Adobe

Adobe Photoshop ® 5.0

OLE Automation Programming Guide

Version 5.0.2 Release 1
December 1998

Most of the material for this document was derived from earlier works by Thomas Knoll, Mark Hamburg and Zalman Stern. Additional contributions came from David Corboy, Kevin Johnston, Sean Parent and Seetha Narayanan. It was then compiled and edited by Dave Wise and Paul Ferguson. It was later edited for content and updates by Andrew Coven and Thomas Ruark.

Version History

Date	Author	Status
7 November 1994	David J. Wise	First draft
15 January 1995	David J. Wise	First release
8 February 1995	Seetharaman Narayanan	MS-Windows modifications
16 July 1995	Paul D. Ferguson	Reformatted and updated for Photoshop 3.0.4
6 February 1996	Andrew Coven	Updated for Photoshop 3.0.5, Cross-application development
20 November 1996	Andrew Coven	Information, modules and callbacks updated for Photoshop 4.0.
19 March 1997	Andrew Coven	Bug fixes and typo updates for Photoshop 4.0.1.
18 April 1997	Andrew Coven	Release 1 for Photoshop 4.0.1.
1 June 1998	Andrew Coven	Release 1 for Photoshop 5.0.
2 December 1998	Thomas Ruark	Removed OLE chapter from the API Guide.

Title Page 1
Version History 2
1. OLE Automation 6
Automation basics. 7
Automation objects 7
Creating OLE Automation with Visual Basic 13
Creating and destroying an application object 13
Opening and closing documents 13
Running an action script by name 14
Iterating through a collection of actions 15
Making a descriptor and executing Actions 15
2. Example Project 16
Project Structure 17
Project components 17
Select Functions (PSSelect.bas). 18
Border() 18
ColorRange() 18
Contract(). 18
DeSelectPath() 18
DuplicatePath() 18
DuplicateSelectionAsChannel() 18
Expand(). 18
Feather() 18
Grow() 18
SelectAll(). 18
SelectBackgroundLayer() 18
SelectChannelByColor() 18
SelectChannelByName() 18
SelectDocumentByIndex(). 18
SelectDocumentByName() 19
SelectDocumentByOffset() 19
SelectInverse() 19
SelectLayerByIndex() 19
SelectLayerByName() 19
SelectNothing() 19
SelectPathByName() 19
SelectPolygon() 19
SelectPrevious() 19
SelectRectangle(). 19
SelectTransparentLayer() 19
SelectWorkPath() 19
Similar() 19

Smooth() 19
Layer Functions (PSLayer.bas) 20
CopyEffects() 20
DeleteLayer() 20
DisableLayerFX() 20
DuplicateLayerToNewDocument() 20
DuplicateLayer() 20
GlobalLightingAngle() 20
Group() 20
LayerUserMaskEnabled() 20
MakeNewLayer() 20
MergeLayers() 20
MoveLayer() 20
PasteEffects() 20
Ungroup() 20
Image Functions (PSImage.bas) 21
CamvasSize() 21
ConvertMode() 21
ConvertModeDepth() 21
Crop() 21
DuplicateDocument() 21
FlattenImage() 21
Flip() 21
ImageSize() 21
NewColorTable() 21
Rotate() 21
Filter Functions (PSFilter.bas) 22
AccentedEdges() 22
AddNoise() 22
BasRelief() 22
Blur() 22
BlurMore() 22
Clouds() 22
ColoredPencil() 22
ColorHalftone() 22
Craquelure() 22
DeInterlace() 22
Diffuse() 22
DiffuseGlow() 22
GaussianBlur() 22
HighPass() 23
Sharpen() 23
UnsharpMask() 23
File Functions (PSFile.bas) 24

CloseDocument() 24
ExportViaOutbound() 24
FileInfo() 24
MakeNewDocument() 24
OpenDocument() 24
OpenDocumentAs() 24
PlacePDFDocument() 24
SaveFile() 24
SaveFileAsBMP() 24
SaveFileAsEPS() 24
SaveFileAsFPX() 24
SaveFileAsIFF() 24
SaveFileAsJPG() 24
SaveFileAsPCt() 25
SaveFileAsPCX() 25
SaveFileAsPDF() 25
SaveFileAsPNG() 25
SaveFileAsPSD() 25
SaveFileAsPXR() 25
SaveFileAsRAW() 25
SaveFileAsSCT() 25
SaveFileAsTGA() 25
SaveFileAsTIF() 25
Edit Functions (PSEdit.bas) 26
Copy() 26
CopyMerged() 26
Cut() 26
DefinePattern() 26
Fill() 26
Paste() 26
PasteInto() 26
Purge() 26
Stroke() 26
Transform() 26
Photoshop Type Library 27
Loading the Type Library 27

1. OLE Automation

Adobe Photoshop 4.0 and 5.0 support OLE automation. With an *OLE automation controller*, like Microsoft's Visual Basic, Visual Basic for Applications, or Borland's Delphi, Adobe Photoshop 4.0 and 5.0 can open and close documents and execute Action scripts.

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.

Table 1-1: Adobe Photoshop OLE Automation version information

Photoshop version	OLE Automation features
4.0	Basic OLE Automation, including IActions.
5.0	Action control system, including IActionDescriptor, IActionControl, IActionList, IActionReference, PSConstants.

Automation basics

As of Adobe Photoshop 4.0, a new “Actions” palette exists, permitting a user to record a sequence of actions and play them back. See the chapter on Scripting for more information.

The actions in the actions palette are exposed 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

Several Adobe Photoshop *automation objects* can be instantiated from an OLE automation controller. 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 (`PhotoshopApplication`)
2. Document (`IAutoPSDoc`)
3. Actions Collection (`IActions`)
4. Action (`IAction`)

These automation objects are new since Photoshop 5.0:

5. Action Control (`IActionControl`)
6. Action Descriptors (`IActionDescriptor`)
7. Action Lists (`IActionList`)
8. Action References (`IActionReference`)
9. Photoshop Constants (`PSConstants`)

The Action objects, specifically `IActionControl`, `IActionDescriptor`, `IActionList`, and `IActionReference` parallel the Action suites defined in `PIAction.h`. The constants, `PSConstants`, parallel the predefined keys in `PITerminology.h`.

Application objects (PhotoshopApplication)

Use an *application object* to start or quit the host; create a document object, or run a script by name.

Table 1-2: Application object attributes

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	Exits the host.
These attributes are new since Adobe Photoshop 5.0:			
Visible	Property	n/a	True if the application isn’t hidden.

Table 1-2: Application object attributes (Continued)

Name	Type	Parameters	Description
MakeControlObject	Method	None	Creates a new IActionControl object.
MakeDescriptor	Method	None	Creates a new IActionDescriptor object.
MakeList	Method	None	Creates a new IActionList object.
MakeReference	Method	None	Creates a new IActionReference object.

Document objects (IAutoPSDoc)

Document objects are instantiated by calling `Open` from the application object.

Table 1-3: Document object attributes

Name	Type	Parameters	Description
Activate	Method	None	Make this document the active document and default target.
Close	Method	None	Save changes and close document.
SaveTo	Method	BSTR	Save the document under a different name.
Title	Property	n/a	The title (filename) of this document.

Actions Collection object (IActions)

The *actions collection object* represents all the scripts currently loaded in the Actions palette. In addition to the attributes in table 1-4, it also supports the *For Each* construct in Visual Basic automation controllers.

Table 1-4: Actions Collection object attributes

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

Action objects (IAction)

Action objects are the individual scripts in the Actions palette.

Table 1-5: Action object attributes

Name	Type	Parameters	Description
Name	Property	n/a	The name (title) of this script.
Play	Method	None	Play a script.

Action Control objects (IActionControl)

Action control objects are used to dispatch events to the host and get action-related property data. For more information on these specific functions, refer to the Action Control Suite in `PIActions.h`. This automation object is new since Photoshop 5.0.

Table 1-6: Action Control object attributes

Name	Type	Parameters	Description
GetActionProperty	Method	Reference, Descriptor	Returns the property referred to in the reference.
Play	Method	Event ID, Descriptor, dialogOptions	Play a specific event with the parameters in descriptor, and display the dialog according to dialogOptions.
StringIDToTypeID	Method	StringID, TypeID	Looks up a UUID for an event and returns its runtime EventID.
TypeIDToStringID	Method	TypeID, StringID	Looks up the runtime EventID and returns a unique string.

Action Descriptor objects (IActionDescriptor)

Action descriptor objects are used to build and manipulate Action descriptors, which are containers that hold the parameters for an event. This automation object is new since Photoshop 5.0.

Table 1-7: Action Descriptor object attributes

Name	Type	Parameters	Description
Clear	Method	None	Clear a descriptor.
Erase	Method	Key	Erase key from descriptor.
GetBoolean	Method	Key	Returns true / false for key.
GetClass	Method	Key	Returns class ID for key.
GetCount	Method	None	Returns number of keys in descriptor.
GetDouble	Method	Key	Returns double for key.
GetEnumerated	Method	Key	Returns enum type and value for key.
GetGlobalClass	Method	Key	Returns the class ID of a globally scoped key.
GetGlobalObject	Method	Key	Returns descriptor object for a globally scoped key.
GetInteger	Method	Key	Returns integer for key.
GetKey	Method	Index	Returns key for an index.
GetList	Method	Key	Returns Actions List for key.
GetObject	Method	Key	Returns object for key.
GetPath	Method	Key	Returns string path for key.
GetReference	Method	Key	Returns Action Reference for key.
GetString	Method	Key	Returns string for key.
GetType	Method	Key	Returns type ID for key.
GetUnitDouble	Method	Key	Returns unit ID and double value for key.
HasKey	Method	Key	Returns true if key in descriptor.
IsEqual	Method	Descriptor1, Descriptor2	Returns true if descriptors match.
PutBoolean	Method	Key, Boolean	Appends key as boolean value.
PutClass	Method	Key, ClassID	Appends key as class ID.
PutDouble	Method	Key, Double	Appends key as double value.
PutEnumerated	Method	Key, TypeID, Value	Appends key as enumerated ID with value.

Table 1-7: Action Descriptor object attributes (Continued)

Name	Type	Parameters	Description
PutGlobalClass	Method	Key, ClassID	Appends key as globally scoped value.
PutGlobalObject	Method	Key, ClassID, Descriptor	Appends key as global class with descriptor.
PutInteger	Method	Key, Integer	Appends key as integer.
PutList	Method	Key, List	Appends key as Actions List.
PutObject	Method	Key, ClassID, Descriptor	Appends key as class with descriptor.
PutPath	Method	Key, String	Appends key as string path.
PutReference	Method	Key, Reference	Appends key as reference.
PutString	Method	Key, String	Appends key as string.
PutUnitDouble	Method	Key, Units, Double	Appends key as specific unit double.

Action List objects (IActionList)

Action list objects are used to build and manipulate sets of Action Lists. Action Lists are contiguous groups of like items. This automation object is new since Photoshop 5.0.

Table 1-8: Action List object attributes

Name	Type	Parameters	Description
GetBoolean	Method	Index	Returns true / false for item.
GetClass	Method	Index	Returns class ID for item.
GetCount	Method	None	Returns number of items in list.
GetEnumerated	Method	Index	Returns enum type and value for item.
GetDouble	Method	Index	Returns double for item.
GetGlobalClass	Method	Index	Returns the class ID of a globally scoped item.
GetGlobalObject	Method	Index	Returns descriptor object for a globally scoped item.
GetInteger	Method	Index	Returns integer for item.
GetList	Method	Index	Returns Actions List for item.
GetObject	Method	Index	Returns object for item.
GetPath	Method	Index	Returns string path for item.
GetReference	Method	Index	Returns Action Reference for item.
GetString	Method	Index	Returns string for item.
GetType	Method	Index	Returns type ID for item.
GetUnitDouble	Method	Index	Returns unit ID and double value for item.
PutBoolean	Method	Boolean	Append item as boolean value.
PutClass	Method	ClassID	Append item as class ID.
PutFloat	Method	Double	Append item as float value.
PutEnumerated	Method	TypeID, Value	Append item as enumerated ID with value.
PutGlobalClass	Method	ClassID	Append item as globally scoped value.
PutGlobalObject	Method	ClassID, Descriptor	Append item as global class with descriptor.

Table 1-8: Action List object attributes (Continued)

Name	Type	Parameters	Description
PutInteger	Method	Integer	Append item as integer.
PutList	Method	List	Append item as Actions List.
PutObject	Method	ClassID, Descriptor	Append item as class with descriptor.
PutPath	Method	String	Append item as string path.
PutReference	Method	Reference	Append item as reference.
PutString	Method	String	Append item as string.
PutUnitDouble	Method	Units, Double	Append item as specific unit double.

Action Reference objects (IActionReference)

Action reference objects used to build and manipulate references to containers and properties of the host. The `Get()` and `Set()` routines refer to items inside the reference container itself. Do not mistake them for the properties of the object the reference specifies. Properties of the object specified by the reference are returned in a descriptor via the `Get()` method in `IActionControl`.

References must be created in order of most specific to least specific. The rule is to construct the reference as if an “of” exists between the objects. To build a reference to a channel of a layer of a document, you’d build it as “channel 1 of previous layer of document ‘foo’” — you may mix names, indexes, offsets, and ids in any one reference, but you only need to refer to any one element by one form. See `PIActions.h` for more info. This automation object is new since Photoshop 5.0.

Table 1-9: Action Reference object attributes

Name	Type	Parameters	Description
GetContainer	Method	None	Returns parent of reference.
GetDesiredClass	Method	None	Returns class in reference.
GetEnumerated	Method	None	Returns enum type and value of reference.
GetForm	Method	None	Returns form type of reference.
GetIndex	Method	None	Returns index value in reference.
GetName	Method	None	Returns C-string name in reference.
GetOffset	Method	None	Returns offset value in reference.
GetProperty	Method	None	Returns property value in reference.
GetIdentifier	Method	None	Returns identifier value in reference.
PutClass	Method	ClassID	Appends a class id.
PutEnumerated	Method	ClassID, TypeID, Value	Appends an enumerated type and value for a desired class.
PutIndex	Method	ClassID, Index	Appends an index for a desired class.
PutName	Method	ClassID, String	Append a string name for a desired class.
PutOffset	Method	ClassID, Value	Append an offset for a desired class.
PutProperty	Method	ClassID, Key	Append a property key for a desired class.
PutIdentifier	Method	ClassID, Value	Append a unique ID for a desired class.

Photoshop Constant objects (PSConstants)

Photoshop constants objects is a set of constants declaring Photoshop's predefined classes, enumerations, types, units, and events. They are directly translated from `PITerminology.h`, which a couple minor changes. If the listing in `PITerminology.h` is:

```
#define eventGaussianBlur      'GsnB'
```

The listing in `PSConstants` will be:

```
phEventGaussianBlur   const phEventGaussianBlur = 1198747202 (&H47736E42)
```

Which is the decimal equivalent of the four character code. Use `phEventGaussianBlur` in your scripts. For other keys, you may look them up in `PITerminology.h` or the object browser under `PSConstants`, but only the the `PSConstants` (not the `PITerminology.h`) definitions will be valid in your scripts.

This automation object is new since Photoshop 5.0.

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 for Photoshop 4.0 and 5.0.

Creating and destroying an application object

Use Visual Basic's `CreateObject` procedure to instantiate a Photoshop application object. The object can be destroyed with the application object's `Quit` method, or by setting the object to `Nothing`, causing the reference count to decrement to zero.

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

Photoshop's automation class factory is a *single use* object that can only be used by one automation controller at a time. You'll get a message that the Photoshop object can't be created if it is already in use by 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. Exceptions are raised:

1. If the file can't be opened 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 PhotoshopApplication
Dim PhotoDoc as IAutoPSDoc
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 iterate among all the currently loaded scripts and run any or all of them.

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 instantiated, target one of them by calling its `Activate` method.

```
Dim App as PhotoshopApplication
Dim PhotoDoc as IAutoPSDoc
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` returns `FALSE`. If the action cannot be played because the host is in a modal state, this method will raise an exception 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 PhotoshopAppliation
Dim PhotoDoc as IAutoPSDoc
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = App.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 a drive letter are used as-is. If the file cannot be saved to the specified path, the host will raise a "Can't open file" exception.

Iterating through a collection of actions

The applciation object’s `Actions` method returns a collection object that can be used to step through all the action objects currently loaded in the palette. The following example steps through all the available actions, asking the user to run a particular script. The name of an individual action in the collection is obtained through the action object’s `Name` method.

If an action’s `Play` method cannot play the script, it raises an “Unexpected” exception that can be caught with Visual Basic’s `On Error` statement.

```
Dim App as PhotoshopApplication
Dim PhotoDoc as IAutoPSDoc
Set App = CreateObject("Photoshop.Application")
Set PhotoDoc = App.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
PhotoDoc.SaveTo("MyNewPicture.PSD")
PhotoDoc.Close
App.Quit
```

Making a descriptor and executing Actions

The application object’s `MakeControlObject` method returns an Actions Control object that can be used to with the application object’s `MakeDescriptor` to create a descriptor and execute an event with the parameters you’ve put into the descriptor.

This snippet creates each of those objects, opens a couple files and executes two different filters on the second file.

```
Dim App as PhotoshopApplication
Dim Desc as IActionDescriptor
Dim Result as IActionDescriptor
Dim Control As Object

Set App = CreateObject("Photoshop.Application")

Set Control = App.MakeControlObject
Set Desc = App.MakeDescriptor

On Error Resume Next
AppActivate (“Adobe Photoshop”)
   If Err <> 0 Then
      MsgBox “Photoshop needs to be running.”
      Exit Sub
   End If

Desc.PutPath phKeyNull, "C:\PathToSomeFiles\File1.psd"
Set Result = Control.Play(phEventOpen, Desc, phDialogSilent)

Desc.PutPath phKeyNull, "C:\PathToSomeFiles\File2.psd"
Set Result = Control.Play(phEventOpen, Desc, phDialogSilent)

Desc.PutDouble phKeyRadius, 10.2
Set Result = Control.Play(phEventGaussianBlur, Desc, phDialogSilent)

Set Result = Control.Play(phEventFindEdges, Desc, phDialogSilent)
```

2. Example Project

The 5.0.2 SDK ships with a VB Automation project. This chapter describes the different components of the project. There are over 200 events that ship with Photoshop 5.0 and this project gives wrapper routines for a small percentage of these events. With the use of the Photoshop 5.0 Actions Guide the other events can be constructed.

Project Structure

The VBAutoPhoto project consists of one project file, two form files, eight module files, and one precompiled executable.

⚠ The precompiled executable, VBAutoPhoto.exe, will not run unless Visual Basic 5.0 or higher is installed.

Project components

Forms

1. VBAutoPhoto (VBAutoPhoto.frm) The main form for the project.
2. PropertiesForm (PropertiesForm.frm) The form displayed to gather information about Photoshop's current state.

Moduels

1. Edit (PSEdit.bas) Actions that come from the Photoshop->Edit menu pull down.
2. File (PSFile.bas) Actions that come from the Photoshop->File menu pull down.
3. Filter (PSFilter.bas) Actions that come from the Photoshop->Filter menu pull down.
4. Image (PSImage.bas) Actions that come from the Photoshop->Image menu pull down.
5. Layer (PSLayer.bas) Actions that come from the Photoshop->Layer menu pull down.
6. Main (VBAutoPhoto.bas) Action routines that are performed from the "Run this event" combo box pull down.
7. Select (PSSelect.bas) Actions that come from the Photoshop->Select menu pull down. Action routines for selecting objects: Documents, Layers, Channels, Paths in Photoshop.
8. Utilities (PSUtilities.bas) Miscellaneous utility routines.

Select Functions (PSSelect.bas)

These functions are wrappers for the events found in the Select menu pull down from within Photoshop. All of these routines will return 0 for no error. There are extra routines here for selecting another document, selecting layers in the document, selecting channels in the document, and selecting paths in the document.

Border()

```
Function Border(Width As Double) As Long
```

ColorRange()

```
Function ColorRange(Fuzziness As Long, MinL As Double, MinA As Double,
MinB As Double, MaxL As Double, MaxA As Double, MaxB As Double) As Long
```

Contract()

```
Function Contract(Pixels As Long) As Long
```

DeSelectPath()

```
Function DeSelectPath() As Long
```

DuplicatePath()

```
Function DuplicatePath(Name As String) As Long
```

DuplicateSelectionAsChannel()

```
Function DuplicateSelectionAsChannel(Name As String) As Long
```

Expand()

```
Function Expand(Pixels As Long) As Long
```

Feather()

```
Function Feather(Radius As Double) As Long
```

Grow()

```
Function Grow(Tolerance As Long, AntiAlias As Boolean) As Long
```

SelectAll()

```
Function SelectAll() As Long
```

SelectBackgroundLayer()

```
Function SelectBackgroundLayer() As Long
```

SelectChannelByColor()

```
Function SelectChannelByColor(Color As Long) As Long
```

SelectChannelByName()

```
Function SelectChannelByName(ChannelName As String) As Long
```

SelectDocumentByIndex()

```
Function SelectDocumentByIndex(Index As Long) As Long
```

SelectDocumentByName()

```
Function SelectDocumentByName(DocumentName As String) As Long
```

SelectDocumentByOffset()

```
Function SelectDocumentByOffset(Offset As Long) As Long
```

SelectInverse()

```
Function SelectInverse() As Long
```

SelectLayerByIndex()

```
Function SelectLayerByIndex(LayerIndex As Long) As Long
```

SelectLayerByName()

```
Function SelectLayerByName(LayerName As String) As Long
```

SelectNothing()

```
Function SelectNothing() As Long
```

SelectPathByName()

```
Function SelectPathByName(PathName As String) As Long
```

SelectPolygon()

```
Function SelectPolygon(EventValue As Long, Horizontal() As Double,
Vertical() As Double, PointCount As Long, antialias As Boolean) As Long
```

SelectPrevious()

```
Function SelectPrevious() As Long
```

SelectRectangle()

```
Function SelectRectangle(Top As Double, Left As Double, Bottom As Double,
Right As Double) As Long
```

SelectTransparentLayer()

```
Function SelectTransparentLayer(Invert As Boolean) As Long
```

SelectWorkPath()

```
Function SelectWorkPath() As Long
```

Similar()

```
Function Similar(Tolerance As Long, AntiAlias As Boolean) As Long
```

Smooth()

```
Function Smooth(Radius As Double) As Long
```

Layer Functions (PSLayer.bas)

These functions are wrappers for the events found in the Layer menu pull down from within Photoshop. All of these routines will return 0 for no error.

CopyEffects()

```
Function CopyEffects() As Long
```

DeleteLayer()

```
Function DeleteLayer() As Long
```

DisableLayerFX()

```
Function DisableLayerFX() As Long
```

DuplicateLayerToNewDocument()

```
Function DuplicateLayerToNewDocument(DocumentName As String, LayerName As
String) As Long
```

DuplicateLayer()

```
Function DuplicateLayer(LayerName As String) As Long
```

GlobalLightingAngle()

```
Function GlobalLightingAngle(Angle As Double) As Long
```

Group()

```
Function Group() As Long
```

LayerUserMaskEnabled()

```
Function LayerUserMaskEnabled(Enabled As Boolean) As Long
```

MakeNewLayer()

```
Function MakeNewLayer(LayerName As String, Opacity As Double, BlendMode
As Long, Group As Boolean) As Long
```

MergeLayers()

```
Function MergeLayers() As Long
```

MoveLayer()

```
Function MoveLayer(NewPosition As Long) As Long
```

PasteEffects()

```
Function PasteEffects() As Long
```

Ungroup()

```
Function Ungroup() As Long
```

Image Functions (PSImage.bas)

These functions are wrappers for the events found in the Image menu pull down from within Photoshop. All of these routines will return 0 for no error.

CamvasSize()

```
Function CanvasSize(Width As Double, Height As Double, HorizontalLocation
As Long, VerticalLocation As Long) As Long
```

ConvertMode()

```
Function ConvertMode(NewMode As Long) As Long
```

ConvertModeDepth()

```
Function ConvertModeDepth(NewDepth As Long) As Long
```

Crop()

```
Function Crop() As Long
```

DuplicateDocument()

```
Function DuplicateDocument(NewName As String) As Long
```

FlattenImage()

```
Function FlattenImage() As Long
```

Flip()

```
Function Flip(Axis As Long) As Long
```

ImageSize()

```
Function ImageSize(Width As Double, Height As Double, ConstrainPropor-
tions As Boolean) As Long
```

NewColorTable()

```
Function NewColorTable(Red() As Double, Green() As Double, Blue() As
Double) As Long
```

Rotate()

```
Function Rotate(Angle As Double) As Long
```

Filter Functions (PSFilter.bas)

These functions are wrappers for the events found in the Filter menu pull down from within Photoshop. All of these routines will return 0 for no error.

AccentedEdges()

```
Function AccentedEdges(Width As Long, Brightness As Long, Smoothness As
Long) As Long
```

AddNoise()

```
Function AddNoise(Amount As Long, Distortion As Long, Monochromatic As
Boolean) As Long
```

BasRelief()

```
Function BasRelief(Detail As Long, Smoothness As Long, LightDirection As
Long) As Long
```

Blur()

```
Function Blur() As Long
```

BlurMore()

```
Function BlurMore() As Long
```

Clouds()

```
Function Clouds() As Long
```

ColoredPencil()

```
Function ColoredPencil(Width As Long, Pressure As Long, Brightness As
Long) As Long
```

ColorHalftone()

```
Function ColorHalftone(Radius As Long, Angle1 As Long, Angle2 As Long,
Angle3 As Long, Angle4 As Long) As Long
```

Craquelure()

```
Function Craquelure(Spacing As Long, Depth As Long, Brightness As Long)
As Long
```

DeInterlace()

```
Function DeInterlace(Eliminate As Long, Create As Long) As Long
```

Diffuse()

```
Function Diffuse(Mode As Long) As Long
```

DiffuseGlow()

```
Function DiffuseGlow(Graininess As Long, Glow As Long, Clear As Long) As
Long
```

GaussianBlur()

```
Function GaussianBlur(Radius As Double) As Long
```

HighPass()

```
Function HighPass(Radius As Double) As Long
```

Sharpen()

```
Function Sharpen() As Long
```

UnsharpMask()

```
Function UnsharpMask(Radius As Double, Amount As Long, Threshold As Long)
As Long
```

File Functions (PSFile.bas)

These functions are wrappers for the events found in the File menu pull down from within Photoshop. All of these routines will return 0 for no error.

CloseDocument()

```
Function CloseDocument() As Long
```

ExportViaOutbound()

```
Function ExportViaOutbound(FullPath As String) As Long
```

FileInfo()

```
Function FileInfo(key As Long, Value As Variant) As Long
```

MakeNewDocument()

```
Function MakeNewDocument(Name As String, Mode As Long, Width As Double,
Height As Double, Resolution As Double, Fill As Long) As Long
```

OpenDocument()

```
Function OpenDocument(FullPath As String) As Long
```

OpenDocumentAs()

```
Function OpenDocumentAs(FullPath As String, Format As Long) As Long
```

PlacePDFDocument()

```
Function PlacePDFDocument(FullPath As String, PageNumber As Long, Hori-
zontal As Double, Vertical As Double) As Long
```

SaveFile()

```
Function SaveFile() As Long
```

SaveFileAsBMP()

```
Function SaveFileAsBMP(FullPath As String, Depth As Long, Platform As
Long, Compression As Boolean) As Long
```

SaveFileAsEPS()

```
Function SaveFileAsEPS(FullPath As String, Preview As Long, Depth As
Long, Encoding As Long, HalftoneScreen As Boolean, Transfer As Boolean,
ColorManagement As Boolean) As Long
```

SaveFileAsFPX()

```
Function SaveFileAsFPX(FullPath As String, Compress As Boolean) As Long
```

SaveFileAsIFF()

```
Function SaveFileAsIFF(FullPath As String) As Long
```

SaveFileAsJPG()

```
Function SaveFileAsJPG(FullPath As String, Quality As Long) As Long
```

SaveFileAsPCt()

```
Function SaveFileAsPCT(FullPath As String, Resolution As Long) As Long
```

SaveFileAsPCX()

```
Function SaveFileAsPCX(FullPath As String) As Long
```

SaveFileAsPDF()

```
Function SaveFileAsPDF(FullPath As String, Encoding As Long, Quality As
Long) As Long
```

SaveFileAsPNG()

```
Function SaveFileAsPNG(FullPath As String, Interlace As Long, Filter As
Long) As Long
```

SaveFileAsPSD()

```
Function SaveFileAsPSD(FullPath As String) As Long
```

SaveFileAsPXR()

```
Function SaveFileAsPXR(FullPath As String) As Long
```

SaveFileAsRAW()

```
Function SaveFileAsRAW(FullPath As String, FileType As String, FileCre-
ator As String, Header As Long, Interleaved As Boolean) As Long
```

SaveFileAsSCT()

```
Function SaveFileAsSCT(FullPath As String) As Long
```

SaveFileAsTGA()

```
Function SaveFileAsTGA(FullPath As String, Depth As Long) As Long
```

SaveFileAsTIF()

```
Function SaveFileAsTIF(FullPath As String, ByteOrder As Long) As Long
```

Edit Functions (PSEdit.bas)

These functions are wrappers for the events found in the Edit menu pull down from within Photoshop. All of these routines will return 0 for no error.

Copy()

```
Function Copy() As Long
```

CopyMerged()

```
Function CopyMerged() As Long
```

Cut()

```
Function Cut() As Long
```

DefinePattern()

```
Function DefinePattern() As Long
```

Fill()

```
Function Fill(Using As Long, Opacity As Double, Mode As Long) As Long
```

Paste()

```
Function Paste() As Long
```

PasteInto()

```
Function PasteInto() As Long
```

Purge()

```
Function Purge(Item As Long) As Long
```

Stroke()

```
Function Stroke(Width As Long, Location As Long, Opacity As Double, Mode
As Long) As Long
```

Transform()

```
Function Transform(PositionHorizontal As Double, PositionVertical As
Double, OffsetHorizontal As Double, OffsetVertical As Double, SkewHori-
zontal As Double, SkewVertical As Double, UsingHorizontal As Double,
UsingVertical As Double, Relative As Boolean, ConstrainProportions As
Boolean, Angle As Double, Height As Double, Width As Double, FreeTrans-
formCenterState As Long) As Long
```

Photoshop Type Library

Visual Basic can display the classes, properties, methods, events and constants from object libraries. The Adobe Photoshop object library is in the TypeLibrary.tlb file found next to the Photoshop executable in your installed directory. Refer to the Visual Basic Help documentation for further information on the Object Browser.

Loading the Type Library

To load the Photoshop Type Library follow the prodecures below. This procedure works for Visual Basic 5.0. Check your documentation for loading the Type Library into other OLE Automation controllers.

1. From the menu select: Project->References
2. From the Available References scroll window find the Adobe Photoshop 5.0 Type Library. NOTE: Use the Browse button if the reference does not show up.
3. Click the checkbox to the left.
4. Click OK.

Verify from the Object Browser

1. From the menus select: View->Object Browser
2. From the top left combo box select: Photoshop TypeLibrary.
3. Click on any item in the Classes list, left pane, to view Members in the right plane.

A
Actions 7, 15
Activate 8, 14
AppleScript 6

C
Clear 9
Close 8, 13, 14, 15
Count 8
CreateObject 13, 14, 15

E
Erase 9

F
FullName 7

G
GetActionProperty 9
GetBoolean 9, 10
GetClass 9, 10
GetContainer 11
GetCount 9, 10
GetDesiredClass 11
GetDouble 9, 10
GetEnumerated 9, 10, 11
GetForm 11
GetGlobalClass 9, 10
GetGlobalObject 9, 10
GetIdentifier 11
GetIndex 11
GetInteger 9, 10
GetKey 9
GetList 9, 10
GetName 11
GetObject 9, 10
GetOffset 11
GetPath 9, 10
GetProperty 11
GetReference 9, 10
GetString 9, 10
GetType 9, 10
GetUnitDouble 9, 10

H
HasKey 9

I
IAction 8
IActionControl 8, 11
IActionDescriptor 9, 15
IActionList 10
IActionReference 11
IActions 8
IAutoPSDoc 8, 13, 14, 15
IsEqual 9
Item 8

M
MakeActionControl 8
MakeControlObject 8, 15
MakeDescriptor 8, 15
MakeList 8
MakeReference 8

N
Name 8, 15

O
Open 7, 13, 14, 15

P
PhotoshopAppliation 14
PhotoshopApplication 7, 13, 14, 15
Play 8, 9, 15
PlayAction 7, 14
PropertiesForm.frm 17
PSConstants 12
PSEdit.bas 17
PSFile.bas 17
PSFilter.bas 17
PSImage.bas 17
PSLayer.bas 17
PSSelect.bas 17
PSUtilities.bas 17
PutBoolean 9, 10
PutClass 9, 10, 11
PutDouble 9, 15
PutEnumerated 9, 10, 11
PutFloat 10
PutGlobalClass 10
PutGlobalObject 10
PutIdentifier 11
PutIndex 11
PutInteger 10, 11
PutList 10, 11
PutName 11
PutObject 10, 11
PutOffset 11
PutPath 10, 11, 15
PutProperty 11
PutReference 10, 11
PutString 10, 11
PutUnitDouble 10, 11

Q
Quit 7, 13, 14, 15

S
SaveTo 8, 14, 15
StringIDToTypeID 9

T
Title 8

V
VBAutoPhoto.bas 17

VBAutoPhoto.frm 17
Visible 7