripts to deal with PCB objects on a PCB document, WorkSpaceManager Object Model to work with Projects and their documents and extract netlist data for example.

The Scripting Online Reference Help contains information on interfaces with respect to Altium Designer Object Models, components, global routines, types, and variables that make up this scripting language. You can consult the Visual Basic documentation by Microsoft for more information on VB Script functions.

Server Processes

A script can execute server processes and thus server processes and parameters are covered in the Server Process Reference.

VB Script source files

You open a script project in Altium Designer and you can edit the contents of a script inside the Altium Designer. A script project is organized to store script documents (script units and script forms). You can execute the script from a menu item, toolbar button or from the *Run Script* dialog from the Altium Designer's system menu.

PRJSCR, VBS and DFM files

The scripts are organized into projects with a PRJSCR extension. Each project consists of files with a vbs extension. Files can be either script units or script forms (each form has a script file with vbs extension and a corresponding form with a dfm extension). A script form is a graphical window that hosts different controls that run on top of Altium Designer.

However it is possible to attach scripts to different projects and it is highly recommended to organize scripts into different projects to manage the number of scripts and their procedures / functions.

Scripts (script units and script forms) consist of functions/procedures that you can call within Altium Designer.

Creating new scripts

You can add existing or new scripts into the specified project in the **Projects** panel in Altium Designer. There are two types of scripts : Script Units and Script Forms. With a project open in Altium Designer, right click on a project in the **Projects** panel, and a pop up menu appears, click on the **Add New to Project** item, and choose **VB Script Unit.** A new script appears.

A script can have at least one routine which defines the main program code. You can, however, define other routines and functions that can be called by your code. The functions and subroutines are defined within a **Function End Function** or **Sub End Sub** statement block.

Note that it is possible to have no routines within a script but at least it is necessary to have a .

Example of a Subroutine

```
Sub Log(Description, Data)
    Call ReportFile.Write(Description)
    Call ReportFile.WriteLine(Data)
End Sub
```

Example of a subroutine less script.

```
' script here with no function/sub routine
A = 50
A = A + 1
ShowMessage(IntToStr(A))
```

Adding scripts to a project

You can add existing scripts to a specified project in the **Projects** panel in Altium Designer. With a project open in Altium Designer, right click on this project in the **Projects** panel, and a pop up menu appears, click on the **Add Existing to Project...** item.

A *Choose Documents To Add to Project* dialog appears. You can multi-select as many scripts you want to add into the specified project.

Executing a script in Altium Designer

In Text Editor workspace

You can configure the **Run** command when you are in the text editor to point to a script and execute it. Every time you click on the **Run** icon from the Text Editor menu or press **F5**, the scripting system executes the script pointed to by the **Set Project Startup Procedure** item. You can change the start up procedure by clicking on the **Set Project Start Up Procedure** item in the **Run** menu which invokes the *Select Item to Run* dialog. You can then select which procedure of a script to be set.

Executing a script on a design document

To execute a script in Altium Designer, there are two methods and there are two different Altium Designer dialogs for each method. These methods are necessary if you wish to run a script on a server specific document such as PCB or Schematic documents.

1. Using the Select Item To Run dialog to execute a script

Click on the **Run Script** from the Altium Designer system menu and the *Select Item to Run* dialog appears with a list of procedures (those parameter-less procedures/functions only appear) within each script in a opened project in Altium Designer.

Note that you can also click on a script unit filename within this *Select Item to Run* dialog and the functionless/procedureless **Begin End.** block within the script gets executed. See code example here

Script unit

```
' script here with no function/procedure
A = 50
A = A + 1
```

```
ShowMessage(IntToStr(A))
```

Now, only parameter-less functions and procedures for each script of an opened project only appear on the *Select Item to Run* dialog. It is a good idea for script writers to write the functions in scripts so that they will appear in this dialog and the other functions with parameters not to appear in this same dialog.

When you are working in a different editor such as PCB editor, you can assign the script to a process launcher and use it to run a specified script easily. See the Assigning a script to a process launcher.

You can add a list of installed script projects so that, every time you invoke the *Select item to Run* dialog, the installed script projects will appear along with other script projects currently open in the **Projects** panel. Invoke **Scripting System Settings** item from **Tools » Editor Preferences** menu in the TextEditor workspace and then drill down to the Altium Designer System, Scripting System, and the *Scripting System page* appears.

2. Using the Run Process dialog to execute a script

Invoke the *Run Process* dialog from Altium Designer's System menu and execute the **ScriptingSystem:RunScript** process in the Process: field and specify the script parameters, the **ProjectName** parameter which is the path to the project name and the **ProcName** parameter to execute the specified procedure from a specified script in the Parameters: field.

You need the following parameters for the **ScriptingSystem:RunScript** process to execute a specified script.

Process:

ScriptingSystem:RunScript

Parameters:

ProjectName (string)

ProcName (string)

Example

Process: ScriptingSystem:RunScript

Parameters : ProjectName = C:\Program Files\Altium Designer 6\Examples\Scripts\VB Scripts\HelloWorld.PrjScr | ProcName = HelloWorld>HelloWorld.

To run a script repeatedly in the text editor in Altium Designer, assign the script to the **Set Project Startup Procedure** item from the **Run** menu of the Text editor server. you can then click on the Run button. or press F5 to execute this script. To run a different script, you will need to re-invoke the **Set Project Startup Procedure** from the Run menu and assign a new script to it.

You can click on the **Run Script** item from the Altium Designer system menu and the *Select Item to Run* dialog appears with a list of procedures (those parameterless procedures/functions only appear) within each script in a project. This may be needed if you wish to run a script on a specific document type such as PCB or Schematic documents.

You can also use the *Run Process* dialog and specify the scriptingsystem server process and specify the parameters for this scripting system server to execute a script, this may also be needed if you wish to run a script on a specific document type such as PCB or Schematic documents.

Assigning a script to a menu, key or toolbar

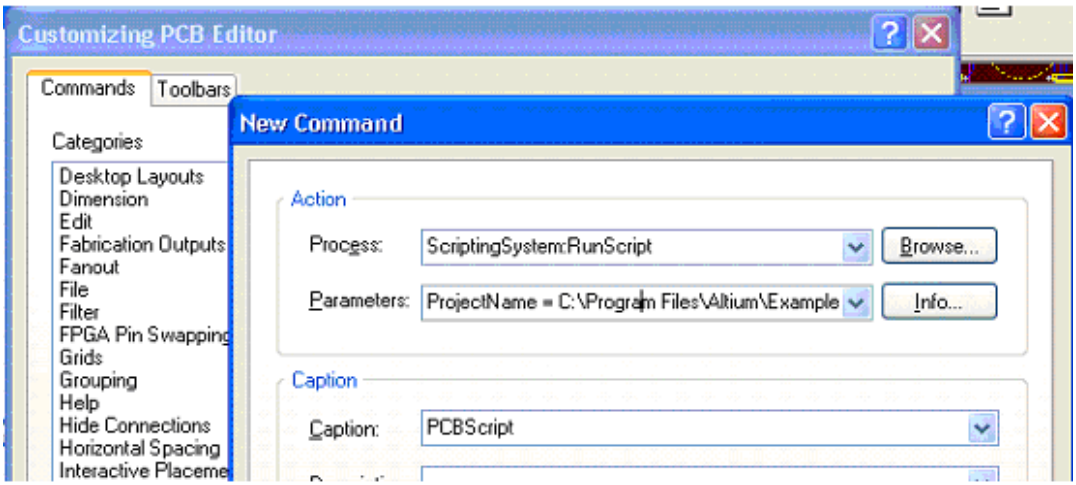
You have the ability to assign a script to a server menu, toolbar or hot key in Altium Designer which makes it possible for you to run the script over a current PCB document for example. You will need to specify the full path to a project where the script resides in and specify which unit and procedure to execute the script.

There are two parameters in this case: the **ProjectName** and the **ProcName**. For the **ProcName** parameter, you need to specify the script filename and the procedure. So the format is as follows: **ProcName = ScriptFileName>ProcedureName**. Note the GreaterThan (>) symbol used between the script file name and the procedure name.

Assigning to a process launcher example

To illustrate this ability to assign a script to a resource, we will open a PCB document in Altium Designer and use the HelloWorld script example from the **\Scripts\VB Scripts** folder.

1. Double click on the PCB menu and the *Customizing PCB Editor* dialog appears.
2. Click on the **New** button from the *Customizing PCB Editor* dialog.
3. Choose **ScriptingSystem:RunScript** process in the **Process**: field of the *Customizing PCB Editor* dialog.
4. Enter **ProjectName = C:\Program Files\Altium Designer 6\Examples\Scripts\VB Scripts\HelloWorld.PrjScr | ProcName = HelloWorld>ShowHelloWorldMessage** text in the Parameters: field for example.



5. You will need to give a name to this new command and assign a new icon if you wish. In this case, the name is **PCBScript** in the Caption: field of this dialog. The new commands appear in the **[Custom]** category of the **Categories** list. Click on the **[Custom]** entry from the **Categories** list. The **PCBScript** command appears in the **Commands** list of this dialog.
6. You then need to drag the new **PCBScript** command onto the PCB menu from the *Customizing PCB Editor* dialog. The command appears on the menu. You can then click on this new command and the HelloWorld dialog appears.

About VB Script examples

The examples that follow illustrate the basic features of VB Script programming in Altium Designer. The examples show simple scripts for the Altium Designer application. The VB Scripts can use script forms, script units, functions and objects from the Altium Designer Run Time Library and a subset of functions and objects from the Borland Delphi that is exposed in the scripting system.

The example scripts are organized into **\Examples\Scripts\VB Scripts**\ folder.

Writing VB Script scripts

In this section:

- VB Script Naming Conventions
- Local and Global variables
- Subroutines and Functions
- Splitting a line of script.

VB Script naming conventions

VB Script variables are case insensitive, that is, variables in upper and lower case have the same meaning:

Example

The variables b and B are the same.

```
b = 60
B = 60
```

Local and Global variables

Since all scripts have local and global variables, it is very important to have unique variable names in your scripts within a script project. If the variables are defined outside any subroutines and functions, they are global and can be accessed by any unit in the same project.

If variables are defined inside a routine, then these local variables are not accessible outside these routines. Since scripts are typeless you do not initialize variables with their types at all.

Variable Initialization

The local variables inside a procedure are automatically initialized.

```
Sub Example
    Dim X
    Dim s
    ' x set to 0
    x = 0
    ' s set to empty
    s = “”
End Sub
```

Subroutines and Functions

VB Script allows two kinds of procedures; subroutines and functions. A function returns a value only. The syntax of calling a subroutine or a function in a script is as follows:

Call SubRoutineA(parameters)

or

SubRoutine parameters

Subroutine example

```
Sub SetTheHeight AHeight
    Set Component.Height = AHeight
End Sub
```

Function example

```
Function Addone(value)
   AddOne = Value + 1
End Function
```

A longer example

```
Function Test(s)
    Test = S + " rules.."
End Function

Sub DisplayName (sName)
    MsgBox sName
End Sub

Sub Main
Dim S
    S = "Altium Designer"
    DisplayName Test(s)
End Sub
```

Parameters and Arguments

The procedure declaration normally has a list of parameters (remember variables are considered typeless and the scripting system works out automatically what the variable types are). The value used in place of the parameter when you make a procedure call is called an argument.

Example of a subroutine with a parameter.

```
Sub DisplayName (sName)
    MsgBox "My Name is " & sName
End Sub
```

Example of calling a subroutine

```
Sub Main
    DisplayName "Altium Designer Rules"
End Sub
```

Notes

The use of the **Call** keyword to invoke a subroutine or a function is optional (maintained for backward compatibility).

Including comments in scripts

In a script, comments are non-executed lines of code which are included for the benefit of the programmer. Comments can be included virtually anywhere in a script.

Any text following ' are ignored by VB Script.

Any text following Rem are ignored by VB Script.

' Comment type example

```
' This whole line is a comment
Rem this whole line is also a comment
DocName = Document.Name ' Get name of active document
```

Splitting a line of script

Each code statement is terminated on each line to indicate the end of this statement. VB Script allows you to write a statement on several lines of code, splitting a long instruction on two or more lines using the underscore character (_).

VB Script does not put any practical limit on the length of a single line of code in a script, however, for the sake of readability and ease of debugging it is good practice to limit the length of code lines so that they can easily be read on screen or in printed form.

Basically if a line of code is very long, you can break this line into multiple lines and this code will be treated by the VB interpreter as if it were written on a single line.

Unformatted code example

```
If Not (PcbApi_ChooseRectangleByCorners(BoardHandle,”Choose first
corner”,”Choose final corner”,x1,y1,x2,y2)) Then EndIf
```

Formatted code example

```
If Not (PcbApi_ChooseRectangleByCorners(BoardHandle,_
                                        “Choose first corner”,_
                                        “Choose final corner”,_
                                        x1,y1,x2,y2)) Then EndIf
```

Using Altium Designer Objects in scripts

The biggest feature of the scripting system, is that the Interfaces of Altium Designer objects are available to use in scripts. For example you have the ability to massage design objects on Schematic and PCB documents through the use of Schematic Interfaces and PCB interfaces.

Therefore the Altium Designer Interfaces are available for use on any script. Normally in scripts, there is no need to instantiate an interface, you just extract the interface representing an existing object in Altium Designer and from this interface you can extract embedded or aggregate interface objects and from them you can get or set property values.

Thus to have access to a schematic document, you invoke the SchServer

Example

```
    ' Checks if the current document is a Schematic document
    If SchServer Is Nothing Then Exit Sub
    Set CurrentSheet = SchServer.GetCurrentSchDocument
    If CurrentSheet Is Nothing Then Exit Sub
```

To have access to a PCB document, you invoke the PCBServer.

Creation of a PCB object using the PCB Object model

```
Sub ViaCreation
    Dim  Board
    Dim  Via

    Set Board = PCBServer.GetCurrentPCBBoard
    If Board is Nothing Then Exit Sub

    ' Create a Via object
    Via           = PCBServer.PCBObjectFactory(eViaObject, eNoDimension,
eCreate_Default)
    Via.X         = MilsToCoord(7500)
    Via.Y         = MilsToCoord(7500)
    Via.Size      = MilsToCoord(50)
    Via.HoleSize  = MilsToCoord(20)
    Via.LowLayer  = eTopLayer
    Via.HighLayer = eBottomLayer
    ' Put this via in the Board object
    Board.AddPCBObject(Via)
End Sub
```

Objects, Interfaces, functions and types in your scripts can be used from the following:

- Client API
- PCB Server API
- Schematic Server API
- Work Space Manager Server API
- Nexus API
- Altium Designer RTL functions
- Parametric processes.

VB Script Keywords

The scripting system supports the VB Script language which is derived from the Microsoft Active Scripting language technology.

Reserved words and functions in VB Script

A, B

Abs, Array, Asc, Atn

C

Call, Case, CBool, CByte, CCur, CDate, CDbl, Chr, CInt, Class, CLng, Const, Conversions, Cos, CreateObject, CSng, CStr

D,E

DateAdd, DateDiff, DatePart, DateSerial, DateValue, Day, Derived Math, Dim, Do, Each, Erase, Escape, Empty, Eval, Execute, Exit, Exp

F,G, H

False, Filter, For, FormatCurrency, FormatDateTime, FormatNumber, FormatPercent, Function GetLocale, GetObject, GetRef, Hex, Hour

I, L, M

If, Is, InputBox, Instr, InStrRev, Int, IsArray, IsDate, IsEmpty, IsNull, IsNumeric, IsObject, Join, LBound, LCase, Left, Len, LoadPicture, Log, LTrim, Maths, Mid, MInute, Month, MonthName, MsgBox

N, O

Next, Nothing, Now, Null, Oct, On Error

P,R

Private, Property, Public, Randomize, ReDim, Rem, RTrim, Replace, RGB, Right, Rnd, Round

S, T

ScriptEngine, ScriptEngineBuildVersion, ScriptEngineMajorVersion, ScriptEngineMinorVersion, Second, Select, Set, SetLocale, Sgn, Sin, Space, Split, Sqr, Stop, StrComp, String, StrReverse, Sub, Tan, Then, Time, Timer, Timeserial, TimeValue, Trim, True, TypeName

U, V, W, X, Y

UCase, Unescape, While, Wend, With, VarType, Weekday, WeekdayName, Year

VB Script Statements

In this section:

- Conditional statements
- Expressions and Operators

Conditional statements

The main conditional statements supported by the VB Script;

- If Then
- For Next Loop
- Exit For
- For Each Next
- Do Loop
- While WEnd
- Select Case

You have to be careful to code your scripts to avoid infinite loops, ie the conditions will eventually be met.

The If.. Then Statement

The syntax is

```
If Condition Then
Else If AnotherCondition Then
Else
End If
```

The For Loop

The For Next statement repeatedly loops through a block of code. The basic syntax is;

```
For counter = start to end
    ' block of code here
Next
```

The Exit For

The Exit For statement exits a For loop prematurely.

```
For counter = start to end
    if condition then Exit For
Next
```

The For Each Loop

The For Each loop is a variation on the For loop which is designed to iterate through a collection of objects as well as elements in an array. The general syntax is;

```
For Each ObjectVar in Collection
    ' block of code here
Next
```

The Do Loop

The Do Loop has several loop variations.

1.

```
Do while until condition
    ' code block
Loop
```

2.

```
Do
    ' code block
Loop while until condition
```

3.

```
Do
    ' code block
Loop
```

The While WEnd Loop

The While WEnd statement repeatedly loops through a block of code. The basic syntax is;

```
While until condition
    ' code block
WEnd
```

The Select Case Statement

You can also use the SELECT statement if you want to select one of many blocks of code to execute:

```
select case payment
 case "Cash"
   msgbox "pay cash"
 case "MasterCard"
   msgbox "pay by Mastercard"
 case Else
```

```
   msgbox "Unknown payment method"
end select
```

Expressions and Operators

An expression is a valid combination of constants, variables, literal values, operators and function results. Expressions are used to determine the value to assign to a variable, to compute the parameter of a function, or to test for a condition. Expressions can include function calls.

VB Script has a number of logical, arithmetic, Boolean and relational operators. Since these operators are grouped by the order of precedence which is different to the precedence orders used by Basic, C etc. For example, the AND and OR operators have precedence compared to the relational one.

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Division with integer result
^	Exponentiation
Mod	Modulo

Comparison Operators (lowest precedence)

=	Test whether equal or not.
<>	Test whether not equal or not.
<	Test whether less than or not.
>	Test whether greater than or not.
<=	Test whether less than or equal to or not.
>=	Test whether greater than or equal to or not.
Is	Compares two object reference variables.

String Operators

&	Concatenation

Logical Operators

Not	Logical NOT
And	Logical AND

Or	Logical OR
XOR	
Eqv	
Imp	
&	

VB Script Sub routines and functions

In this section:

- Passing parameters to procedures
- Dates Times
- File IO Routines
- Math Routines
- String Routines
- Server Process Routines.

Passing parameters to Sub routines and functions

When you define a function or sub routine in a script that can accept parameters, you can pass variables to the function or sub routine in two ways: by reference or by value.

To declare the method that parameters are passed, use the **ByRef** or **ByVal** keywords in the parameter list when defining the function or sub routine in a **Sub** or **Function** statement. For example, the following code fragment defines a sub routine that accepts two parameters. The first is passed by value and the second by reference:

```
Sub Test (ByVal Param1 As Integer , ByRef B As String)
```

The difference between the two methods is that **ByRef** passes a reference to the variable passed and allows the sub routine or function to make changes to the actual variables that are passed in as parameters (this is the default method of passing parameters and is used if the method is not explicitly declared).

The **ByVal** passes the value of the variable only. The sub routine or function can use this value, but the original variable passed is not altered.

The following examples illustrate the differences between methods. The main procedure is as follows:

```
Sub Main
 Dim X, Y
 X = 45 : Y = "Number"
 Test X, Y      ' Call to a subprocedure called Test.
 MsgBox X
 MsgBox Y
End Sub
```

The above procedure includes a call to a subprocedure, **Test**. If the subroutine is defined as follows:

```
Sub Test (ByRef A, ByRef B)
 B = B & " = " & A : A = 10*A
End Sub
```

then the variables X and Y in the main procedure are referenced directly by the sub routine. The result is that the values of X and Y are altered by the sub routine so that after the Test is executed X = 450 and Y = "Number = 45".

If, however, the sub routine is defined as follows:

```
Sub Test (ByVal A, ByVal B)
 B = B & " = " & A : A = 10*A
End Sub
```

Then after Test is executed X = 45 and Y = "Number", i.e. they remain unchanged.

If the sub routine is defined as follows:

```
Sub Test (ByRef A, ByVal B)
 B = B & " = " & A : A = 10*A
End Sub
```

Then after Test is executed, X = 450 and Y = "Number". Because Y was passed by value, it remains unchanged.

You can override the **ByRef** setting of a function or sub routine by putting parentheses around a variable name in the calling statement. Calling Test with the following statement:

```
Test (X), Y
```

would pass the variable X by value, regardless of the method defined for that parameter in the procedure definition.

Dates and Times routines

The VB Script language set supports a set of Date/Time routines and a few routines outlined below:

- Date
- Day
- Hour
- IsDate
- Minute
- Month
- Now
- Second
- Time
- Year

File IO Routines

The VB Script language set supports a set of File IO routines:

- Dir
- FileLen
- FileTimeDate
- FileCopy
- Kill
- Name
- RmDir
- MkDir

Math Routines

The VB Script language set supports a set of Math routines:

- Abs
- Atn
- Cos
- Exp
- Log
- Not
- Oct
- Rnd
- Sin

- Sqn
- Tan

String Routines

The VB Script language set supports a set of String routines and a few string routines outlined below:

- Asc
- Chr
- Format
- InStr
- InStrRev
- LCase
- Len
- Left
- Mid
- Right
- Str
- Trim
- LTrim
- RTrim
- UCase

Server Process Routines

The server process routines are used when you are dealing with processes in your scripts especially if you need to extract or set strings for the parameters of processes.

To execute processes and parameters in scripts, use the following functions

- AddColorParameter
- AddIntegerParameter
- AddLongIntParameter
- AddSingleParameter
- AddWordParameter
- GetIntegerParameter
- GetStringParameter
- ResetParameters
- RunProcess

Useful functions

- SetCursorBusy
- ResetCursor

- CheckActiveServer
- GetActiveServerName
- GetCurrentDocumentFileName
- RunApplication
- SaveCurrentDocument

Useful Dialogs

- ConfirmNoYes
- ConfirmNoYesCancel
- ShowError
- ShowInfo
- ShowWarning

Forms and Components

Although Forms and Components are based on Borland Delphi's Visual Component Library, you still use the **Tool Palette** to drop controls on a form and generate VB Script based event handlers and write code in VB Script language.

In this section:

- Components
- Designing Script Forms
- Writing Event Handlers.

Components

The scripting system handles two types of components: Visual and Nonvisual components. The visual components are the ones you use to build the user interface, and the nonvisual components are used for different tasks such as these Timer, OpenDialog and MainMenu components. You use the Timer nonvisual component to activate specific code at scheduled intervals and it is never seen by the user. The Button, Edit and Memo components are visual components for example.

Both types of components appear at design time, but non visual components are not visible at runtime. Basically components from the **Tool Palette** panel are object orientated and all these components have the three following items:

- Properties
- Events
- Methods

A property is a characteristic of an object that influence either the visible behaviour or the operations of this object. For example the Visible property determines whether this objet can be seen or not on a script form.

An event is an action or occurrence detected by the script. In a script the programmer writes code for each event handler which is designed to capture a specific event such as a mouse click.

A method is a procedure that is always associated with an object and define the behavior of an object.

All script forms have one or more components. Components usually display information or allow the user to perform an action. For example a Label is used to display static text, an Edit box is used to allow user to input some data, a Button can be used to initiate actions.

Any combination of components can be placed on a form, and while your script is running a user can interact with any component on a form, it is your task, as a programmer, to decide what happens when a user clicks a button or changes a text in an Edit box.

The Scripting system supplies a number of components for you to create complex user interfaces for your scripts. You can find all the components you can place on a form from the Toolbox palette.

To place a component on a form, locate its icon on the **Tool Palette** panel and double-click it. This action places a component on the active form. Visual representation of most components is set with their set of properties. When you first place a component on a form, it is placed in a default position,

with default width and height however you can resize or re-position this component. You can also change the size and position later, by using the Object Inspector.

When you drop a component onto a form, the Scripting system automatically generates code necessary to use the component and updates the script form. You only need to set properties, put code in event handlers and use methods as necessary to get the component on the form working.

Designing Script Forms

A script form is designed to interact with the user within the Altium Designer environment. Designing script forms is the core of visual development in the Altium Designer. Every component you place on a script form and every property you set is stored in a file describing the form (a DFM file) and has a relationship with the associated script code (the VBS file). Thus for every script form, there is the VBS file and the corresponding DFM file.

When you are working with a script form and its components, you can operate on its properties using the *Object Inspector* panel. You can select more than one component by shift clicking on the components or by dragging a selection rectangle around the components on this script form. A script form has a title which is the **Caption** property on the *Object Inspector* panel.

Creating a new script form

With a project open in Altium Designer, right click on a project in the *Projects* panel, and a pop up menu appears, click on the **Add New to Project** item, and choose **Script Form** item. A new script form appears with the Form1 name as the default name.

Displaying a script form

In a script, you will need to have a routine that displays the form when the script form is executed in Altium Designer. Within this routine, you invoke the ShowModal method for the form. The **Visible** property of the form needs to be false if the ShowModal method of the script form is to work properly.

ShowModal example

```
Sub RunDialog
    DialogForm.ShowModal
End Sub
```

The ShowModal example is a very simple example of displaying the script form when the RunDialog from the script is invoked. Note, you can assign values to the components of the DialogForm object before the DialogForm.ShowModal is invoked.

ModalResult example

```
sub bOKButtonClick(Sender)
  ModalResult := mrOK
end sub

sub bCancelButtonClick(Sender)
  ModalResult := mrCancel
end sub
```

```
sub RunShowModalExample
    'Form Visible property must be false for ShowModal to work properly.
    If Form.ShowModal = mrOk     Then ShowMessage("mrOk")
    If Form.ShowModal = mrCancel Then ShowMessage("mrCancel")
end sub
```

The **ModalResult** property example here is a bit more complex. The following methods are used for buttons in a script form. The methods cause the dialog to terminate when the user clicks either the OK or Cancel button, returning **mrOk** or **mrCancel** from the **ShowModal** method respectively.

You could also set the **ModalResult** value to **mrOk** for the OK button and **mrCancel** for the Cancel button in their event handlers to accomplish the same thing. When the user clicks either button, the dialog box closes. There is no need to call the Close method, because when you set the **ModalResult** method, the script engine closes the script form for you automatically.

Note, if you wish to set the form's **ModalResult** to cancel, when user presses the **Escape** key, simply enable the **Cancel** property to **True** for the **Cancel** button in the Object Inspector panel or insert **Sender.Cancel := True** in the form's button cancel click event handler.

Accepting input from the user

One of the common components that can accept input form the user is the **EditBox** component. This **EditBox** component has a field where the user can type in a string of characters. There are other components such as masked edit component which is an edit component with an input mask stored in a string. This controls or filters the input.

The example below illustrates what is happening, when user clicks on the button after typing something in the edit box. That is, if the user did not type anything in the edit component, the event handler responds with a warning message.

```
sub TScriptForm.ButtonClick(Sender)
    If Edit1.Text = “” Then
        ShowMessage(“Warning - empty input!”)
        Exit
    End
    ' do something else for the input
End sub
```

Note, A user can move the input focus by using the Tab key or by clicking with the house on another control on the form.

Responding to events

When you press the mouse button on a form or a component, Altium Designer sends a message and the Scripting System responds by receiving an event notification and calling the appropriate event handler method.

Writing Event Handlers

Each component, beside its properties, has a set of event names. You as the programmer decide how a script will react on user actions in Altium Designer. For instance, when a user clicks a button on a

form, Altium Designer sends a message to the script and the script reacts to this new event. If the OnClick event for a button is specified it gets executed.

The code to respond to events is contained in event handlers. All components have a set of events that they can react on. For example, all clickable components have an OnClick event that gets fired if a user clicks a component with a mouse. All such components have an event for getting and loosing the focus, too. However if you do not specify the code for OnEnter and OnExit (OnEnter - the control has focus; OnExit - the control loses focus) the event will be ignored by your script.

Your script may need to respond to events that might occur to a component at run time. An event is a link between an occurrence in Altium Designer such as clicking a button, and a piece of code that responds to that occurrence. The responding code is an event handler. This code modifies property values and calls methods.

List of properties for a component

To see a list of properties for a component, select a component and in the **Object Inspector,** activate the **Properties** tab.

List of events for a component

To see a list of events a component can react on, select a component, and in the **Object Inspector** activate the **Events** tab. To create an event handling procedure, decide on what event you want your component to react, and double click the event name.

For example, select the Button1 component from the **Toolbox** panel and drop it on the script form, and double click next to the **OnClick** event name. The scripting system will bring the Code Editor to the top of the Altium Designer and the skeleton code for the OnClick event will be created.

For example, a button has a Close method in the CloseClick event handler. When the button is clicked, the button event handler captures the on click event, and the code inside the event handler gets executed. That is, the Close method closes the script form.

In a nutshell, you just select a button component, either on the form or by using the **Object Inspector** panel, select the **Events** page, and double click on the right side of the OnClick event, a new event handler will appear on the script. OR double click on the button and the scripting system will add a handler for this OnClick event. Other types of components will have completely different default actions.

List of methods for a component

To see a list of methods for a component, see the Components Reference.

Using components in your scripts

Dropping components on a script form

To use components from the Tool Palette panel in your scripts, you need to have a script form first before you can drop components on the form. Normally when you drop components on a script form, you do not need to create or destroy these objects, the script form does them for you automatically.

The scripting system automatically generates code necessary to use the component and updates the script form. You then only need to set properties, put code in event handlers and use methods as necessary to get the script form working in Altium Designer.

Creating components from a script

You can also directly create and destroy components in a script – normally you don't need to pass in the handle of the form because the script form takes care of it automatically for you, thus you just normally pass a Nil parameter to the Constructor of a component.

For example, you can create and destroy Open and Save Dialogs (TOpenDialog and TSaveDialog classes as part of Borland Delphi Run Time Library).

Index

Revision History

Date	Version No.	Revision
01-Dec-2004	1.0	New product release
26-Apr-2005	1.1	Altium Designer
15-Dec-2005	1.2	Updated for Altium Designer 6
6-Jul-2006	1.3	Updated for Altium Designer 6.3 DXP references changed to Altium Designer. For To Next syntax corrected and While WEnd loop added.
1-Sept-2006	1.4	Updated for Altium Designer 6.4 Typing of variables corrected. Variables are typeless and functions declaration clarified.