Adobe® Customer Support

Photoshop 4.0 OLE Automation Programming Guide

Adobe Photoshop 4.0 supports OLE Automation. With an *OLE automation controller*, like Microsoft's Visual Basic, Visual Basic for Applications, or Borland's Delphi, Adobe Photoshop can open and close documents and execute Action scripts.

Note: OLE Automation is only available on Windows 95 and Windows NT platforms. It is not available on Windows 3.1 or Macintosh. A similar external automation mechanism exists on the Macintosh using AppleScript.

OLE automation is intended for users savvy with scripting. Adobe does not provide Technical Support for troubleshooting or debugging scripts. For complete information on using OLE automation, see your Windows and OLE automation controller documentation.

Automation Basics

Photoshop 4.0 implements the "Actions" palette, permitting a user to record a sequence of actions and play them back.

The actions in the Actions palette can be triggered via OLE Automation. Once an Action has been recorded, it can be played back using OLE Automation in addition to interactively by pressing the *play* button in the Actions palette.

Automation Objects

Through an OLE automation controller, you can work with several Adobe Photoshop *automation objects*. By accessing properties and methods associated with different objects, you can make Photoshop open, close, and save documents, as well as run pre-recorded scripts. The Automation Objects are:

1. Application
2. Document
3. Actions Collection
4. Action

Application Object

Use the *application object* to start or quit Photoshop, to create a *document object*, or to run a action script by name.

Name	Type	Parameters	Description
Actions	Property	n/a	Returns an Actions Collection, which contains all the actions in the currently loaded Actions palette
FullName	Property	n/a	The full name of the application
Open	Method	BSTR	Opens a new document and returns a document object
PlayAction	Method	BSTR	Plays an action by name on the current document
Quit	Method	none	Quits Photoshop

Document Object

To create a *document object,* call *Open* from the *application object*. Document objects have the following attributes:

Name	Type	Parameters	Description
Activate	Method	None	Make this document the Active document (the default target for an Action script)
Close	Method	None	Saves changes and closes document
SaveTo	Method	BSTR	Save the file under a different name
Title	Property	n/a	The title (filename) of this document

Actions Collection Object

The *Actions Collection Object* represents all the *Actions* in the Actions palette. In addition to the following attributes, it also supports the Visual Basic's *For Each* automation control. The actions collection attributes are:

Name	Type	Parameters	Description
Count	Method	None	Returns the number of Actions in the Actions palette
Item	Method	Integer	Returns a particular Action object

Action Object

The *action object* is what the Actions collection object contains, and has the following properties:

Name	Type	Parameters	Description
Name	Property	n/a	The name (title) of this action
Play	Method	None	Play this action on the currently active document

Creating OLE Automation with Visual Basic

This section contains programming examples that show how to use Microsoft Visual Basic to access the OLE automation objects in Adobe Photoshop 4.0 for Windows.

Creating and destroying a Photoshop Application Object

Use Visual Basic's `CreateObject` procedure to make a Photoshop *Application Object*. The object can be deleted with the application object's *Quit* method, or by setting the object to *Nothing*.

```
Dim App as Object
Set App = CreateObject("Photoshop.Application")
App.Quit
```

Only one controller at a time can access Photoshop's automation. You'll get a message that the Photoshop object cannot be created if it is already in use from another application.

Opening and Closing Documents

The application object's *Open* method creates a new *document object*. It takes a file name (with path) as a parameter and returns a *document object*. Error messages are returned if:

1. If the file can't be open because it doesn't exist;
2. If the file is in an unrecognized format;
3. If Photoshop is in a modal state and can't process requests at this time.

These exceptions can be caught with Visual Basic's `On Error` statement.

To close a document and save any changes that have been made, use the document object's *Close* method.

```
Dim App as Object
Dim PhotoDoc as Object
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = App.Open("C:\files\photoshop\MyPicture.PSD")
PhotoDoc.Close
App.Quit
```

Running an Action Script by Name

Typically, you'll want to perform an action on the current document by executing a script from the palette. You can run a script by specifying its name, or you can access all the currently loaded scripts and run any or all of them by index or name.

To run an action by name, use the *PlayAction* method from the Application object. Adding to our previous example, we'll run an action called "BlurMe" on the active document. If you have more than one document object, activate one of them by calling its *Activate* method.

```
Dim App as Object
Dim PhotoDoc as Object
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = App.Open("C:\files\photoshop\MyPicture.PSD")
App.PlayAction("BlurMe")
PhotoDoc.Close
App.Quit
```

PlayAction returns a Boolean value that indicates whether the action was found and played or not. If the action doesn't exist, *PlayAction* will return `False`. If the Action cannot be played because Adobe Photoshop is in a modal state, this method will raise an error message that can be handled with Visual Basic's `On Error` statement.

Saving under a different name

To save the file under a different name, use the document object's SaveTo method to specify a name.

```
Dim App as Object
Dim PhotoDoc as Object
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = AppObject.Open("C:\files\photoshop\MyPicture.PSD")
App.PlayAction("BlurMe")
PhotoDoc.SaveTo("MyNewPicture.PSD")
PhotoDoc.Close
App.Quit
```

If you don't specify a fully qualified path name, the file will be saved relative to the directory of the original file. Fully qualified path names beginning with a backslash or drive letter are used as-is. If the file cannot be saved to the specified path, Adobe Photoshop will return a "Can't open file" error message.

Collection of Actions

The application object's *Actions* method returns a collection object that can be used to step through all the action objects in the currently loaded Actions palette. The following example steps through all the available Actions, and asks the user if he wants to run a particular Action. The name of a particular Action in the collection is obtained with the action object's *Name* method.

If an action's *Play* method cannot play the Action, it returns an "Unexpected" error message that can be caught with Visual Basic's `On Error` statement.

```
Dim App as Object
Dim PhotoDoc as Object
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = AppObject.Open("C:\files\photoshop\MyPicture.PSD")
For Each Action in App.Actions
     response = MsgBox(Action.Name, vbYesNo, "Run This Action")
     If response = vbYes Then
      Action.Play
     End If
Next
PhotoDoc.SaveTo("C:\MyNewPicture.PSD")
PhotoDoc.Close
App.Quit
```