ImageMagick Plug-in

By Thomas Maul, General Manager, 4D
TN 06-32

Introduction

This Plug-in enhances the support of picture-handling commands using the Open Source Framework ImageMagick (please refer to www.ImageMagick.org).
Even if ImageMagick can be called directly using LAUNCH EXTERNAL PROCESS, this plug-in removes the need to install the application on each client computer, using 4D's plug-in mechanism it is automatically deployed to all computers.

ImageMagick is a very feature-rich graphical application that allows you to resize, crop, rotate, enhance and manipulate images. It supports a wide area of image formats. QuickTime is not needed to use this plug-in.

License

ImageMagick is free software packaged with full source code and can be freely used, copied, modified and distributed. See http://www.imagemagick.org/script/license.php for the license information. It allows the usage for commercial purpose and distribution.

In short:
To be allowed to include ImageMagick into your application, you must include the license agreement file on your Distribution CD and provide clear credits and attribution to ImageMagick Studio LLC in your About dialog (copyright notice).

Main Features

Static link
The plug-in includes ImageMagick as a static link, which makes it possible to run on computers without a full ImageMagick installation. All the code is contained in the plug-in, which makes it larger. A major drawback is that not all picture formats are supported, because their libraries are not included: either because of legal problems (no free license) or because of their size. The plug-in uses ImageMagick version 6.2.7 (current version of June 2006).

Cross platform
The plug-in runs on both Mac OS and Windows with the same feature set.

External document and blob support
The plug-in allows you to use either external documents or images stored in a 4D blob object, which allows you to easily store pictures in the 4D database and display them using 4D's commands or the Pict Container plug-in.

More documentation
The plug-in is a wrapper of Magick++ and often directly passes the parameters. More info about the parameters may be found at:
http://www.imagemagick.org/Magick++/Documentation.html

Plug-in workflow

All commands are written around image objects. Several image objects can exist simultaneously; this is limited only by available memory.
An image objects can be read, written, or copied, the contained image properties can be read or written and the image can be manipulated.

Image Objects

IM New Object

IM New Object-> error code

Parameter	***Type***	***Description***
Function result	Longint	<- Error code (0 = No error)

Description
This command creates a new, empty, image object in memory.
Errors are not to be expected in this phase, see error addendum for details.

IM Clear Object

IM Clear Object(object) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Function result	Longint	<- Error code (0 = No error)

Description
The command clears an object from memory. If it is the last object referring to an image the memory it used is released.

Errors are only to be expected for invalid Object values, see error addendum for details.

IM Copy Object

IM Copy Object(dest_object; source_object) -> error code

Parameter	***Type***	***Description***
Dest_object	Longint	-> ImageMagick object
Source_object	Longint	-> ImageMagick object
Function result	Longint	<- Error code (0 = No error)

Description
This command creates a copy of the reference to the image stored in source_object and sets it into dest_object. Both objects must have been created with IM New Object beforehand, Source_object must be a valid object.

This command does not always copy the image itself, usually only a reference and modification profile. Only for major modifications is the whole image copied, in order to reduce memory usage.

Expect errors if one of the objects is not created or if the source_object is empty, refer to the error addendum for more details.

Image Usage

IM Open File

IM Open File(object; path) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
path	Text	-> full Path to document
Function result	Longint	<- Error code (0 = No error)

Description
This command opens the specified image and places it the existing object. Only the file content is used to define the image type.

Settings from the existing object are used, like Geometry (which allows you to specify for some image types like JPEG or PictureCD the requested image size). To overwrite any existing information use IM Clear Object/IM New Object before.

For multiple page formats (GIF/TIF) it is possible to specify the requested page by adding the page number after the file name. Example:
C:\folder\document.gif -> C:\folder\document.gif[0] for first page.

Refer to the error addendum for possible error codes.

IM Open Blob

IM Open Blob(object; blob) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
blob	Blob	-> 4D blob field/variable
Function result	Longint	<- Error code (0 = No error)

Description
This command tries to open the specified image stored in a 4D blob. Only the blob content is used to define the image type.

Settings from the existing object are used, like Geometry (which allows you to specify the requested image size for some image types like JPEG or PictureCD). To overwrite any existing information use IM Clear Object/IM New Object beforehand.

Refer to the error addendum for possible error codes.

IM Save File

IM Save File(object; path) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
path	Text	-> full Path to document
Function result	Longint	<- Error code (0 = No error)

Description
This command tries to save the specified image. The format is specified with the file extension. For documents without file extension, the format can also be set using:
$err:=IM Modify Image(object;IM_Magick;"PDF";0;0;0;0;0;0)
The compression quality can be set using:
$err:=IM Set Image Properties (IM;IM_Pref_Quality;"";50;0;0;0)
` for 50% compression

Refer to the error addendum for possible error codes.

IM Save Blob

IM Save Blob(object; blob) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
blob	Blob	<- 4D blob field/variable
Function result	Longint	<- Error code (0 = No error)

Description
This command tries to save the specified picture. The picture is saved in its original format. The format can be changed using:
$err:=IM Modify Image(object;IM_Magick;"PDF";0;0;0;0;0;0)
The compression quality can be set using:
$err:=IM Set Image Properties (IM;IM_Pref_Quality;"";50;0;0;0)
` for 50% compression

Refer to the error addendum for possible error codes.

IM Create

IM Create(object; Width; Height; Color) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Width	Longint	-> Width of image
Height	Longint	-> Height of image
Color	String	-> Background color of image
Function result	Longint	<- Error code (0 = No error)

Description
This command creates a new empty picture, overwriting all information's in object. Width and Height specify the size of the image, Color is the background color in RGB format (eg #FFFFFF) or X11 (eq red, black, blue). If the string starts with the # sign, then the color in RGB format as hex. For examples how to calculate the RGB color see 4D's programmers manual: SET RGB COLORS.

Refer to the error addendum for possible error codes.

IM Get Image Properties

IM Get Image Properties(object; Selector; String; para1; para2; para3; para4) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Selector	Longint	-> function selector
String	Alpha/Text	<-> text result
Para1	Num	<- numeric result 1
Para2	Num	<- numeric result 2
Para3	Num	<- numeric result 3
Para4	Num	<- numeric result 4
Function result	Longint	<- Error code (0 = No error)

Description

This command reads the properties of a loaded picture. Some selectors can be used for Write, all for Read.

Selector

1	IM_Pref_Quality	returns in Para1 the quality of a compressed image
2	IM_Pref_Size	returns in Para1 and Para2 the X/Y Size of the image in Pixels using ImageMagick baseColumns/baseRows. This is the raw size of the image
3	IM_Pref_Geometry	returns in Para1 and Para2 the X/Y the Size of the image in Pixel, in Para3 and Para4 the X/Y Resolution of the image, using ImageMagick columns/rows/xResolution/yResolution. This is the preferred size of the image when encoding.
4	IM_Pref_Magick	returns in String the format of the current image (like JPEG or GIF)
5	IM_Pref_Attribute	returns in String the content of the requested Exif, ITPC or ICC attributes. Specify the requested attribute in String, like "Exif:DateTime" or "Exif:ShutterSpeedValue". After the command is executed, the String parameter contains the content of this attribute. If the attribute does not exist String is empty. For a list of possible attribute names see: http://www.exif.org/specifications.html
6	IM_Pref_BackgroundColor	returns in String the background color in RGB format (i.e. #FFFFFF) or X11 (i.e. red, black, and blue). If the string starts with the # sign, then the color in RGB format as hex. For examples how to calculate the RGB color see 4D's programmers manual: SET RGB COLORS. The background color is used for operations like rotate, wave, shear, etc.
7	IM_Pref_BorderColor	Returns in String the BorderColor. Format like BackgroundColor
8	IM_Pref_BoxColor	Returns in String the BoxColor. Format like BackgroundColor

Refer to the error addendum for possible error codes.

IM Set Image Properties

IM Set Image Properties(object; Selector; String; para1; para2; para3; para4) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Selector	Longint	-> function selector
String	Alpha/Text	-> text result
Para1	Num	-> numeric parameter 1
Para2	Num	-> numeric parameter 2
Para3	Num	-> numeric parameter 3
Para4	Num	-> numeric parameter 4
Function result	Longint	<- Error code (0 = No error)

Description

The command writes properties from a loaded image. Some selectors can be used for Write, all for Read.

Selector

1	IM_Pref_Quality	use Para1 to set the quality of a compressed image
2	IM_Pref_Magick	use String to change the format of an image, like JPEG, GIF, BMP, TIFF, etc
6	IM_Pref_BackgroundColor	use value of String to set the background color in RGB format (i.e. #FFFFFF) or X11 (i.e. red, black, and blue). If the string starts with the # sign, then the color in RGB format as hex. For examples on how to calculate the RGB color see 4D Language Reference manual: SET RGB COLORS. The background color is used for operations like rotate, wave, shear, etc.
7	IM_Pref_BorderColor	use value of String to set the Border color. Format it like Backgroundcolor.
8	IM_Pref_BoxColor	use value of String to set the Box color. Format it like Backgroundcolor.

Refer to the error addendum for possible error codes.

IM Modify Image

IM Modify Image(object; Selector; String; para1; para2; para3; para4; para5, Para6) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Selector	Longint	-> function selector
String	Alpha/Text	-> text result
Para1	Num	-> numeric parameter 1
Para2	Num	-> numeric parameter 2
Para3	Num	-> numeric parameter 3
Para4	Num	-> numeric parameter 4
Para5	Num	-> numeric parameter 5
Para6	Num	-> numeric parameter 6
Function result	Longint	<- Error code (0 = No error)

Description
This command writes properties from a loaded image. Some selectors can be used for Write, all for Read.

Selector

All values are based on the ImageMagick Mackick++ Image Manipulation Methods:
http://www.imagemagick.org/Magick++/Image.html#Image%20Manipulation%20Methods
You may find additional information on the ImageMagick pages at the location above.

IM_Mod_addNoise **1**
Adds noise to images using the specified noise type. Possible values for Para1 are:

UniformNoise	0
GaussianNoise	1
MultiplicativeGaussianNoise	2
ImpulseNoise	3
LaplacianNoise	4
PoissonNoise	5

IM_Mod_flip **2**
Flips image (reflects each line in the vertical direction)

IM_Mod_flop **3**
Flops image (reflects each line in the horizontal direction)

IM_Mod_implode **4**
Implodes image (special effect). Factor is passed in Para1 as floating value; usual values are -20 to +20.

IM_Mod_rotate **5**
Rotates image counter-clockwise by a specified number of degrees; Usual values for Para1 is 0..359 (floating value).

IM_Mod_crop **6**
Crops image (subregion of original image). Para 1-4 defines: Left, Top, Right, Bottom.

IM_Mod_blur **7**
Blur image. Para1 specifies the radius of the Gaussian blur, in pixels, not counting the center pixel. Para2 specifies the standard deviation of the Laplacian, in pixels.

IM_Mod_contrast **8**
Contrast image (enhances intensity differences in an image). Para1 is a longint defining the sharpen value.

IM_Mod_despeckle **9**
'Despeckles' image (reduce speckle noise). No parameters.

IM_Mod_adaptiveThreshold **10**
Apply an adaptive thresholding to the image. An adaptive threshold is useful if the ideal threshold level is not known in advance, or if the illumination gradient is not constant across the image. Adaptive thresholds work by evaluating the average pixel value for a given area (size specified by Para1=*width* and Para2=*height*) and using the average as the threshold value. In order to remove residual noise from the background, the threshold may be adjusted by subtracting a constant *offset*=Para3 (default zero) from the mean to compute the threshold.

********** **11**

IM_Mod_Zoom **12**
Zooms image to specified size (Para1=Width, Para2=Height)

IM_Mod_Edge **13**
Highlights edges of an image). Para1 is the radius of the pixel neighborhood (longint). Specify a radius of zero for automatic radius selection.

IM_Mod_Emboss **14**
Emboss image (highlights edges with a 3D effect). Para1 specifies the radius of the Gaussian, in pixels, not counting the center pixel. Para2 specifies the standard deviation of the Laplacian, in pixels. Both parameters are floating values.

IM_Mod_Enhance **15**
Enhances the image (minimize noise). No parameters.

IM_Mod_Equalize **16**
Equalizes the image (histogram equalization). No parameters.

IM_Mod_Frame **17**
Adds decorative frame around image. Para1/Para2 = Width/Height. Para3/Para4 = InnerBevel/OuterBevel

IM_Mod_Gamma **18**

Applies a gamma correction. If Para1=0: Gamma correction of the separate channels red (Para2), green (Para3), and blue (Para4). If Para1 # 0: uniform red, green, and blue correction.

IM_Mod_GaussianBlur 19
Applies a Gaussian blur to the image. The number of neighboring pixels to be included in the convolution mask is specified by Para 1. For example, a width of one gives a (standard) 3x3 convolution mask. The standard deviation of the Gaussian bell curve is specified by Para2. Both parameters are floating values.

IM_Mod_Level 20
Adjust the levels of the image by scaling the colors falling between specified white and black points to the full available quantum range. The parameters provided represent the black, mid (gamma), and white points. The black point specifies the darkest color in the image. Colors darker than the black point are set to zero. Mid point (gamma) specifies a gamma correction to apply to the image. White point specifies the lightest color in the image. Colors brighter than the white point are set to the maximum quantum value. Para1 specifies the black point, Para2 the white point, both as percentage (0..100), Para 3 is the mid (gamma) and has a useful range of 0 to 10.

IM_Mod_Magnify 21
Magnifies the image by integral size. No parameters.

IM_Mod_Minify 22
Reduces the image by integral size. No parameters.

IM_Mod_MedianFilter 23
Filters image by replacing each pixel component with the median color in a circular neighborhood. Para1 = radius (floating value).

IM_Mod_Modulate 24
Sets the percentage of modulation for the hue (Para3), saturation (Para2), and brightness (Para1) of an image. Modulation of saturation and brightness is as a ratio of the new value over the current value (1.0 for no change). Modulation of hue is an absolute rotation of -180 degrees to +180 degrees from the current position corresponding to an argument range of 0 to 2.0 (1.0 for no change).

IM_Mod_Negate 25
Creates the negative of an image. Replaces every pixel with its complementary color (white becomes black, yellow becomes blue, etc.). Set Para1 to 1 to only negate grayscale values in image.

IM_Mod_Normalize 26
Increases the contrast by adjusting the pixel values so that they cover the full range of color values. No parameters.

IM_Mod_OilPaint 27
Make the image look like an oil painting. Para 1 = radius (longint)

IM_Mod_Raise **28**
Lightens or darkens the edges of an image to give a 3-D raised or lowered effect. Para1 = Width, Para2= Height. Set Para5 to #0 to lower, 0 to raise.

IM_Mod_ReduceNoise **29**
The principal function of noise peak elimination filters is to smooth the objects within an image without losing edge information and without creating undesired structures. The central idea of the algorithm is to replace a pixel with its next neighboring value within a pixel window, if this pixel has been found to be noise. A pixel is defined as noise if and only if this pixel is a maximum or minimum within the pixel window. Use Para1 to specify the width of the neighborhood. Set to 0 for automatic.

IM_Mod_Roll **30**
Rolls the image (rolls image vertically and horizontally) by specified number of columns (Para1) and rows (Para2)).

IM_Mod_Sample **31**
Resizes the image using a pixel sampling algorithm. Para1 = Width, Para2= Height.

IM_Mod_Scale **32**
Resize image by using simple ratio algorithm. Para1 = Width, Para2= Height.

IM_Mod_Segment **33**
Segments (coalesces similar image components) by analyzing the histograms of the color components and identifying units that are homogeneous with the fuzzy c-means technique. Also uses quantizeColorSpace and verbose image attributes. Specify Para1, as the number of pixels each cluster must exceed for the cluster threshold to be considered valid. Default is 1.0. Para2 eliminates the noise in the second derivative of the histogram. As the value is increased, you can expect a smoother second derivative. The default is 1.5.

IM_Mod_Shade **34**
Simulate the addition of a shade image using a distant light source. Specify azimuth (Para1, default 30) and elevation (Para2, default 30) as the position of the light source. By default, the shading results as a grayscale image. Set Para3 to 1 to shade the red, green, and blue components of the image. Para1 and Para2 are floating values.

IM_Mod_Sharpen **35**
Sharpens pixels in the image. Para1 (Default 1.0) specifies the radius of the Gaussian, in pixels, not counting the center pixel. Para2 (Default 0.5) specifies the standard deviation of the Laplacian, in pixels, both are floating values.

IM_Mod_Shave **36**
Shave pixels from image edges. Para1 = Width, Para2= Height.

IM_Mod_Shear	**37**

Shearing slides one edge of an image along the X or Y axis, creating a parallelogram. An X direction shear slides an edge along the X axis, while a Y direction shear slides an edge along the Y axis. The amount of the shear is controlled by a shear angle. For X direction shears, x degrees is measured relative to the Y axis, and similarly, for Y direction shears y degrees is measured relative to the X axis.
Para1 is the xShearAngle, Para2 the yShearAngle, both floating values.
Empty triangles left over from shearing the image are filled with the color defined as borderColor.

IM_Mod_Solarize	**38**

Solarize image (similar to the effect seen when exposing a photographic film to light during the development process). Para1 = factor (Default 50.0), floating value.

IM_Mod_Spread	**39**

Spreads pixels randomly within image by specified amount (Para1, Default = 3, longint)

IM_Mod_Swirl	**40**

Applies a swirl effect to an image (image pixels are rotated by an angle, expressed in degrees (Para1, floating value)

IM_Mod_Threshold	**41**

Threshold image set by Para1, floating value.

IM_Mod_Trim	**42**

Trim edges that have the same color as the background color. No parameters.

IM_Mod_Unsharpmask	**43**

Replaces the image with a sharpened version of the original image using the unsharp mask algorithm. Para1 specifies the radius of the Gaussian, in pixels, not counting the center pixel. Para2 specifies the standard deviation of the Gaussian, in pixels. Para3 specifies the percentage of the difference between the original and the blur image that is added back into the original. Para4 specifies the threshold in pixels needed to apply to the difference amount. All parameters are floating values.

IM_Mod_Wave	**44**

Alters an image along a sine wave. Para1 = amplitude (Default = 25.0), Para2 = wavelength (Default = 150.0).

Refer to the error addendum for possible error codes.

IM Draw

IM Draw(object; Selector; String; para1; para2; para3; para4) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
Selector	Longint	-> function selector
String	Alpha/Text	-> text result
Para1	Num	-> numeric result 1
Para2	Num	-> numeric result 2
Para3	Num	-> numeric result 3
Para4	Num	-> numeric result 4
Function result	Longint	<- Error code (0 = No error)

Description

The command allows drawing operations in an object, which can be a loaded image or a newly created image.
The attributes of the drawing can be set (before executing IM Draw) with IM Set Image Properties and selectors like IM_Pref_StrokeColor, IM_Pref_FillColor, etc.

Selector

1	IM_Draw_Circle	Creates a Circle. Para1/2 define the center point (X/Y), Para3/4 define a point on the parameter to define the radius
2	IM_Draw_Rectangle	Creates a Circle, defined by Para1-4: left, top, right, bottom
3	IM_Draw_Arc	Draws an arc using the *stroke* color and based on the circle starting at coordinates startX_,startY (Para1/Para2*)*, and ending with coordinates *endX_,*endY (Para3/Para4), and bounded by the rotational arc startDegrees_,endDegrees (Para5/Para)
4	IM_Draw_Color	Pixel is specified by X/Y = Para1/Para2. Color image according to paintMethod (Para3). The dot method redefines the color of the target pixel. The replace method redefines the color any pixel that matches the color of the target pixel. Floodfill redefines the color any pixel that matches the color of the target pixel and is a neighboring pixel, whereas filltoborder redefines the color any neighbor pixel that is not the border color. Finally, reset redefines the color all pixels. See below for PaintMethod.
5	IM_Draw_Composite	Composite current image with contents of specified image, rendered with specified width and height (Para4/Para5), using specified composition algorithm (Para6), at specified coordinates (Para2/Para3). The composite image is either passed as IM_Object in Para1 (created with IM New Object) or as File path in stringpara. See below for a list of possible values for CompositeOperator (Para5).
6	IM_Draw_Ellipse	Draws an ellipse using the *stroke* color and thickness, specified origin (Para1/Para2), x & y radius (Para3/Para4), as well as specified start and end of arc in degrees (Para5/Para6). If a *fill* color is specified, then the object is filled with that color.

7	IM_Draw_Line	Draws a line using *stroke* color and thickness using starting and ending coordinates (Para1-Para4)
8	IM_Draw_Matte	Changes the pixel matte value to transparent. The dot method changes the matte value of the target pixel (Para1/Para2). The replace method (Para3) changes the matte value of any pixel that matches the color of the target pixel. Floodfill changes the matte value of any pixel that matches the color of the target pixel and is a neighboring pixel, whereas filltoborder changes the matte value of any neighbor pixel that is not the border color. Finally reset changes the matte value of all pixels. See below for PaintMethod.
9	IM_Draw_Point	Draws a dot using *stroke* color and thickness at coordinate Para1/Para2.
10	IM_Draw_RoundRectangle	Draw a rounded rectangle using *stroke* color and thickness, with specified center coordinate (Para1/Para2), specified width and height (Para3/Para4), and specified corner width and height (Para5/Para6). If a *fill* color is specified, then the object is filled.

PaintMethod

0	PointMethod	Replaces pixel color at designated point.
1	ReplaceMethod	Replaces color for all image pixels whose color match.
2	FloodfillMethod	Replaces color for pixels surrounding pixel until encountering a pixel that fails to match the color.
3	FillToBorderMethod	Replaces color for pixels surrounding point until encountering pixels matching the border's color.
4	ResetMethod	Replaces colors for all pixels in the image with the pen's color.

CompositeOperator

CompositeOperator is used to select the image composition algorithm used to compose a composite image with the other image. Usually (12 =) each of the composite image pixels are replaced by the corresponding image tile pixel.

1	OverCompositeOp	The result is the union of the two image shapes with the composite image obscuring the resulting image in the region of the overlap.
2	InCompositeOp	The result is simply a composite image cut by the shape of the other image. None of the other image's data is included in the result.
3	OutCompositeOp	The resulting image is a composite image with the shape of the other image cut out.
4	AtopCompositeOp	The result is the same shape as image, with the composite image obscuring image there the image shapes overlap. Note that this differs from

		OverCompositeOp because the portion of the composite image outside of image's shape does not appear in the resulting image.
5	XorCompositeOp	The result is the image data from both images for the area that is outside the overlap region. The overlap region is left blank.
6	PlusCompositeOp	The result is just the sum of the images' data. Output values are cropped to 255 (no overflow). This operation is independent of the matte channels.
7	MinusCompositeOp	The result is the subtraction of one image from the other, with overflow cropped to zero. The matte channel is ignored (set to 255).
8	AddCompositeOp	The result of is the addition of the two images, with overflow 'wraparound' (modulo 256).
9	SubtractCompositeOp	The result is the subtraction of one image from the other, with underflow 'wraparound' (mod 256). The add and subtract operators can be used to perform reversible transformations.
10	DifferenceCompositeOp	The result is comparable to that of abs(composite image - image). This is useful for comparing two very similar images.
11	MultiplyCompositeOp	The result is comparable to that of (composite image x image).
12	BumpmapCompositeOp	The resulting image is shaded by the composite image.
13	CopyCompositeOp	The resulting image is an image replaced with composite image. Here the matte data is ignored.
14	CopyRedCompositeOp	The resulting image is the red layer in the image replaced with the red layer in the composite image. The other layers are copied untouched.
15	CopyGreenCompositeOp	The resulting image is the green layer in the image replaced with the green layer in the composite image. The other layers are copied untouched.
16	CopyBlueCompositeOp	The resulting image is the blue layer in the image replaced with the blue layer in the composite image. The other layers are copied untouched.

17	CopyOpacityCompositeOp	The resulting image is the matte layer in the image replaced with the matte layer in composite image. The other layers are copied untouched. The image compositor requires a matte, or alpha channel in the image for some operations. This extra channel usually defines a mask which represents a sort of a ‘cookie-cutter mask’ for the image. This is the case when matte is 255 for pixels inside the shape, zero outside, and between zero and 255 on the boundary. For certain operations, if the image does not have a matte channel, it is initialized with 0 for any pixel matching in color to pixel location (0,0), otherwise 255 (to work properly borderWidth must be 0).
18	ClearCompositeOp	N.A.
19	DissolveCompositeOp	N.A.
20	DisplaceCompositeOp	N.A.
21	ModulateCompositeOp	N.A.
22	ThresholdCompositeOp	N.A.

refer to the error addendum for possible error codes.

Utlities

IM Last Error

IM Last Error (object; string) -> error code

Parameter	***Type***	***Description***
Object	Longint	-> ImageMagick object
string	Text	<- Error documentation
Function result	Longint	<- Error code (0 = No error)

Returns an error message in English for the last error with this object. This returns useful information only if a command returned -2.

IM Get Format List

IM Get Format List (list) -> error code

Parameter	***Type***	***Description***
list	Array Text	<- Format list
Function result	Longint	<- Error code (0 = No error)

This command fills a text array with the list of image formats supported by ImageMagick. Each element describes a supported format, like:
GIF (CompuServe graphics interchange format) rwm
JP2 (JPEG-2000 File Format Syntax) rw
JPEG (Joint Photographic Experts Group JFIF format) rw
TIFF (Tagged Image File Format) rwm

The first word is the format (used as file extension and as "Magick" String with IM Set Image Properties(…;IM_ Pref_Magick…)). Separated by a space is the description of the format. The last word (in fact always the last 3 characters) describes the supported features: r=read, w=write, m=multiple image/page.

ImageMagick handles many image formats indirectly by calling other tools or libraries. This means the supported formats depend on the installed libraries, which makes ImageMagick often difficult to install and maintain.
To avoid this, the plug-in is compiled as a "static" build, including many needed libraries or other tools. The disadvantage is that non included libraries are not supported at all, even if they are installed on the customer's computer. Especially Postscript formats (like EPS/EPSF/PS/PDF) are not supported for reading, only for writing. Also movie formats (MPEG, MV2, AVI) are not supported.

Error codes

The plug-in returns the following error codes:

-1	Unknown Object (not initialized with IM New Object)
-2	ImageMagick Framework Error, use IM Last Error to get more info
-3	Uninitialized Object (created with IM New Object, but does not contain an image yet)
-4	Unsupported Selector