The Image Object API

Many GUI types managed by the Presentation Server support images in one form or another; some have background images, some have glyphs, some have image lists and so on. However, many of these images have properties of their own, and in most cases these image properties are common to all.

Because of this most Presentation Server types that support images expose them as intrinsic "sub-objects", thorough a model we refer to as the "Image Object API". The image object's lifetime is managed by the system and it cannot be programmatically destroyed from Basic+, but it can be manipulated by the Get/Set_Property and Exec_Method functions just like any other Presentation Server object.

The entire Image API is described in this section but be aware that not all image objects support all of the API. For example, a PUSHBUTTON type does not support an "INDEX" property for its background image, and its SplitGlyph image doesn't support an ALIGN property: All such exceptions are documented in the relevant sections that describe each object type.

Supported image formats

The presentation server supports the following image formats via WIC:

- BMP
- GIF
- ICO
- JPEG
- PNG
- TIFF

Image Object API Properties

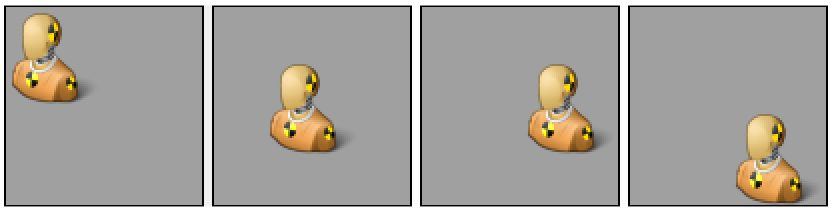
These properties apply to most IMAGE objects unless noted otherwise.

Name	Description
ALIGN	Specifies the horizontal and vertical alignment of the image within its parent.
AUTOSCALE	Specifies if the image should be scaled along with its parent object.
COLORKEY	Specifies the color in the image that should be treated as the "transparent color".
COUNT	Specifies the number of sub-images within an image file.
FILENAME	Returns the name of the image file being displayed.
FILENAMES	Specifies an array of DPI-specific image files to display.
FRAMECOUNT	Returns the number of "frames" within an image.
FRAMEDELAY	Returns the delay time in milliseconds for a multi-frame image.
FRAMEINDEX	Specifies the frame to display within a multi-frame image.
INDEX	Specifies the index of the sub-image to display within a multi-image file.
OFFSET	Specifies the point within the image (not the object) to begin drawing from.
ORIGIN	Specifies the point within the object (not the image) to begin drawing from.
SIZE	Returns the width and height of the image in pixels.
STYLE	Specifies how the image is drawn into an object (Tiled, Stretched, Clipped or Scaled).
TRANSLUCENCY	Specifies the degree of transparency applied to an image when it is drawn.

ALIGN property

Description

Specifies the horizontal and vertical alignment of the image.



Property Value

A numeric value that describes the position of the image within its parent object:

Value	Description	Value	Description
0	Top-Left (the default value)	**5**	Center-Right
1	Top Center	**6**	Bottom-Left
2	Top-Right	**7**	Bottom-Center
3	Center-Left	**8**	Bottom-Right
4	Centered		

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

The ALIGN property is ignored if either of these conditions are met:

- If the ORIGIN property is set to a value other than (0,0)
- If the STYLE property is set to any value other than "Clip" (0).

Equated constants for the ALIGN property can be found in the PS_EQUATES insert record.

Example

```
// Set the alignment of the background image for the PNL_MAIN control
$Insert PS_Equates
Call Set_Property_Only( @Window : "PNL_MAIN.IMAGE", "ALIGN", |
                        PS_IA_BOTTOMRIGHT$ )
```

See also

Image ORIGIN property, Image STYLE property.

AUTOSCALE property

Description
Specifies if the image should be scaled along with its parent object.

Property Value
This property is a boolean value. If set to TRUE$ (the default value) then the image will be scaled along with its parent control when the underlying parent form's DPI or SCALEFACTOR property is changed.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks
N/a.

Example

```
$Insert Logical

// Example: Set the auto-scaling of the background image for the
// current window to FALSE$
Call Set_Property_Only( @Window : ".IMAGE", "AUTOSCALE", FALSE$ )
```

See also
DPI property, WINDOW SCALEFACTOR property, WINDOW SCALED event, Appendix K – High-DPI Programming.

COLORKEY property

Description
Specifies the color in the image that is considered to be the "transparent color".

Property Value
This property can be a standard RGB color value or one of the special "color-key" values listed below:

Value	Description
-1	Top-Left pixel contains the color to use for the color-key.
-2	Top-Right pixel contains the color to use for the color-key.
-3	Bottom-Left pixel contains the color to use for the color-key.
-4	Bottom-Right pixel contains the color to use for the color-key.
-5	No Color-Key (this is the default value)

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks
Earlier versions of OpenInsight have supported basic image transparency via a method called "color-keying", where a specific color in the image is nominated as the transparent color (by default the color of the top-left pixel). When the image is drawn to screen the pixels that match the transparent color are not rendered so the background pixels show through instead. Whilst this can be effective in simple cases, it is sub-optimal for images that have smooth curves and shadows as these tend to appear very pixelated.

A much better alternative is to use an image format that supports an "alpha channel" such as a PNG file. Basically this means that each pixel in the image has an extra byte that describes its transparency - a value of 0 means that the pixel is totally transparent, while a value of 255 means that the pixel is totally opaque, Values in between are used to calculate how the image pixel is combined with the background pixel when drawn, so that it gives the appearance of being translucent, allowing the background pixel to show through to some degree. This is a technique known as Alpha-blending and is supported fully in this version of OpenInsight.

Because of this you should only use color-keying on older format images like 24-bit bitmap images that do not have an alpha channel - using more modern formats like PNG files will give much better results.

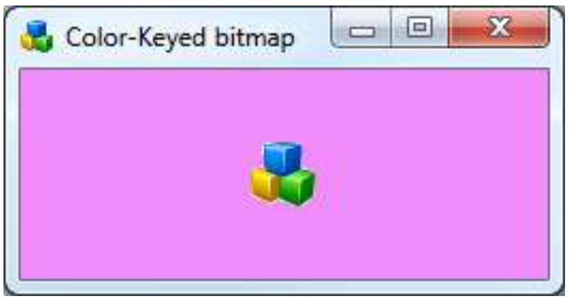
Equated constants for the COLORKEY property can be found in the PS_EQUATES insert record with the prefix "PS_TC_".

Example

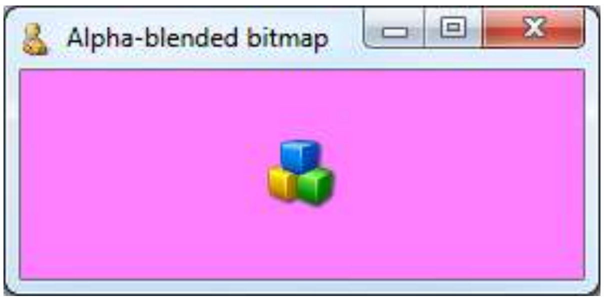
The 24-bit bitmap image below (i.e.no alpha-channel) has a white background. To draw it "transparently" onto a surface that is a different color we have to set the COLORKEY property. In this case we can hard-code it to White (RGB value of 0x00FFFFFF), or we can tell it to use one of the corner pixels instead.

```
$Insert PS_Equates

// Example 1: Set the ColorKey of the background image for the
// current window to use the top-left pixel of the image.

Call Set_Property_Only( @Window : ".IMAGE", "COLORKEY", PS_TC_TOPLEFT$ )
```

The code above gives the following results:



Note the pixelated edge due to the shadow in the image. A PNG file with an alpha-channel gives much better results however, as the shadow is blended correctly:



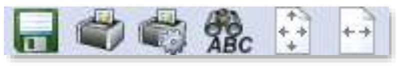
See also

N/a.

COUNT property

Description

Specifies the number of sub- images in an image file. For an object that can support many images (like a TOOLBAR), or several image states (like a PUSHBUTTON), it is often better to have a single file containing all the required images arranged in a horizontal strip rather than loading individual files. The INDEX property is then used to display the desired image. For example, the following image file would have a COUNT property of 6, and setting the INDEX property to 4 would display the "search" image:



Property Value

This property is an integer value, greater than 0.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Note that this property is *not* the same as the FRAMECOUNT property (see below). Frames are intrinsic to the image format itself, whereas those defined by the COUNT property are multiple sub-images combined in a file with a single frame (In effect this emulates a multi-frame file format). Each of these "sub-images" should have the same width.

Example

```
// Set the number of images in the IMAGE object and display the
// the 4th image

ObjxArray =        @Window : ".IMAGE"
PropArray =        "COUNT"
DataArray =        6

ObjxArray := @Rm : @Window : ".IMAGE"
PropArray := @Rm : "INDEX"
DataArray := @Rm : 4

Call Set_Property_Only( ObjxArray, PropArray, DataArray )
```

See also

Image FRAMECOUNT property, Image INDEX property.

FILENAME property

Description

When used with Get_Property this property returns the name of the file being displayed. When used with Set_Property it operates the same as the FILENAMES property.

Property Value

When used with Get_Property the value is the name of the selected image file name being displayed.

When used with Set_Property the value can be an @Fm-delimited array of file names as per the FILENAMES property.

Image file names can be in one of several formats:

- A path and file name of an image file.
- A path and file name to a resource file (such as a DLL) containing the image, along with its resource ID. The latter component is separated from the file name by a "#" character.

  E.g.

  ```
  .\res\MyAppRes.Dll#192
  .\res\MyAppRes.Dll#MYIMAGE
  ```

  Note that if the image is stored in a custom resource section (rather than the usual BITMAP section) the custom section name may be specified by inserting it before the resource name like so:

  ```
  .\res\MyAppRes.Dll#JPG#192
  .\res\MyAppRes.Dll#PNG#MYIMAGE
  ```

- The name of a "System Icon" (See Appendix J).
- The string "<memory> if the image has been set by the SETIMAGE method.
- The string "<HBITMAP>" if the image was set by the SETHBITMAP method.

(Note that the latter two formats cannot be used with Set_Property).

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

When a control has been given a set of multiple filenames to display, the Presentation Server selects the closest match based on the current scaling information for the parent form (See the FILENAMES property for more details).

Example

```
// Example - get the current file name being used be the current form's IMAGE object
ImgFile = Get_Property( @Window : ".IMAGE", "FILENAME" )
```

See also

Image FILENAMES property, WINDOW DPI property, WINDOW SCALEFACTOR property, Image SAVETOFILE method, Image SETHBITMAP method, Image SETIMAGE method, WINDOW SCALED event, Appendix J – System Icons, Appendix K – High-DPI Programming.

FILENAMES property

Description

Specifies an array of DPI-specific files to be displayed.

Property Value

This property is an @Fm-delimited array of file names used to display the image at different DPI/Scaled settings. The position of a filename in the array corresponds to the DPI value for which it should be used, as illustrated in the following table:

Position	Display DPI	Factor	Position	Display DPI	Factor
<1>	96	100%	**<7>**	288	300%
<2>	120	125%	**<8>**	336	350%
<3>	144	150%	**<9>**	384	400%
<4>	168	175%	**<10>**	432	450%
<5>	192	200%	**<11>**	480	500%
<6>	240	250%			

Image file names can be in one of several formats:

- A path and file name of an image file.
- A path and file name to a resource file (such as a DLL) containing the image, along with its resource ID. The latter component is separated from the file name by a "#" character.

  E.g.

  ```
  .\res\MyAppRes.Dll#192
  .\res\MyAppRes.Dll#MYIMAGE
  ```

  Note that if the image is stored in a custom resource section (rather than the usual BITMAP section) the custom section name may specified by inserting it before the resource name like so:

  ```
  .\res\MyAppRes.Dll#JPG#192
  .\res\MyAppRes.Dll#PNG#MYIMAGE
  ```

- The name of a "System Icon" (See Appendix J).
- The string "<memory> if the image has been set by the SETIMAGE method.
- The string "<HBITMAP>" if the image was set by the SETHBITMAP method.

(Note that the latter two formats cannot be used with Set_Property).

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

When the Presentation Server needs to display an image it first takes the product of the parent form's SCALEFACTOR property and the DPI of the monitor on which it is displayed to calculate a "required DPI". This value is then compared to the files in the FILENAMES property to find the closest fit, which is then rendered on the object in question. If an exact DPI match cannot be found the file with the next largest DPI is used because image quality is usually better when an image is resized smaller than when it is resized larger.

Example

```
// Example - set an array of files for an IMAGE object so that the system
// can pick one that looks better depending on the form's DPI and SCALEFACTOR.
//
// For a required DPI of 96 or less the object will select ImgFiles<1>
// For a required DPI between 144 (inc) and 96 the object will select ImgFiles<2>
// For a required DPI greater than 144 it will select ImgFiles<3>

ImgFiles    = ""
ImgFiles<1> = ".\images\test_dummy.png"     ; // 96DPI   (100%)
ImgFiles<3> = ".\images\test_dummy_150.png" ; // 144DPI  (150%)
ImgFiles<5> = ".\images\test_dummy_200.png" ; // 192DPI  (200%)

PrevImages = Set_Property( @Window : ".IMAGE", "FILENAMES", ImgFiles )
```

See also

Image FILENAME property, WINDOW DPI property, WINDOW SCALEFACTOR property, Image SAVETOFILE method, Image SETHBITMAP method, Image SETIMAGE method, WINDOW SCALED event, Appendix J – System Icons, Appendix K – High-DPI Programming.

FRAMECOUNT property

Description

Some image formats, such the animated GIF format, hold more than one image "internally", and each image therein is referred to as a "frame". This property returns the number of frames contained in such an image.

Property Value

If the IMAGE object has a valid image file loaded then this property is an integer value greater than 0. If a valid image is not loaded this property returns 0.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

Note that this property is *not* the same as the COUNT property. Frames are intrinsic to the image format itself, whereas those defined by the COUNT property are "multiple images" combined in a file within a single frame; i.e. the COUNT indicates how many "slices" to chop the image into and the INDEX specifies which slice to display (In effect this emulates a multi-frame file format).

Most common image file formats do not support more than one frame.

Example

The following example shows how to play an animated GIF file in a BITMAP control by using two events:

- A TIMER event for the BITMAP control that increments the FRAMEINDEX of the image:

```
Function TIMER( CtrlEntID, CtrlClassID )

   $Insert Logical

   FrameCount = Get_Property( CtrlEntID : ".IMAGE", "FRAMECOUNT" )
   FrameIndex = Get_Property( CtrlEntID : ".IMAGE", "FRAMEINDEX" )

   FrameIndex += 1

   If ( FrameIndex > FrameCount ) Then
      FrameIndex = 1
   End

   Call Set_Property_Only( CtrlEntID : ".IMAGE", "FRAMEINDEX", FrameIndex )

Return TRUE$
```

- A PUSHBUTTON CLICK event that starts the BITMAP TIMER event by setting its TIMER property to the same value as the image's FRAMEDELAY value:

```
Function CLICK( CtrlEntID, CtrlClassID )

   $Insert Logical

   FrameDelay = Get_Property( CtrlEntID : ".IMAGE", "FRAMEDELAY" )

   Call Set_Property_Only( CtrlEntID : ".IMAGE", "TIMER", FrameDelay )

Return TRUE$
```

See also

Image COUNT property, Image FRAMEDELAY property, Image FRAMEINDEX property, SYSTEM GETIMAGEINFO method.

FRAMEDELAY property

Description

Returns the "frame delay" for a multi-frame image. This is the number of milliseconds that should elapse between changing frames when the image is animated.

Property Value

This property is an integer value. This value always returns 0 if the image is not a multi-frame image.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

If the image has multiple frames then the property value always returns at least 90 milliseconds – this is a default value that matches the default on most browsers if the frame delay cannot be extracted from the image file for some reason.

Example

See the FRAMECOUNT property for an example of how to use the FRAMEDELAY when animating a GIF image file.

See also

Image FRAMECOUNT property, Image FRAMEINDEX property.

FRAMEINDEX property

Description
Specifies the frame to display in a multi-frame image.

Property Value
This property is an integer value. It cannot be greater that the image's FRAMECOUNT value.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get/Set	No	No	No

Remarks
N/a.

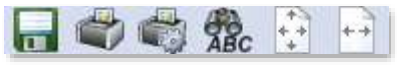
Example
See the FRAMECOUNT property for an example of how to use the FRAMEINDEX when animating a GIF image file.

See also
Image FRAMECOUNT property, Image FRAMEDELAY property.

INDEX property

Description

Specifies the image number to display in a multi-image file as described in the COUNT property. For example, the following image file would have a COUNT property of 6, and setting the INDEX property to 3 would display the "print setup" image:



Property Value

This property is an integer value. It cannot be greater that the image's COUNT value.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get/Set	No	No	No

Remarks

When the INDEX property is changed the system uses the image file's total width divided by its COUNT property to find the width of a "sub-image". It then uses this value, along with the COUNT property, to calculate an offset into the image where the desired sub-image is located.

Example

```
// Example - show the 3rd sub-image in a multi-image file, assuming that
// it has a COUNT of 6...

$Insert Logical

Call Set_Property_Only( CtrlEntID : ".IMAGE", "INDEX", 3 )
```

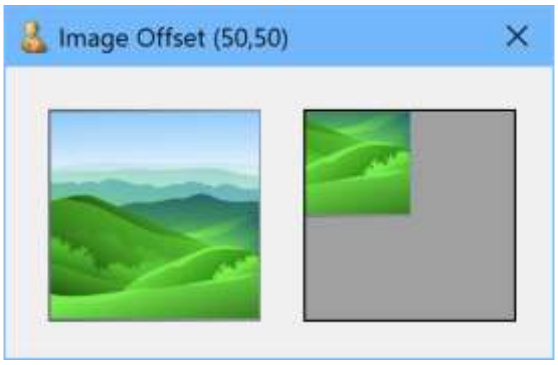
See also

Image COUNT property.

OFFSET property

Description

Specifies the point within the image (not the object) to begin drawing from.



Property Value

This property is an @Fm-delimited array of coordinates:

```
<1> Left coordinate
<2> Top coordinate
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	Yes	No

Remarks

This property can be used to scroll an image – e.g. repeatedly incrementing the Left coordinate will scroll the image to the left.

The OFFSET property only applies to images with the STYLE property set to "Clip".

Example

```
// Example - Set the OFFSET property of the image in the current control
// to begin drawing from 20,40

Call Set_Property( CtrlEntID : ".IMAGE", "OFFSET", 20 : @Fm : 40 )
```

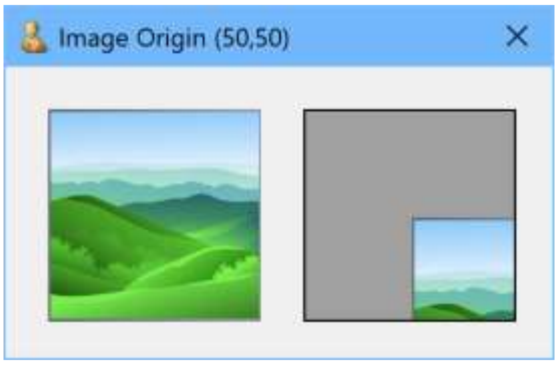
See also

Image ORIGIN property, IMAGE STYLE property.

ORIGIN property

Description

Specifies the point within the object (not the image) to begin drawing from.



Property Value

This property is an @Fm-delimited array of coordinates:

```
<1> Left coordinate
<2> Top coordinate
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	Yes	No

Remarks

The ORIGIN property only applies to images with the STYLE property set to "Clip".

Example

```
   // Example - Set the ORIGIN property of the image in the current control
   // to begin drawing from 50,50

   Call Set_Property( CtrlEntID : ".IMAGE", "ORIGIN", 50 : @Fm : 50 )
```

See also

Image OFFSET property, IMAGE STYLE property.

SIZE property

Description

Returns the dimension of the image in pixels (not DIPs).

Property Value

An @Fm-delimited array of image information formatted as follows:

```
<1> Image Width (pixels)
<2> Image Height (pixels)
```

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
N/a	Get	No	No	No

Remarks

N/a.

Example

```
// Example - get the size of the image from the current form's IMAGE sub-object
ImgSize = Get_Property( @Window : ".IMAGE", "SIZE" )
```

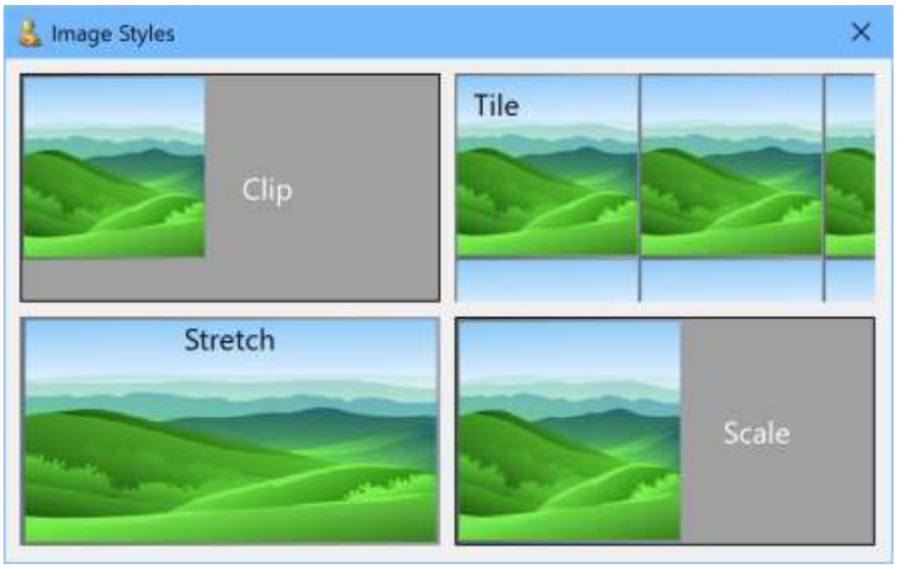
See also

SYSTEM GETIMAGEINFO method.

STYLE property

Description

Specifies the basic method for how the image is drawn onto the object.



Property Value

This property can be one of the following numeric values:

Value	Name	*Description*
0	Clip	This is the default style. Image is drawn directly onto the object without resizing. The position where is it drawn can be modified by the ORIGIN and ALIGN properties.
1	Tile	image is repeated along the X and Y axis until the object surface is fully covered.
2	Stretch	Image is resized along both the X and Y axis to fully fit the surface of the object.
3	Scale	Image is resized to fit either the X or Y axis (whichever is smaller), whilst keeping the image proportions the same.

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Equates constants for the Image STYLE property can be found in the PS_EQUATES insert record.

Example

```
// Example - Set the STYLE property of the image in the current control
// to "Tile"
$Insert PS_Equates

Call Set_Property_Only( CtrlEntID : ".IMAGE", "STYLE", PS_IS_TILE$ )
```

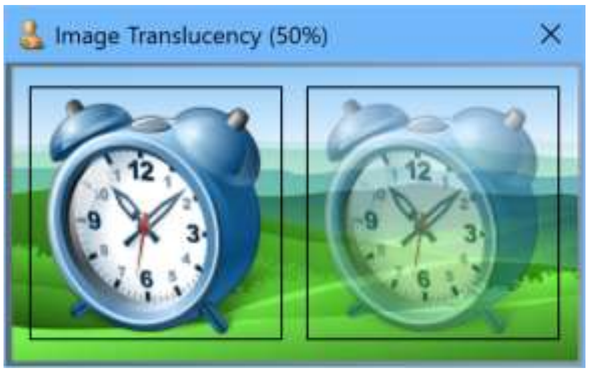
See also

Image ALIGN property, Image ORIGIN property.

TRANSLUCENCY property

Description

Specifies the degree of transparency applied to an image when it is drawn onto an object.



Property Value

This property is an integer value between 1 and 100 which represents the percentage of transparency applied to the image. A value of 0 means fully opaque, while a value of 100 means fully transparent (i.e. the image will not be drawn).

Property Traits

Development	*Runtime*	*Indexed*	*Scaled*	*Synthetic*
Get/Set	Get/Set	No	No	No

Remarks

Be aware that the use of translucency does involve extra overhead when drawing objects because there are the extra steps of painting in the object's background and then blending it with image itself – while the PS attempts to mitigate this using cached bitmaps and double-buffering there will always be *some* impact.

Example

```
// Set the TRANSLUCENCY of the current form's IMAGE to 60%

PrevVal = Set_Property( @Window : ".IMAGE", "TRANSLUCENCY", 60 )
```

See also

Common GUI Object TRANSLUCENCY property, WINDOW TRANSLUCENCY property.

Image Object API Methods

These methods apply to most IMAGE objects unless noted otherwise.

Name	Description
SAVETOFILE	Saves the current image to a file.
SETHBITMAP	Loads an image from a Windows BITMAP handle (HBITMAP).
SETIMAGE	Loads an image from an array of "raw" image bytes.
SETREPOSIMAGE	Specifies an image file using an OpenInsight repository IMAGE entity.

SAVETOFILE method

Description

Saves the current image to a file.

Syntax

```
SuccessFlag = Exec_Method( ObjImageID, "SAVETOFILE", FileName )
```

Parameters

Name	Required	Description
ObjImageID	Yes	Name of the image object to access. e.g. @Window : ".IMAGE"
FIleName	Yes	Path and filename to save the image as. The format of the saved image file is deduced from the FileName extension.

Returns

Returns TRUE$ if the image was saved successfully, or FALSE$ otherwise.

Remarks

This method uses the Bitmap Save method in the Windows GDI+ library. Please see the Microsoft website for more details.

Example

```
// Example – Save the contents of the current control's IMAGE object to a PNG file.

FileName = "C:\Temp\TestImage.png"
IsOK = Exec_Method( CtrlEntID : ".IMAGE", "SAVETOFILE", FileName )
```

See also

Image FILENAME property, Image FILENAMES property.

SETHBITMAP method

Description

Loads an image using a Windows GDI BITMAP handle (HBITMAP).

Syntax

```
SuccessFlag = Exec_Method( ObjImageID, "SETHBITMAP", hBitmap, hPalette, |
                           Options )
```

Parameters

Name	Required	Description
ObjImageID	Yes	Name of the image object to access. e.g. @Window : ".IMAGE"
hBitmap	Yes	GDI handle (HBITMAP) of the image to load.
hPalette	No	GDI handle (HPALETTE) of a palette to use when loading the image.

Options	No	Numeric values specifying how the alpha channel is handled:
		
		`0   Use alpha channel (the default)`
		`1   Use pre-multiplied alpha channel`
		`2   Ignore alpha channel`

Returns

Returns TRUE$ if the image was loaded successfully, or FALSE$ otherwise.

Remarks

This method copies the bitmap from the passed BITMAP handle when loading the image, so it is safe to release the HBITMAP after the method returns.

If an image is loaded via this method the object's FILENAME and FILENAMES properties will return the string "<HBITMAP>".

32-bit bitmaps produced with normal GDI functions (like the example below) should use the "Ignore alpha channel" option, otherwise they will appear transparent as GDI is not alpha channel aware. Equates constants for the Options parameter can be found in the PS_EQUATES insert record.

Automatic image selection for DPI scaling cannot be applied to the image when it is set using this method. Only the FILENAMES property supports this feature.

For more information on Windows GDI and BITMAP objects please consult the Microsoft website.

Example

```
   // Example - draw into a bitmap and display it in the BMP_TEST control
   Declare Function MSWin_GetDC, MSWin_ReleaseDC, MSWin_CreateCompatibleBitMap
   Declare Function MSWIn_CreateCompatibleDC, MSWin_SelectObject, MSWin_MoveToEx
   Declare Function MsWin_LineTo, MSWin_Rectangle, MSWin_Ellipse, MSWin_RoundRect
   $Insert PS_Equates
   $Insert Logical

   // Create a new bitmap in memory the same size as the current control
   CtrlSize = Get_Property( @Window : ".BMP_TEST", "CLIENTSIZE" )
   ClientX  = CtrlSize<1>
   ClientY  = CtrlSize<2>

   // Create the bitmap in memory, compatible with the control
   hwnd     = Get_Property( @Window : ".BMP_TEST", "HANDLE" )
   hdc      = MSWin_GetDC( Hwnd )
   hdcMem   = MSWIn_CreateCompatibleDC( hdc )
   hBitmap  = MSWin_CreateCompatibleBitMap( hdc, ClientX , ClientY )
   hBmpOrig = MSWin_SelectObject( hdcMem, hBitmap )

   // Done with this...
   Call MSWin_ReleaseDC( hwnd, hdc )

   // Draw into the memory DC and therefore the bitmap
   Left  = int( ClientX / 8 ); Top = int( ClientY / 8 )
   Right = 7 * Left; Bottom = 7 * Top
   Call MSWin_Rectangle( hdcMem, Left, Top, Right, Bottom )

   Call MSWin_MoveToEx( hdcMem, 0, 0, 0 );
   Call MsWin_LineTo( hdcMem, ClientX, ClientY );

   Call MSWin_MoveToEx( hdcMem, 0, ClientY, 0 );
   Call MsWin_LineTo( hdcMem, ClientX, 0 );

   Call MSWin_Ellipse( hdcMem, Left, Top, Right, Bottom )

   Left  = int( ClientX / 4 ); Top = int( ClientY / 4 )
   Right = 3 * Left; Bottom = 3 * Top
   Call MSWin_RoundRect( hdcMem, Left, Top, Right, Bottom,      |
                         int( ClientX / 4 ), int( ClientY / 4 ) )

   // Put the original bitmap back
   Call MSWin_SelectObject( hdcMem, hBmpOrig  )

   // Set the new bitmap in the control's IMAGE sub-object (creates a copy)
   IsOK = Exec_Method( @Window : ".BMP_TEST.IMAGE", "SETHBITMAP", |
                       hBitmap, "", PS_SBM_OPT_IGNOREALPHA$ )

   // Delete the bitmap
   Call MSWin_DeleteDC( hDCMem )
   Call MSWin_DeleteObject( hBitMap )

return TRUE$
```

See also

Image FILENAME property, Image FILENAMES property, Image SETIMAGE method.

SETIMAGE method

Description

Loads an image using an array of raw bytes.

Syntax

```
SuccessFlag = Exec_Method( ObjImageID, "SETIMAGE", ImageBytes )
```

Parameters

Name	Required	Description
ObjImageID	Yes	Name of the image object to access. e.g. @Window : ".IMAGE"
ImageBytes	Yes	An array of bytes that define the image.

Returns

Returns TRUE$ if the image was loaded successfully, or FALSE$ otherwise.

Remarks

The format of the image (PNG,JPG,BMP etc) is deduced from the passed array of bytes.

If an image is loaded via this method the object's FILENAME and FILENAMES properties will return the string "<memory>".

Automatic image selection for DPI scaling cannot be applied to the image when it is set using this method. Only the FILENAMES property supports this feature.

Example

```
// Example - Read a JPG file and use the contents to set the image in
// the current control

OSRead JPGBytes From "C:\Temp\MyImage.jpg" Then
   IsOK = Exec_Method( CtrlEntID : ".IMAGE", "SETIMAGE", JPGBytes )
End
```

See also

Image FILENAME property, Image FILENAMES property, Image SETHBITMAP method.

SETREPOSIMAGE method

Description

Loads an image file and related properties into an image object using an OpenInsight repository entity ID.

Syntax

```
SuccessFlag = Exec_Method( ObjImageID, "SETREPOSIMAGE", ReposID )
```

Parameters

Name	Required	Description
ObjImageID	Yes	Name of the image object to set the image properties for. e.g. @Window : ".IMAGE"
ReposID	Yes	Fully qualified repository ID of the image entity to load.

Returns

Returns TRUE$ if the image entity was loaded successfully, or FALSE$ otherwise. Error information is reported via the Get_Status stored procedure.

Remarks

Image entities stored in the OpenInsight repository contain the following image object properties:

- FILENAMES
- COLORKEY
- COUNT

The SETREPOSIMAGE method uses the passed entity ID to access these properties and set them for the image object.

Example

```
// Example - Load the RTI_IDE_DEFAULT_IMAGE PNG file into the current
// control's IMAGE sub-object

ReposID = @AppID<1> : "*IMAGE*PNG*RTI_IDE_DEFAULT_IMAGE"
If Exec_Method( CtrlEntID : ".IMAGE", "SETREPOSIMAGE", ReposID ) Else
   Call Get_Status( ErrInfo )
End
```

See also
COLORKEY property, COUNT property, FILENAMES property.