

Summary

This tutorial describes the scripting system and how to write and execute simple scripts in Altium Designer using the DelphiScript scripting language set. The concepts of Object Interfaces and Server processes used in Altium Designer are also briefly covered.

This tutorial is designed to give you an overview of how to write scripts using the DelphiScript language supported in Altium Designer. It will outline how to create a script project and to store different scripts in this script project. It also briefly covers concepts such as editing scripts, debugging scripts, accessing DXP object interfaces and executing server processes.

The Scripting System in Altium Designer supports DelphiScript, EnableBasic, VBScript and JavaScript (both based on Microsoft ActiveX scripting engine) and TCL scripting language sets. In this tutorial, you will learn how to harness the power of the DelphiScript, the scripting language based largely on the Borland Delphi language set.

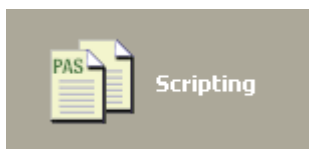
Script Projects support two script document types – Script Units and Script Forms. Script Units allow you to write standalone procedures and functions. Script Forms allow you to build dialogs with controls and event handlers as well as procedures and functions. Script Projects including Script Forms and Script Units will be explained in more detail in this tutorial.

Creating a new Script Project

To start writing a script, first create a Script Project which will help you to manage your scripts. In this tutorial, you will write two different scripts, one using a DelphiScript Unit and another using a DelphiScript Form. We will start with writing a script based on DelphiScript Unit that displays a Hello World! message on a dialog in Altium Designer.

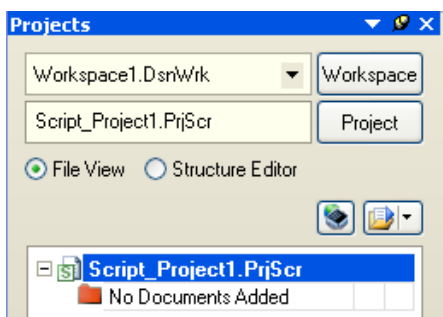
To create a new Script Project:

1. Click on **Script Development** in the **Pick a Task** section of the Home Page, then click **New Blank Script Project**.



Alternatively, you could click on **Blank Script Project** in the **New** section of the **Files** panel. If this panel is not displayed, click on the **Files** tab or click **System** button at the bottom right part of the Altium Designer application and a popup menu of different panels appears. Click **Files** item to activate the **Files** panel.

2. The **Projects** panel displays. The new project file, `Script_Project1.PrjScr`, is listed here with no documents added.

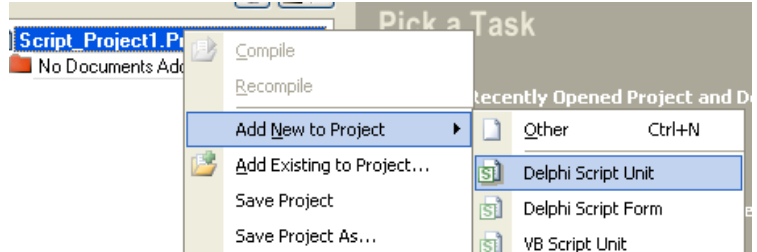


3. Rename the new project file (with a `.PrjScr` extension) by selecting **File » Save Project As**. Navigate to your desired location to save the project, type the name `HelloWorld.PrjScr` in the **File name** field and click on the **Save** button.

Creating a new DelphiScript Unit

We need to create a single DelphiScript Unit for the Hello World example.

1. Right click on the script project filename in the *Projects* panel, a popup menu appears. Select the **Add New to Project** menu item and then select **Delphi Script Unit** sub menu item. A blank script document named `EditScript1.pas` is then displayed in the design window.
2. Rename this new script file (with a `.pas` extension) by selecting **File » Save As**. Navigate to your desired location to save the script, type the name `HelloWorld.pas` in the **File name** field and click on the **Save** button.



Writing and Executing a DelphiScript Unit

You are now ready to write a script. You will write a script that has a procedure with no parameters. All procedures or functions that don't have parameters appear in the *Select Item To Run* dialog. More information about the *Select Item to Run* dialog will be discussed later in this document.

Within this procedure is a DelphiScript standard `ShowMessage` function, when this function is executed, it shows a simple dialog with the "Hello World" message.

1. Enter the following lines, as shown below, for this `HelloWorld` script.

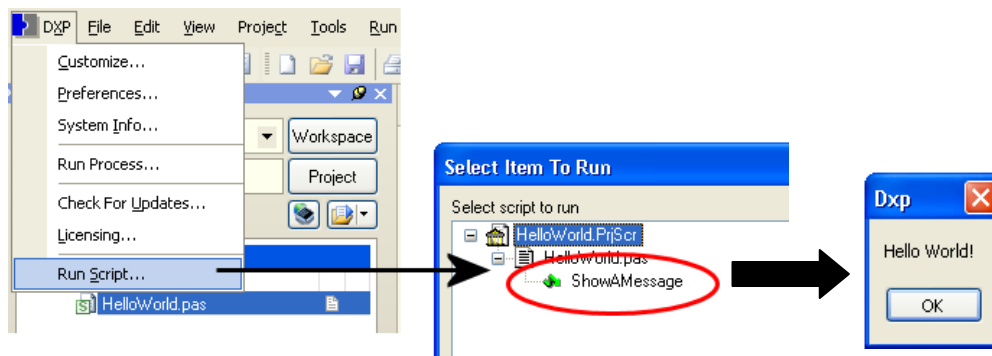
```
Procedure ShowAMessage;
Var
    DefaultMessage;
Begin
    DefaultMessage := 'Hello World!';
    ShowMessage(DefaultMessage);
End;
```

Every time a document is edited, an asterisk appears next to the script document and the icon on the **Projects** panel changes to red.

2. Save this script by selecting **File » Save**.

Since all variables are of variant type in scripts so there is no need to define the `DefaultMessage` variable type in the script.

To execute your script, invoke the *Select Item To Run* dialog from the **DXP** Menu and select **Run Script**. The *Select Item To Run* dialog displays parameter-less procedures and functions only.



You can execute your script using the Run command such as the **Run** button in the Text Editor. You will need to specify the main procedure name from your script in the **Run » Set Project Startup Procedure** menu first before you can use the **Run** command.

The *Select Item to Run* dialog appears with the `ShowAMessage` procedure in the list. Double click the `ShowAMessage` item, the `ShowAMessage` dialog appears as shown above.

If there are errors in your script, Altium Designer will prompt you with an error message. Examine your script to make sure it is the same as the example shown above.

Creating and Executing a DelphiScript Form

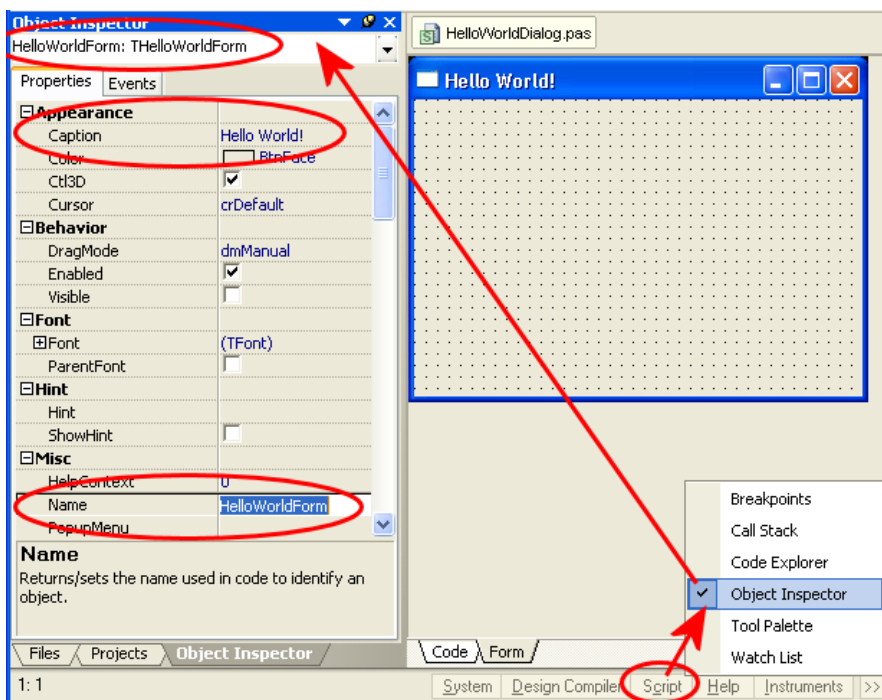
You are now ready to create a DelphiScript Form in the existing `HelloWorld` project. A script form is a window that can host different kinds of controls such as buttons, memos and list boxes and has event handlers that respond when a control has generated an event such as a button being clicked.

You will create a simple script form as a dialog with two buttons. One button will be used to display a Hello World! message on another dialog and the second button to close the dialog. Note, the terms dialog and script form (or form for short) will be used interchangeably. The “form” term refers to the window that is being designed real-time in Altium Designer and a “dialog” is an active form that is waiting for user feedback and takes action when a button or a control on this dialog has been clicked.

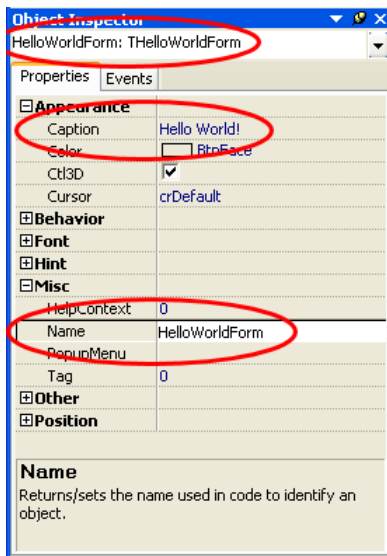
1. Right click on the `HelloWorld.PrjScr` filename in the *Projects* panel (not on the script filename), a popup menu appears. Select **Add New to Project** and then select **Delphi Script Form** from this popup menu. A blank script document named `EditScript1.pas` displays in the design window. You will notice there are two tabs at the bottom of this script document; **Code** and **Form** tabs.



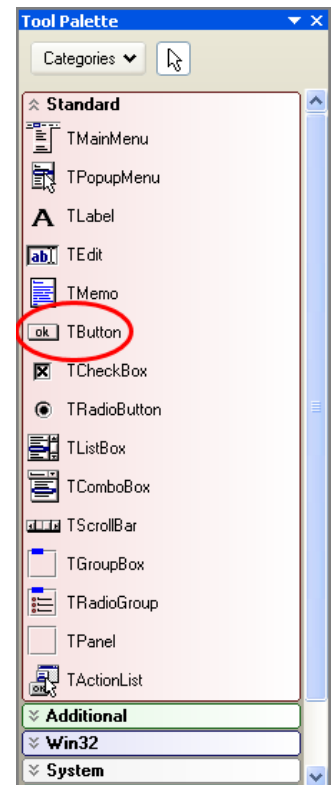
2. Rename the new script file (with a `.pas` extension) by selecting **File » Save As**. Navigate to your desired location to save the script, type the name `HelloWorldDialog.pas` in the **File Name** field and click the **Save** button.
3. Click on the **Form** tab, a window with a form appears. Select **Object Inspector** menu item from the **Script** button on the bottom of the design window. The *Object Inspector* panel appears with the properties and their values for the form window. You can change the color, size etc of the Form and insert your code in the event handlers associated with this form.



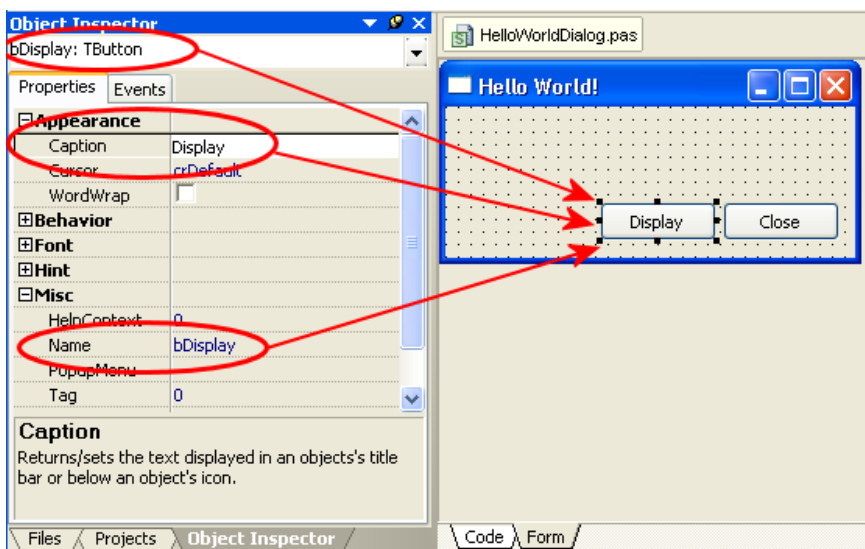
4. The *Object Inspector* panel shows the values of the properties for the **currently focused** form or its components. Click on the form first, and then you can change the **Name** and the **Caption** properties of the form on the *Object Inspector* panel to `HelloWorldForm` and `Hello World!` respectively, as shown in the *Object Inspector* panel on the right.



You can place components on a script form by double clicking the component on the **Tool Palette** panel or clicking the component once and then clicking on the form where you want the component to appear.



- Bring up the **Tool Palette** panel from the **Script** button on the bottom of the design window, this is a component palette consisting of different window controls. We wish to place two buttons on the form, thus click **TButton** (image of an OK button) on the **Standard** tab of the **Tool Palette** panel as shown below. Do this twice, so you get two buttons on the form.
- There are two buttons, **Button1** and **Button2** on the form. We will rename Button1 caption to Display, and the Button2 caption to Close. First, we will rename Button1. To do this, we need the **Object Inspector** panel to change the **Caption** and **Name** properties. Click on the first button, Button1, and then type **Display** in the **Caption** field and **bDisplay** in the **Name** field. Repeat for Button2 and change the **Caption** and **Name** properties to **Close** and **bClose** respectively on the **Object Inspector** panel.



Note, the comments box at the bottom of the **Object Inspector** panel gives a description of the highlighted property.

- Now we will put code in the event handlers for both the **Display** button and the **Close** button. The **Display** button will display a **ShowMessage** dialog on top of this form, and the **Close** button will close this form. Double click on the **Display** button on the form or click on the **Events** tab of the **Object Inspector** panel. Scroll to locate the **OnClick** event and double click the **OnClick** event. You will be switched to the **Code** view where you can type the **ShowMessage** statement in the event handler.

```
Procedure THelloWorldForm.bDisplayClick(Sender: TObject);
Begin
    ShowMessage('Hello World!');
End;
```

To view pre-defined event handlers for a component on your script form, select the component, then on the **Object Inspector** panel, click the **Events** tab.

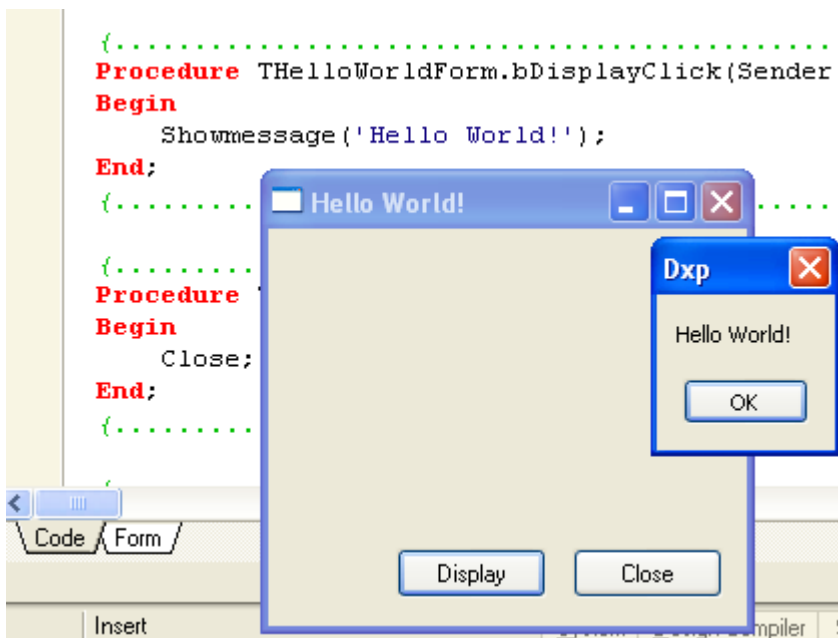
8. Define the event handler for this Close button by generating an OnClick event handler. Fill in the code for the Close click event handler of the Close button as follows:

```
Procedure THelloWorldForm.bCloseClick(Sender: TObject);
Begin
    Close;
End;
```

9. We need to have a procedure appropriately called `RunHelloWorld` in the script which is used as the starting point to call up the dialog from Altium Designer. Add the `RunHelloWorld` procedure at the end of the code script. Note the form has the `HelloWorldForm` name. It is important to have unique form names in the same script to avoid form name clashes.

```
Procedure RunHelloWorld;
Begin
    HelloWorldForm.ShowModal;
End;
```

10. Save the script and then execute the **RunScript** command from **DXP** system menu. The *Select Item to Run* dialog appears. Double click on the `RunHelloWorld` procedure item under the `HelloWorldDialog` entry on this dialog.



An existing copy of the Hello World script resides in the `\Examples\Scripts\Delphiscript Scripts\General` folder of the Altium Designer installation.


11. You can experiment with the properties of the Hello World form. For example to change the position of the form with respect to the screen of the monitor (desktop) or the workspace of Altium Designer, choose `poScreenCenter` value for the `Position` property of the `HelloWorldForm` form from the Object Inspector panel. Run the script and you will notice that the dialog is placed at the center of the desktop.

A Script Form has two associated files; a `*.PAS` file that contains has event handlers and procedures/functions and a `*.DFM` file that has details of the script form itself, the components on this form, and their locations on this form as well.

Scripting Terminology

Some of the terms commonly used in Altium Designer and Scripting are outlined below:

- Altium Designer Run Time Library has a collection of Application Programming Interfaces (APIs).
- Each API represents an editor in Altium Designer. For example the PCB editor has its PCB API, the Schematic editor has its Schematic API and the Project Manager has its Workspace manager API and so on. All of these APIs are in the Altium Designer Run Time Library.
- Each API has an Object Model which consist of object Interfaces. Each object interface can consist of methods and properties (but no variables).
- The APIs are automatically available to use in your scripts so there is no need to include the API units in your scripts.
- Object Interfaces represent design objects that you can use in your scripts (any language set supported by the scripting system) to extract and modify data from opened design documents in Altium Designer.
- Methods of Object Interfaces are the actions an Object Interface can perform.
- Properties of Object Interfaces represent the data contained in the object that is represented by the interface.
- DXP denotes the technology platform inside the Altium Designer application. DXP and Altium Designer terms can be interchanged in this document.
- A Script Form is a form that is supported by the script, when this form is active in Altium Designer, it is a dialog window.
- Components are visual objects from the *Tool Palette* panel that you can drop and manipulate on a Script Form at design time. Components are also called controls.
- A component consists of methods, properties, and events which can be used by a Script Form.
- Events are the conditions a component (control) on a Script Form can react to.
- Processes are command strings that you can use to execute commands from your scripts.

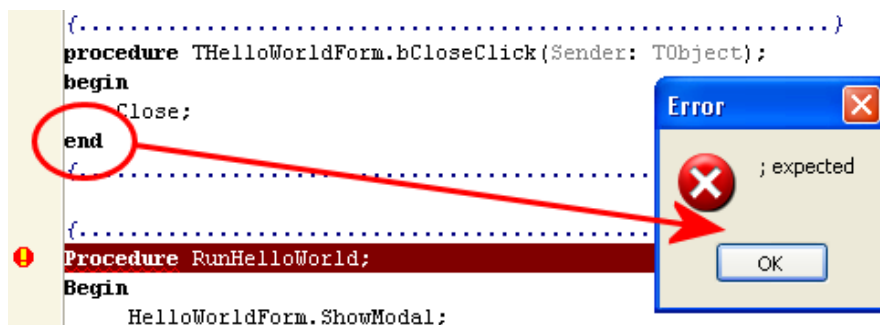
 For information on the differences between DelphiScript and Object Pascal (used in Borland Delphi) section, please refer to the [Delphi Script Reference](#) document.


You now know how to create Unit and Form scripts, next steps are to familiarize yourself with the debugger, DXP Run Time Library functions and DXP Object interfaces.

An Introduction to Debugging Scripts

The Scripting system is composed of two parts; the editor and the debugger. The Scripting Debugger has various panels including the *Watch List* panel, *Call Stack* panel and *Breakpoint List* panel.

The Scripting Debugger helps you to identify errors in your scripts. Common errors include errors in logic, invalid assumptions, misplaced grouping operators and typographical errors. An example is shown below with a missing terminating character ; and this is flagged as an error .



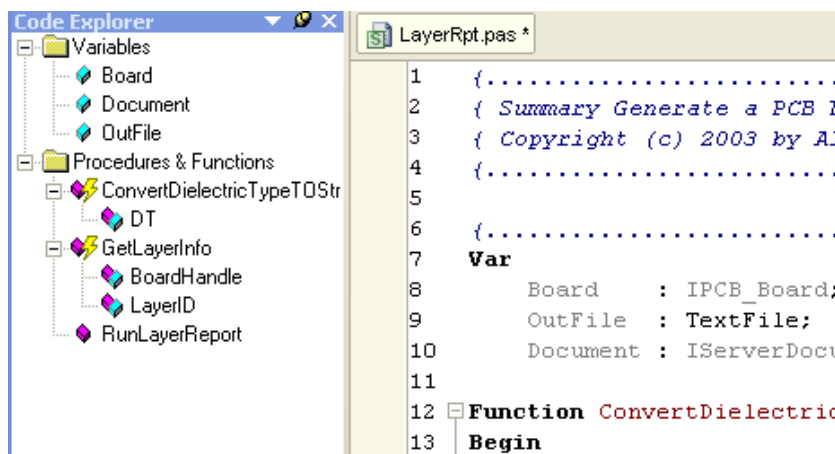
When an error dialog is displayed, close this dialog by clicking **OK** button and stop the scripting system by clicking on the **Stop** button  on the menu or pressing the keys, **Ctrl+ F3**.

There are various tools in the scripting system to help you debug your scripts including using break points in your script, using the *Watches List* panel to monitor the variable values and using bookmarks in the script to jump to different statements more efficiently.

 Refer to [A Tour of the Scripting System](#) for more information on debugging scripts.

Scripting System's Code Completion Features

The *Code Explorer* panel displays the variables, methods and objects used in the current script. The variables used in a current script are marked as aqua boxes under the **Variables** folder. The functions used are marked with pink box icons along with yellow flash, procedures used are marked just as pink box icons under the **Procedures & Functions** folder in the *Code Explorer* panel.

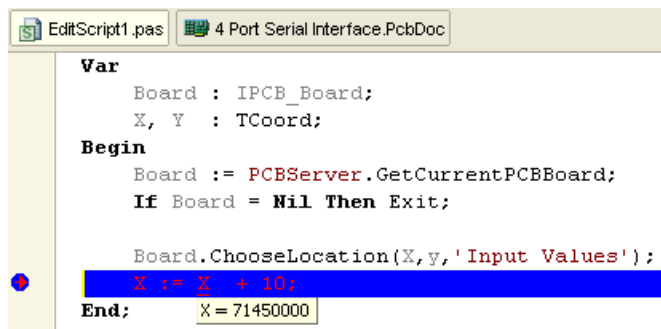


Code Completion Facility

The Scripting System supports the Code Completion facility. When you type an object interface name and a dot after this name, after a brief pause, the list of properties and methods for that object are displayed in a popup menu. For example, just after the *PCBServer* function in your script, type in the dot character '.' and after a brief pause, the Code Completion pop-up window appears with a list of properties and methods for this *PCBServer* function.



The Expression Evaluation ToolTip feature displays data values for the variable that the cursor is hovering over on the script.



 Refer to [A Tour of the Scripting System](#) for more information on script panels.

DXP Object Models

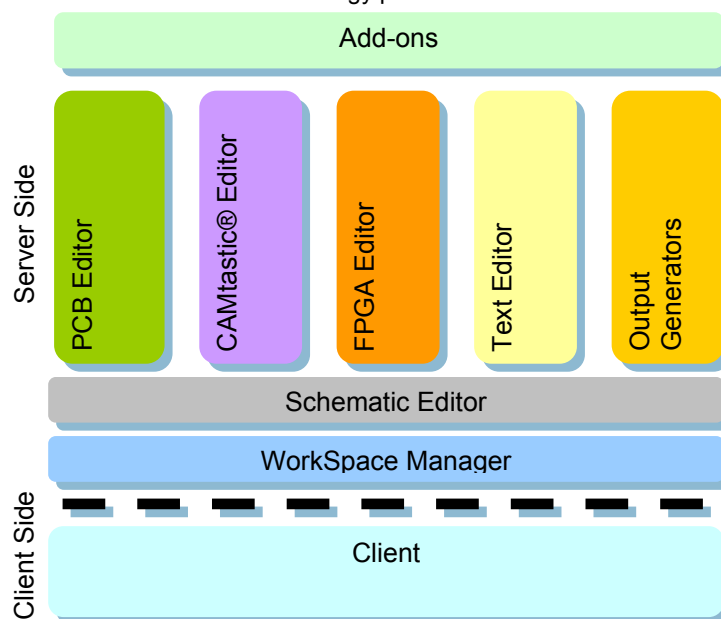
One of the powerful features of the scripting system is accessibility to the design objects from open design documents through the Application Programming Interface (API). This means you can read and modify an object's properties.

You can access objects in Altium Designer through the implementation of Borland Delphi's Object Interfaces technology. The scripting system is written in Borland Delphi and the *Tool Palette* panel is a subset of the Visual Component Library (VCL) which is also written in Borland Delphi.

Note, you have the ability to use the same components from the *Tool Palette* panel in your DelphiScript, JavaScript and VBScript Script Forms.

The Altium Designer system is composed of a single Client executable along with plugged in servers as shown in the diagram below. The client module is part of the DXP software technology platform.

Object Interfaces are available to use in your scripts. Basically Object interfaces reference existing objects in Altium Designer.



The Client executable deals with actions generated by the user of the Altium Designer application. The servers provide specialized functionality depending upon the task requested by the user.

The Schematic server and PCB server are two main document editor servers used for the design process, each having their own document types (design and library documents).

DXP Object Interfaces

The scripting system implements the Object Models from the DXP Run Time Library (RTL) which has a set of APIs and various Borland Delphi Objects along with a subset of the Borland Delphi's Visual Component Library. The Object Interfaces from these Object Models are exposed in the scripts. When you are writing a script whether it is in DelphiScript or VBScript, you can use the same objects and functions.

For example, the `ShowMessage` procedure is available for use in your scripts. This procedure implements a modal dialog from the DXP Run Time Library which can be used to display a dialog with a short text message along with an **OK** button.

An Object Interface will have methods and properties listed (not all interfaces will have both methods and properties listed, that is, some interfaces will only have methods). To recap;

- A method is a procedure or function that is invoked from its interface.
- A property of an object interface is like a variable, you get or set a value in a property, but some properties are read only properties meaning they can only return values but cannot be set. A property is implemented by its Get and Set methods.

For example, the `IPCB_Component` interface from the PCB API's Object model has a `Height` property and two associated methods supported by its Read and Write identifiers. Note, that the `IPCB_Component` declaration comes from the PCB API

(you cannot see the source code of the DXP RTL in the scripting system, but you can see the methods and properties for an object interface using the Code Completion feature when writing a script).

IPCB_Component interface declaration

```
IPCB_Component = Interface
...
Function GetState_Height : TCoord;
Procedure SetState_Height (Value : TCoord);
Property Height : TCoord Read GetState_Height Write SetState_Height;
...
End;
```

Set and get the IPCB_Component property example in a script

```
//Set the Selected value
PCBComponent.Selected := True;

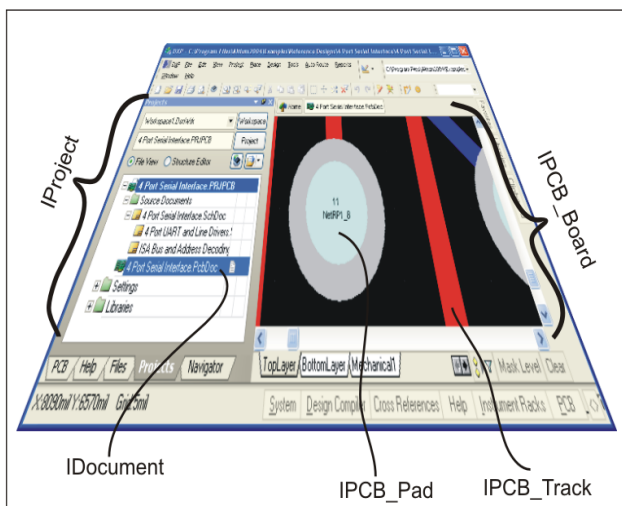
//Get the Selected value
ASelected := PCBComponent.Selected;
```

Using DXP Object Interfaces in a script

You don't create instances of DXP object interfaces, you just use a specific interface that references an existing object in Altium Designer. In Altium Designer, there is the Client Object model and several editor/server object models:

- Client Object Model
- Integrated Library Object Model
- WorkSpace Manager Object Model
- Schematic Object Model
- PCB Object Model
- Nexus Object Model

The figure below is an illustration of the relationship between objects in Altium Designer and the Object Interfaces supported by the various DXP Object Models.



From the figure above, the projects and the corresponding documents are managed by the Workspace Manager. A project open in Altium Designer is represented by the `IProject` object interface, and the documents from this project are represented by the `IDocument` interfaces.

The PCB documents and PCB design objects are managed by the PCB Editor and its PCB Object Model. The PCB document open is represented by its `IPCB_Board` interface and the design objects, for example, the pad object and the track object are represented by `IPCB_Pad` and `IPCB_Track` interfaces.


PCB Object Interfaces

The PCB Object Model enables scripts to have access to PCB objects and PCB documents. The gateway to PCB objects on a PCB document or in a PCB Editor Server is the `PCBServer` function. Invoke the `PCBServer` function in the script, this function returns the `IPCB_ServerInterface` interface, representing the PCB Editor server.

Normally we invoke the `GetCurrentPCBBoard` method from the `PCBServer` function to get the `IPCB_Board` interface. The `IPCB_ServerFunction` encapsulated by the `PCBServer` function has a `GetCurrentPCBBoard` method amongst other methods.

To take advantage of the code completion feature in a script, type the Object name and the dot after the object name. The Code Completion popup window appears after a short period with a list of properties and methods for that object. The screen shot below shows the list of available properties and methods for the `IPCB_Board` interface.

```

1  Var
2      Board : IPCB_Board
3  Begin
4      If Board := PCBServer.GetCurrentPCBBoard;
5      Board.f
6
7      
8  End;

```

Many objects in Altium Designer are referenced by their corresponding interfaces, for example pad objects are encapsulated by `IPCB_Pad` interfaces and PCB documents are encapsulated by `IPCB_Board` interfaces.

In the script example below, we invoke the `GetCurrentPCBBoard` method from the `PCBServer` function and this method returns the `IPCB_Board` interface. The `IPCB_Board` interface represents the current PCB document in Altium Designer.

This script example counts the number of pads on a PCB document, therefore, firstly you need to have a PCB document open with a number of pad objects on it before running the script on this document.

This example uses an iterator to look for specific PCB design objects on a PCB document. You can control how an iterator looks for specified objects within a specified region or on a specified layer. In this case, only pad objects are searched on all layers of the PCB document.

PCB Interfaces Example

```

Procedure CountPads;
Var
    Board      : IPCB_Board;
    Pad        : IPCB_Pad;
    Iterator    : IPCB_BoardIterator;
    PadNumber   : Integer;
Begin
    // Retrieve the current board
    Board      := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    // Create the iterator and set up the search criteria
    Iterator    := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet (MkSet (ePadObject));
    Iterator.AddFilter_LayerSet (AllLayers);
    Iterator.AddFilter_Method (eProcessAll);
    // Search and count pads
    PadNumber := 0;
    Pad := Iterator.FirstPCBObject;
    While (Pad <> Nil) Do
    Begin
        PadNumber := PadNumber + 1;
        Pad := Iterator.NextPCBObject;
    End;
    // Finished looking for pads, thus destroy the iterator.
    Board.BoardIterator_Destroy (Iterator);

    ShowMessage (IntToStr (PadNumber));
End;
```



Open the **PCB_Scripts** project from `\Examples\DelphiScript Scripts\PCB\` and run the `Count_Pads.pas` script using the **Run Script** command from the **DXP** menu. Note, you will need to have a PCB document open and in focus before running this script.



Consult the Scripting Resources accessible through the bottom part of the *Knowledge Center* panel in Altium Designer. Navigate to the PCB API Reference document via **Configuring the System » Scripting in Altium Designer » Altium Designer RTL Reference**.

Schematic Object Interfaces

The Schematic Object Model enables scripts to have access to Schematic objects and Schematic documents. The gateway to Schematic objects on a Schematic document or in a Schematic Editor Server is the `SCHServer` function which returns the `ISCH_ServerInterface` interface. This object interface represents the Schematic Editor server.

The script example below when executed, uses a method that waits for the user to click on two points that define a boundary box on a schematic sheet. The spatial iterator, which is a search object, then looks for components inside the defined boundary on the sheet and reports their locations in terms of X,Y coordinates. Again, you need to have a schematic document open first before running this script. The spatial iterator is encapsulated by the `ISch_Iterator` interface in the Schematic Object Model.

Schematic Interfaces Example

```

Procedure RunSpatialIteratorExample;
Var
    CurrentSheet      : ISch_Document;
    SpatialIterator    : ISch_Iterator;
    P,P1              : String;
    Rect              : TCoordRect;
    GraphicalObj       : ISch_GraphicalObject;
Begin
    // Obtain the schematic server interface and the
    // current schematic document
    If SchServer = Nil Then Exit;
    CurrentSheet := SchServer.GetCurrentSchDocument;
    If CurrentSheet = Nil Then Exit;

    // Obtain the coordinates from the corners clicked by the user.
    If Not CurrentSheet.ChooseRectangleInteractively(Rect,
        'Please select the first corner',
        'Please select the second corner') Then Exit;

    // Create a spatial iterator
    SpatialIterator := CurrentSheet.SchIterator_Create;
    If SpatialIterator = Nil Then Exit;

    Try
        // Look for components only within the defined boundary
        SpatialIterator.AddFilter_ObjectSet(MkSet(eSchComponent));
        SpatialIterator.AddFilter_Area(Rect.left, Rect.bottom, Rect.right, Rect.top);

        GraphicalObj := SpatialIterator.FirstSchObject;
        While GraphicalObj <> Nil Do
            Begin
                P1 := 'X:' + IntToStr(GraphicalObj.Location.X) +
                    ', Y:' + IntToStr(GraphicalObj.Location.Y);
                P := P + P1 + #13;
                GraphicalObj := SpatialIterator.NextSchObject;
            End;
    End;

```

```

Finally
    CurrentSheet.SchIterator_Destroy(SpatialIterator);
End;
ShowInfo('Component objects found at'      + #13 + P);
End;

```



You can open the SCH_Scripts.PrjScr project and execute the RunSpatialIteratorExample procedure from the UsingASpatialIterator.pas script located in the \Examples\DelphiScript Scripts\SCH\ folder on a schematic document



Consult the Scripting Resources accessible through the bottom part of the *Knowledge Center* panel in Altium Designer. Navigate to the Schematic API Reference document via **Configuring the System » Scripting in Altium Designer » Altium Designer RTL Reference**.

Workspace Manager Object Interfaces

The WorkSpace Manager server deals with projects and their associated documents. This server provides project and multi-sheet design support, compiling, connectivity navigation tools and so on. The Workspace manager is schematic-centric and also manages output generators such as netlisters. To retrieve the Workspace Manager interface, invoke the `GetWorkspace` function.

The example below shows how to insert strings in the *Messages* panel in Altium Designer. The *Messages* panel is encapsulated by the `IMessagesManager` interface from the `WorkSpaceManager` object model.

Workspace Manager Interfaces Example

```

Var
    WSM          : IWorkSpace;
    MM           : IMessagesManager;
    ImageIndex   : Integer;
Begin
    // Obtain the interface of the workspace manager server
    WSM := GetWorkSpace;
    If WSM = Nil Then Exit;
    MM := WSM.DM_MessagesManager;
    If MM = Nil Then Exit;
    // Tick Icon for the lines in the Messages panel
    ImageIndex := 3;
    // Bring the Messages panel into view.
    WSM.DM_ShowMessageView;
    MM.ClearMessages;
    MM.BeginUpdate;
    MM.AddMessage('C1','T1','DXP Msg','Doc 1','','', ImageIndex,False);
    MM.AddMessage('C2','T2','DXP Msg','Doc 2','','', ImageIndex,False);
    MM.EndUpdate;
End;

```



You can open the WSM_Scripts.PrjScr project and execute the UsingMessagePanel.pas script from the \Examples\DelphiScript Scripts\WSM\ folder.



Consult the Scripting Resources accessible through the bottom part of the *Knowledge Center* panel in Altium Designer. Navigate to the Workspace Manager API document via **Configuring the System » Scripting in Altium Designer » Altium Designer RTL Reference** and open the Workspace Manager API Reference.

Client Object Interfaces

The Client executable module within the Altium Designer application is responsible for dispatching commands to the appropriate server and handles the user interfaces and all other system level functions. To retrieve the Client module interface, invoke the `Client` function. The `Client` function retrieves the `IClient` interface which encapsulates the Client executable module.

In the `OpenAndShowATextDocument` script example below, the `OpenDocument` method of the `IClient` interface is used to open (but not show) a text file in Altium Designer. The `OpenDocument` method returns a `IServerDocument` interface which is used as a parameter for the `ShowDocument` method which displays the document in Altium Designer.

Client Interfaces Example

```
Procedure OpenAndShowATextDocument;
Var
    Filename      : TString;
    ReportDocument : IServerDocument;
Begin
    If Client = Nil Then Exit;
    // Opens a text file only
    FileName := 'C:\Program Files\Altium Designer\Readme.txt';
    ReportDocument := Client.OpenDocument('Text',FileName);
    If ReportDocument <> Nil Then
        Client.ShowDocument(ReportDocument);
End;
```



You can open the `DXP_Scripts.PrjScr` project and execute the `OpenADoc.pas` script from the `\Examples\DelphiScript Scripts\DXP\` folder.



Consult the Scripting Resources accessible through the bottom part of the *Knowledge Center* panel in Altium Designer. Navigate to the System Reference document via **Configuring the System » Scripting in Altium Designer » Altium Designer RTL Reference**.

Object Interfaces Examples

To run a script that needs access to DXP Object interfaces, you need to run the script on a focused document in Altium Designer. For example, if your script has PCB object interfaces and you attempt to run the script in the text editor, you will get undeclared identifier errors. You need to run the script on an open PCB document. The script examples, mostly in `DelphiScript`, are in the `\Examples\Scripts\` folder.



For more information on script examples, refer to the [Script Examples Gallery Reference](#).

Executing Processes from a Script

The Client module of the Altium Designer interprets the commands in terms of server processes and then delegates these processes to the appropriate servers. A command is an action performed and is supported by a process string via a packaged process launcher.

A process is implemented as a `server name:server process` string. Processes are stored in a command launcher table maintained by the server. Every time you execute a process via the Graphical User Interface in Altium Designer, it consults the appropriate server's command table to fetch the process string and then sends this string over to the server to determine which process to execute. The functionality of a server is exposed by that server's processes.

For example, when a user is clicking on the Schematic menu to place a wire, the Client interprets this action as a `'SCH:PlaceWire'` command and delegates the command to the Schematic Editor server. The Schematic server responds by executing the command.

In a nutshell, a server is supported by its set of processes and these processes act as a link between the Graphical User Interface and this server.

A simple example presented below places a new PCB Polygon on a PCB document. You need to have a new PCB document open first. Open the `Processes_Scripts.PrjScr` project from `\Examples\Scripts\DelphiScript` `Scripts\Processes` folder and then open the `PlacePCBPolygon.pas` script.

The `PlacePCBPolygon` script uses PCB Server processes. The `ResetParameters` method is invoked first to clear out the process buffer in Altium Designer. Setup a server process using `AddIntegerParameter`, `AddStringParameter` etc and finally invoking the server process with the `RunProcess` method.

```

Procedure PlaceAPolygon;
Begin
    ResetParameters;
    AddIntegerParameter('Location.X', 5000);
    AddIntegerParameter('Location.Y', 5000);
    AddStringParameter('PourOver', 'True');
    AddStringParameter('RemoveDead', 'False');
    AddStringParameter('GridSize', '10');
    AddStringParameter('TrackWidth', '12');
    AddStringParameter('HatchStyle', '90Degree');
    AddStringParameter('Name', 'GND');
    AddStringParameter('Layer', 'Top');
    AddStringParameter('PourOver', 'True');
    AddStringParameter('PolygonType', 'Polygon');
    AddStringParameter('Selected', 'True');

    // Polygon Vertices
    AddStringParameter('Kind0', '0');
    AddStringParameter('Vx0', '1000');
    AddStringParameter('Vy0', '1000');

    AddStringParameter('Kind1', '0');
    AddStringParameter('Vx1', '2500');
    AddStringParameter('Vy1', '1000');

    AddStringParameter('Kind2', '0');
    AddStringParameter('Vx2', '2500');
    AddStringParameter('Vy2', '2500');

    AddStringParameter('Kind3', '0');
    AddStringParameter('Vx3', '1000');
    AddStringParameter('Vy3', '2500');

    AddStringParameter('Kind4', '0');
    AddStringParameter('Vx4', '1000');
    AddStringParameter('Vy4', '1000');

    RunProcess('PCB:PlacePolygonPlane');
End.

```



Consult the [Server Process Reference](#) documentation for more information about Server Processes



Check out the `Processes` examples in the `\Examples\Scripts\Processes` folder.

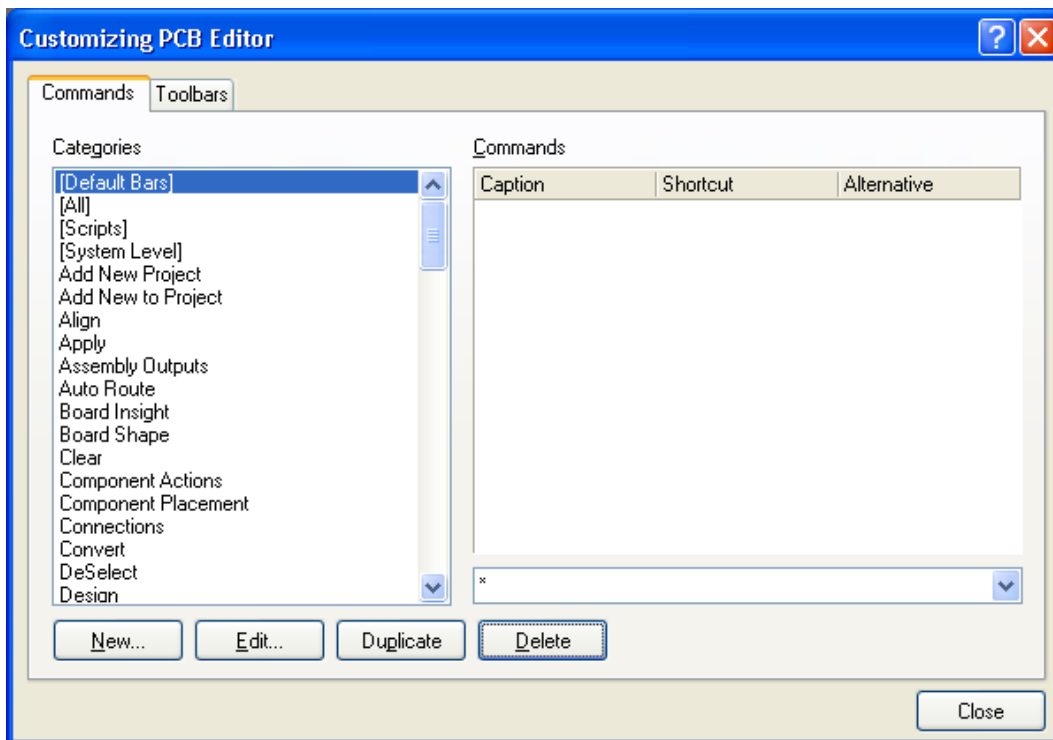
Executing Scripts within Altium Designer

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

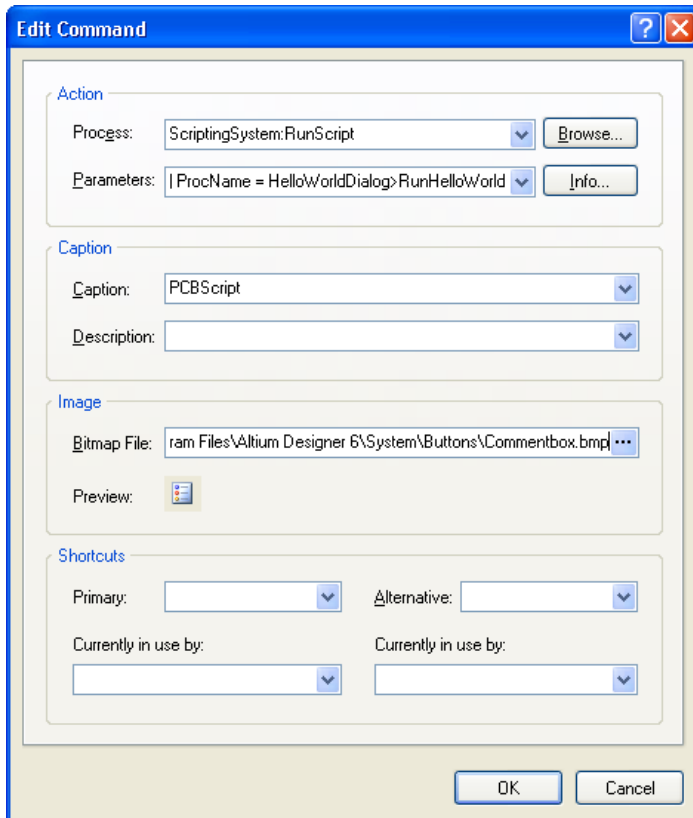
There are two parameters in this case: the `ProjectName` and the `ProcName`. The `ProjectName` parameter will have the full path of the project the script resides in. For the `ProcName` parameter, you need to specify the script filename and the main procedure in this script.

The format is as follows: `ProcName = ScriptFileName > ProcedureName`. Note the **GreaterThan (>)** symbol used between the script file name and the procedure name.

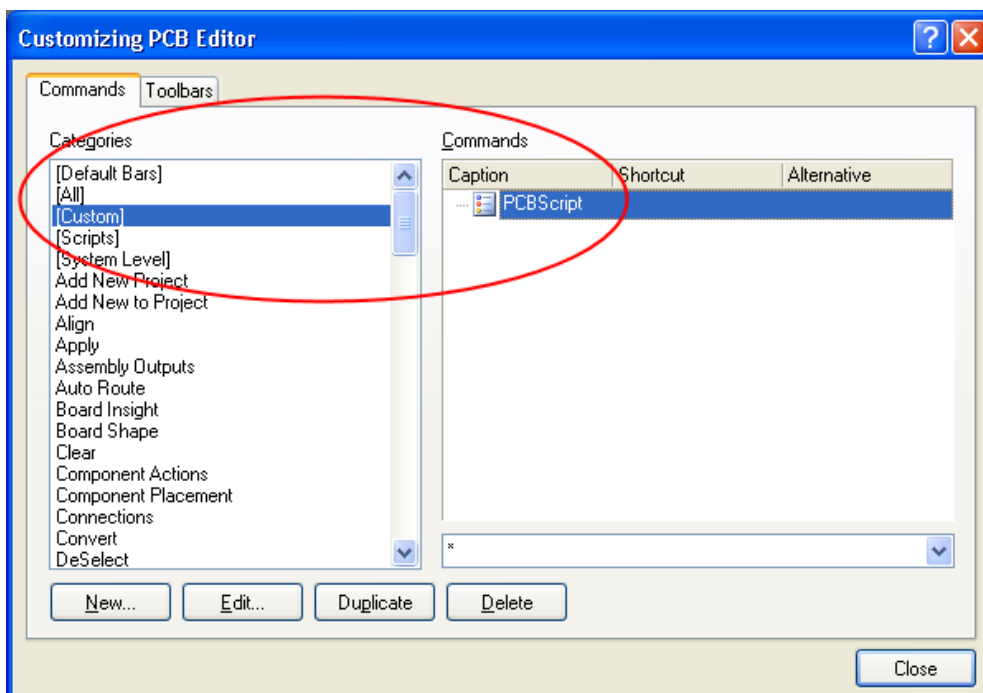
1. Make sure you have a PCB document open in Altium Designer first. Double click on the PCB menu or click on **DXP » Customize** menu and then the *Customizing PCB Editor* dialog appears.



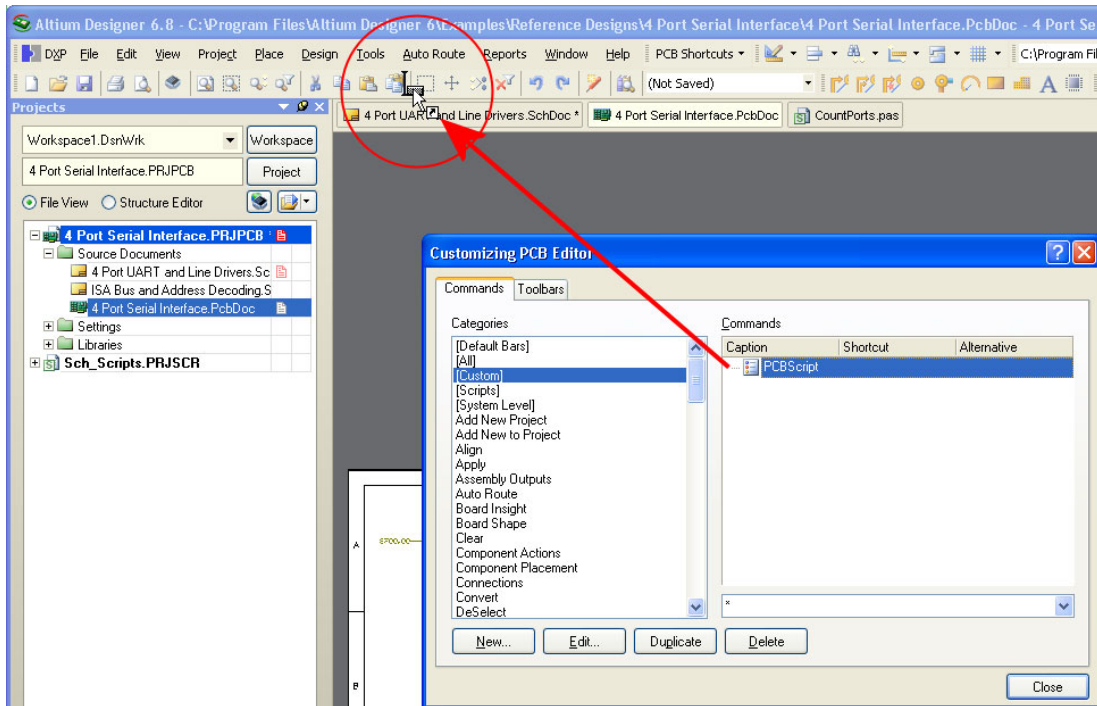
2. Click the **New** button of the *Customizing PCB Editor* dialog.
3. Choose `ScriptingSystem:RunScript` process in the **Process** field of the *Edit Command* dialog.
4. Enter `ProjectName = C:\Program Files\Altium Designer\Examples\Scripts\DelphiScript Scripts\General\HelloWorld.PrjScr | ProcName = HelloWorldDialog>RunHelloWorld` text in the **Parameters** field of the *Edit Command* dialog. This points to the `HelloWorld` dialog script. The exact path of the `ProjectName` parameter depends on the location of the Altium Designer installation.



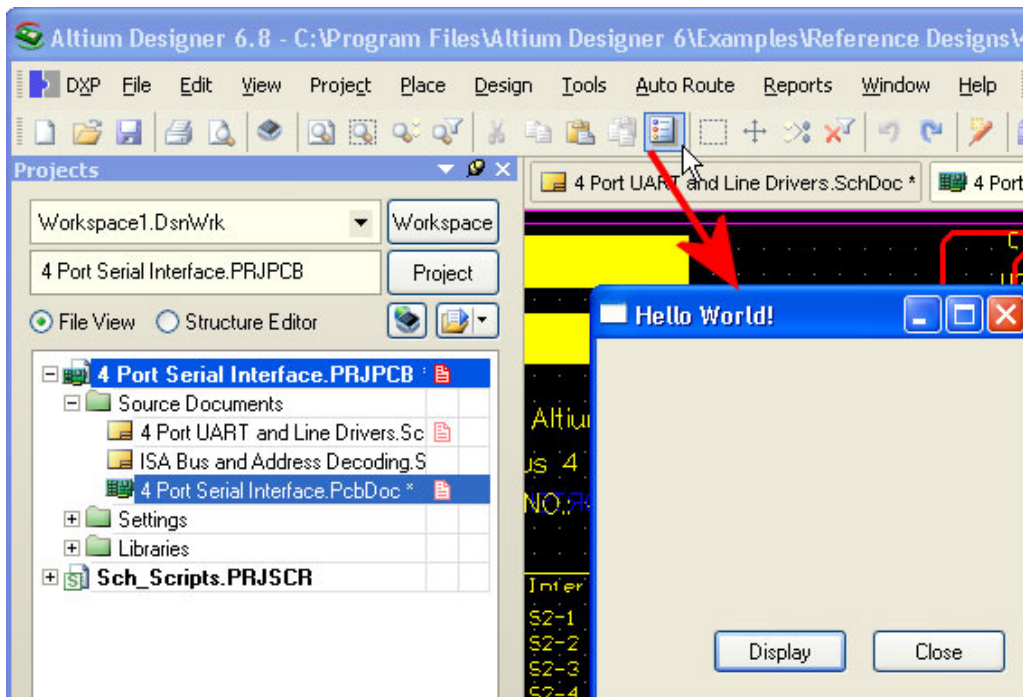
5. You will need to give a name to this new command in the **Caption** field. You can also enter text for the Description field and assign a new button image to the **BitMap File** field in the *Edit Command* dialog if you wish. In this case, the name is `PCBScript` in the **Caption** field of this dialog as shown above.
6. 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.



7. You then need to drag and drop the new **PCBScript** command onto the **PCB** menu from the *Customizing PCB Editor* dialog. The command appears on the menu as shown below.



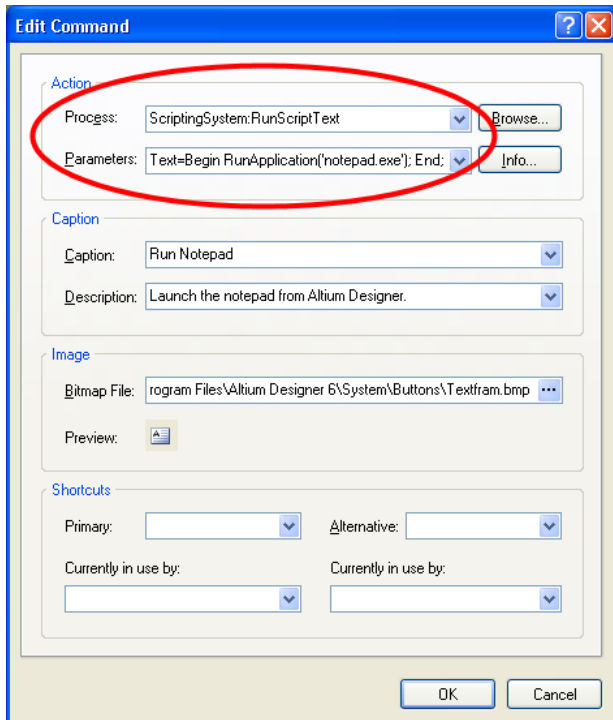
8. You can then click on this new command and the HelloWorld dialog appears.



Launching an External Application from Altium Designer

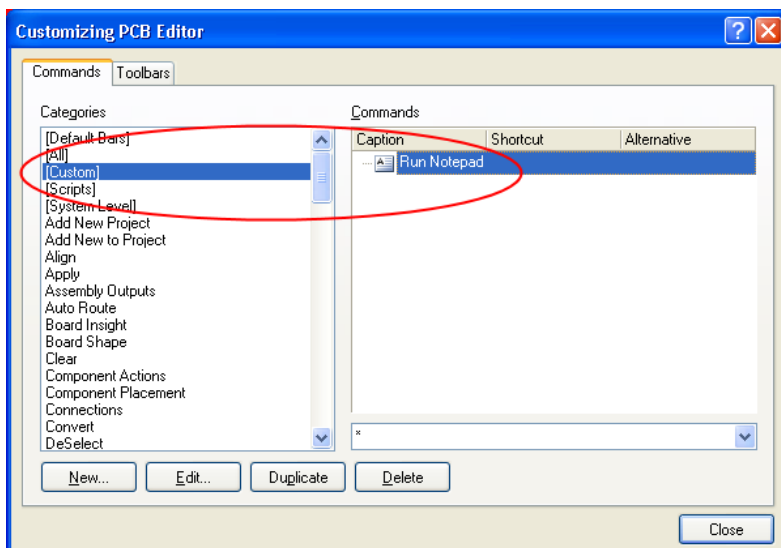
You can launch an external application such as the Microsoft Notepad from Altium Designer through software control. To do this, you will need the `ScriptingSystem:RunScriptText` process in the **Process** field and the `Begin RunApplication End;` string in the **Parameters** field of *Edit Command* dialog from the *Customizing [ServerName] Editor* dialog.

1. To show you how, you will learn how to launch a Microsoft Notepad from the PCB Editor in Altium Designer. Make sure you have a PCB document open in Altium Designer first. Invoke the *Customizing PCB Editor* dialog from the **DXP » Customize** command and click on the **New** button.
2. The *Edit Command* dialog then shows up. Insert `ScriptingSystem:RunScriptText` in the **Process** field, and insert `Text=Begin RunApplication('notepad.exe'); End;` in the **Parameters** field.



Note you would normally need to specify the full path and filename of this external application as a parameter for the `RunApplication` method. You will need to have the single quotes around the path and filename because of the white spaces in the path. Window applications such as the Calculator and the Notepad have their paths already specified by the operating system.

3. You need to enter the caption value for the **Caption** field for example; `Run Notepad` and you can assign a bitmap in the **Bitmap File** field. Click **OK** and this command appears in the **[Commands]** item within the **Categories** list of the *Customizing PCB Editor* dialog.



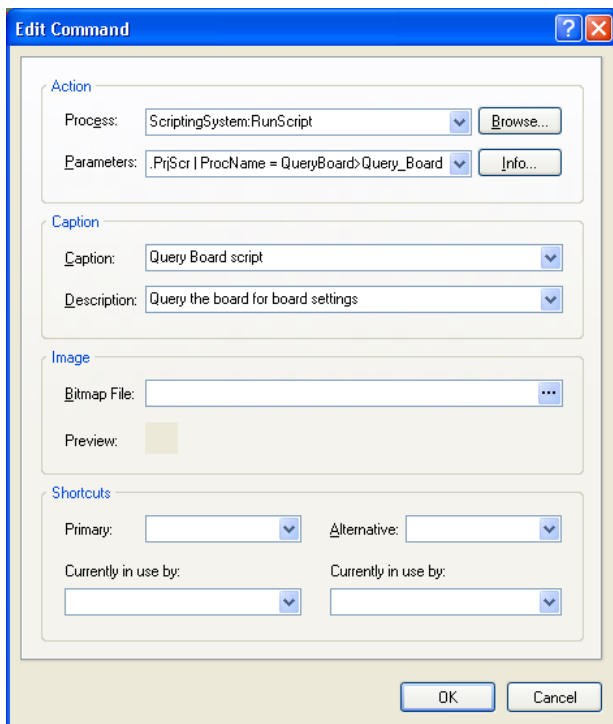
4. You need to drag and drop the new **Run Notepad** command onto the **PCB** menu from the *Customizing PCB Editor* dialog. The new command then appears on the menu.

Executing a Script from a popup menu

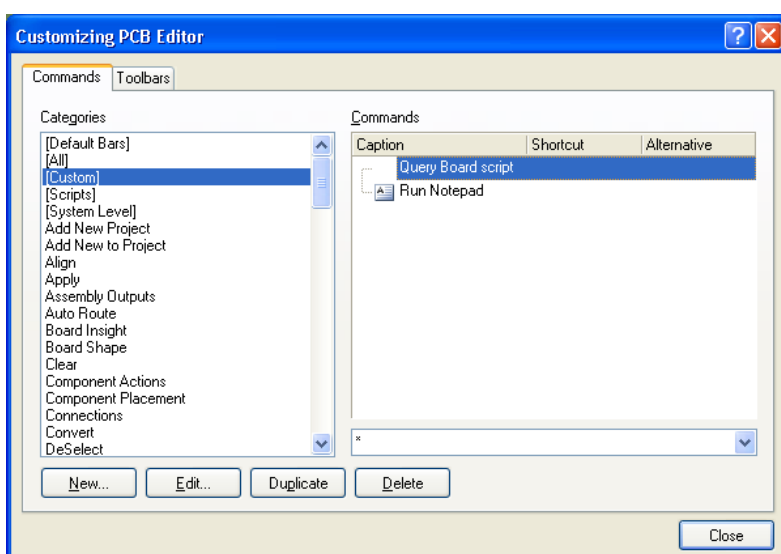
You can assign a script to a popup menu in one of the editors in Altium Designer. For example, you can assign a script to a popup menu in the PCB editor. To do this, invoke the *Customizing PCB Editor* dialog from the **DXP » Customize** menu and then open the *Edit Command* dialog.

Add the `ScriptingSystem:RunScriptText` process in the **Process** field and then specify the parameters for this process in the **Parameters** field in this *Edit Command* dialog.

1. As an example, to execute the QueryBoard script from a popup menu in the PCB editor, double click on the PCB menu bar, and then with the *Customizing PCB Editor* dialog open, click the **New** button to invoke the *Edit Command* dialog. Again make sure there is a PCB document open first (any other document kind will do but the *Customizing [ServerName] Editor* dialog will reflect the document kind).
2. Insert `ScriptingSystem:RunScript` in the **Process:** field, and insert `ProjectName = C:\Program Files\Altium Designer\Examples\Scripts\DelphiScript Scripts\PCB\PCB_Scripts.PrjScr | ProcName = QueryBoard>Query_Board` in the **Parameters:** field.

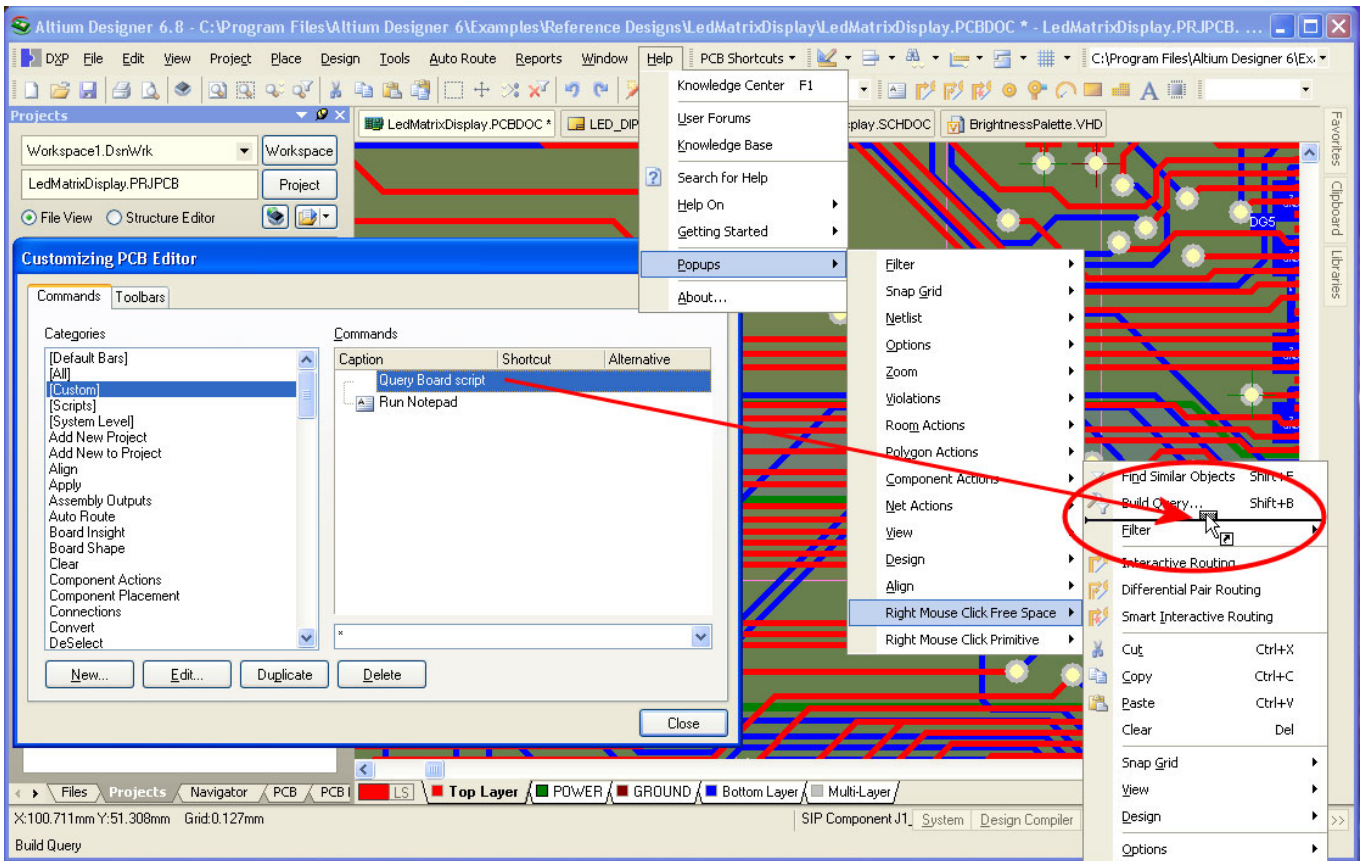


3. Enter `Query Board script` for the **Caption** field and `Query the board for board settings` for the **Description** field. Click **OK** button and this command appears in the **[Commands]** item within the **Categories** list of the *Customizing PCB Editor* dialog.

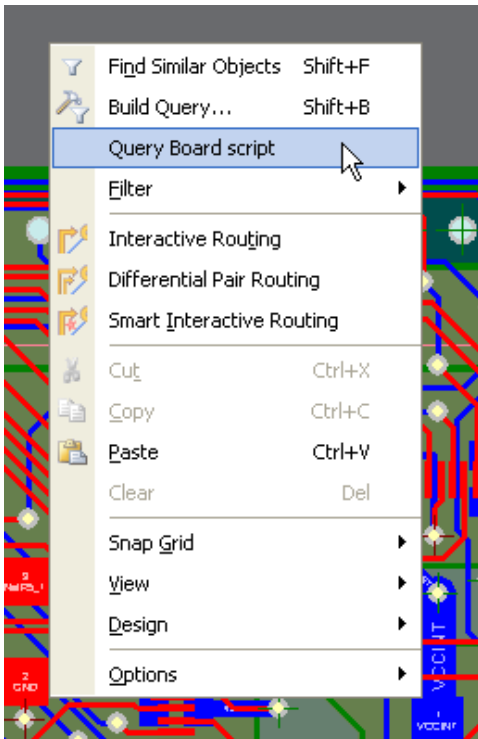


4. Drag this new command (and the cursor changes to a different cursor) from the list of **Commands** on the *Customizing PCB Editor* dialog to the **Help** menu on the PCB menu (don't release the left mouse button), once the **Help** menu appears, drag

to the **Popups** submenu and again drag further to the **Right Mouse Click Free Space** or **Right Mouse Click Primitive** submenu. Drop the command somewhere on this submenu.



- Alternatively drop the command in the **Right Mouse Click Free Space** or **Right Mouse Click Primitive** entry within the **Categories** section of the *Customizing PCB Editor* dialog.



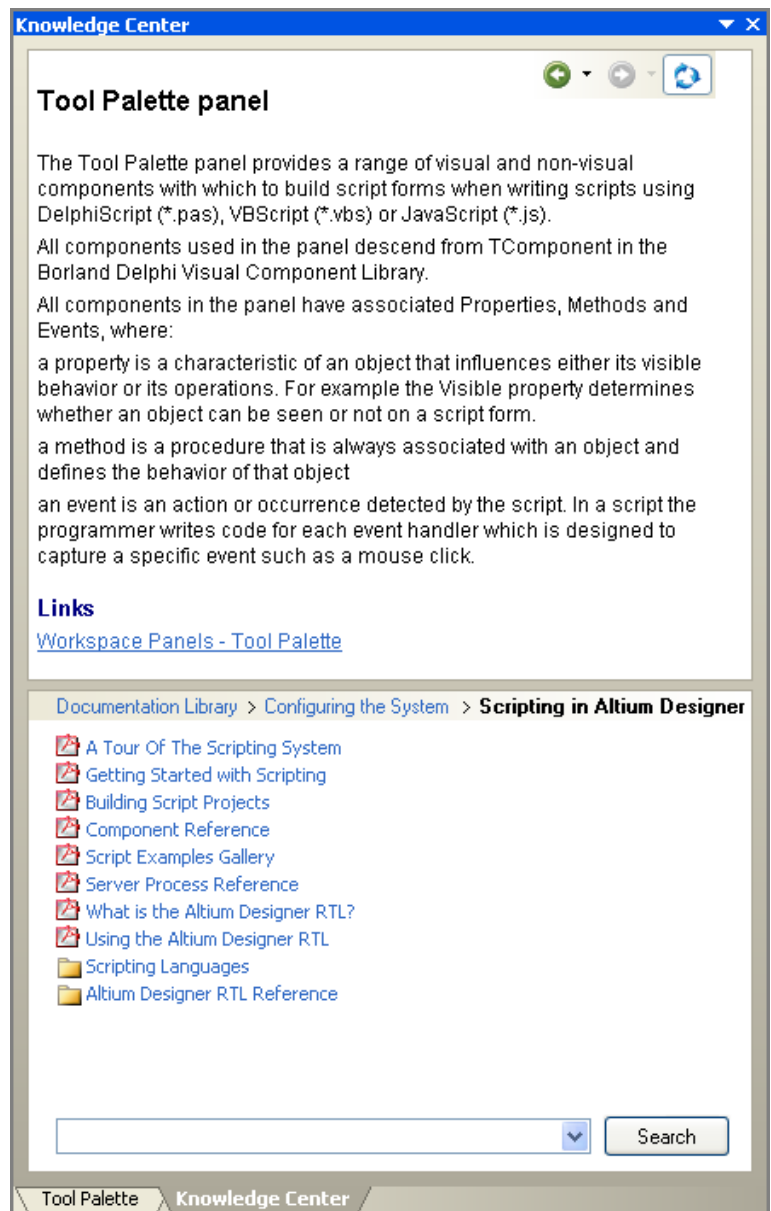
Conclusion

You have learnt how to create Script Units and Script Forms and save them in a Script Project.

You have seen that you can execute server processes and DXP object interfaces' methods and properties in a script.

Finally, you can explore various script examples in the \Examples\Scripts folder.

Consult the Scripting Resources from the Knowledge Center panel in Altium Designer as shown in the figure on the right.



Revision History

Date	Version No.	Revision
9-Mar-2004	1.1	New product release.
28-Oct-2004	1.2	Service Pack 2 release.
4-Apr-2005	1.3	Updated for Altium Designer
12-Dec-2005	1.4	Path references updated for Altium Designer 6
6-Dec-2007	1.5	Updated for Altium Designer 6.8
17-Mar-2008	1.6	Updated page size to A4.
20-Apr-2008	1.7	Path references updated for Altium Designer Summer 08
17-Mar-2011	-	Updated template.

Software, hardware, documentation and related materials:

Copyright © 2011 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment.

Altium, Altium Designer, Board Insight, DXP, Innovation Station, LiveDesign, NanoBoard, NanoTalk, OpenBus, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.