

# **Imagen**

## **for Windows/Linux**

### **Runtime Functions**

*March 28, 2001. Document Revision 3*

---

Netclime Inc.

P.O.Box 251666, LA, CA-90025  
Tel (310) 571-3135 Fax (646) 514-0412

email: [imagen@netclime.com](mailto:imagen@netclime.com)

# Introduction

This document describes the built in functions accessible through Imagen scripts.

## Function Prototypes

### Image Operations

#### Properties

Includes functions that give information about width and height for a certain image.

Function	<code>int image_width(string file_name)</code>
Description	Returns the width of an image in pixels
Parameters	<code>string file_name</code> - a string representing a file path to the image relative to the path of the <code>.cgi</code> script. This path must not point to a directory that is on a higher level from the directory where the <code>.cgi</code> script is
Returns	on success - integer value representing the width of the image on failure - returns -1 if the file cannot be opened for reading

Function	<code>int image_height(string file_name)</code>
Description	Returns the height of an image in pixels
Parameters	<code>string file_name</code> - a string representing a file path of the image relative to the path of the <code>.cgi</code> script. This path must not point to a directory that is on a higher level from the directory where the <code>.cgi</code> script is
Returns	on success - integer value representing the height of the image on failure - returns -1 if the file cannot be opened for reading

### Loading

The functions in this section allow you to load an existing image onto the virtual screen and to have available for editing.

Function	<code>int load_image(string file_name, int x, int y)</code>
Description	Loads image on the virtual screen and positions it's left-top corner on position (x, y) on the virtual screen
Parameters	<p><code>string file_name</code> - a string representing a file path of the image relative to the path of the .cgi script. This path must not point to a directory that is on a higher level from the directory where the .cgi script is</p> <p><code>int x</code> - x coordinate of the left top point of the image rectangle relative to the virtual screen</p> <p><code>int y</code> - y coordinate of the left top point of the image rectangle relative to the virtual screen</p>
Returns	<p>on success - returns 1</p> <p>on failure - returns -1 if the file cannot be opened for reading</p>

### Saving

Includes functions that allow you to save the image into a file or to return the current image on the virtual screen to the browser in the desired format

Function	<code>void save_image(string file_name, int x, int y, int width, int height)</code>
Description	Saves the image on the virtual screen, enclosed by the rectangle made from the x, y, width and height parameters, in a file
Parameters	<code>string file_name</code> - a string representing a file path of the image relative to the path of the .cgi script. This path must not point to a directory that is on a higher level from the directory

	<p>where the .cgi script is</p> <p><code>int x</code> - x coordinate of the left top point of the image rectangle relative to the virtual screen</p> <p><code>int y</code> - y coordinate of the left top point of the image rectangle relative to the virtual screen</p> <p><code>int width</code> - width of the image rectangle</p> <p><code>int height</code> - height of the image rectangle</p>
Returns	Nothing

Function	<pre>void return_image(string file_format,                  int x,                  int y,                  int width,                  int height)</pre>
Description	Returns the image on the virtual screen, enclosed by the rectangle made from the x, y, width and height parameters, in a desired file format
Parameters	<p><code>string file_format</code> - string representing the format of the image, for example "gif", "jpg", "png"</p> <p><code>int x</code> - x coordinate of the left top point of the image rectangle relative to the virtual screen</p> <p><code>int y</code> - y coordinate of the left top point of the image rectangle relative to the virtual screen</p> <p><code>int width</code> - width of the image rectangle</p> <p><code>int height</code> - height of the image rectangle</p>
Returns	Nothing

### Saving Parameters

Includes functions that set various parameters that are used when saving the image

Function	<code>void transparent_color(int red, int green, int blue)</code>
Description	Sets image's transparent color value. All pixels that have this color will be rendered transparent (if supported by the output format)
Parameters	int red - red color value  int green - green color value  int blue - blue color value
Returns	Nothing

Function	<code>void transparent_color_from(int x, int y)</code>
Description	Sets image's transparent color value that's the value of the color on position x,y relatively to the virtual screen's position
Parameters	int x - x coordinate  int y - y coordinate
Returns	Nothing

Function	<code>void output_colors(int max_colors)</code>
Description	Set color palette for the image. The <code>max_colors</code> value is used to produce maximum number of colors to quantize to
Parameters	int max_colors - maximum number of colors to quantize to
Returns	Nothing

Function	<code>void jpeg_ratio(int percent_quality)</code>
Description	Sets the percent quality of the image which value is used only if the image is returned (or saved) into JPEG format
Parameters	int percent_quality - value from 1 to 100
Returns	Nothing

## Operations

Includes functions that apply various image effects

**Note:** Note that the value of parameters that have “100” at the end of the name is divided by 100 and then used in the filter.

Function	<pre>void copy(int x,           int y,           int width,           int height,           int new_x,           int new_y)</pre>
Description	Clears area from the screen enclosed by the rectangle with left-top coordinate the (x, y) point and respectively width and height relative to the virtual screen. The copied rectangle's left-top corner is put on position (new_x, new_y)
Parameters	<pre>int x      – the x coordinate of the rectangle int y      – the y coordinate of the rectangle int width  – the width of the rectangle int height – the height of the rectangle int new_x  – new x coordinate int new_y  – new y coordinate</pre>
Returns	nothing

Function	<pre>void clear(int x,            int y,            int width,            int height,            int red,            int green,            int blue)</pre>
Description	Clears area from the screen enclosed by the rectangle with left-top coordinate the (x, y) point and respectively width and height relative to the virtual screen. The cleared area is filled with the color represented by the red, green, blue values.
Parameters	<pre>int x      – the x coordinate of the rectangle int y      – the y coordinate of the rectangle int width  – the width of the rectangle</pre>

	<pre>int height – the height of the rectangle  int red – red color value  int green – green color value  int blue – blue color value</pre>
Returns	nothing

Function	<pre>void tile(int source_x,           int source_y,           int source_width,           int source_height,           int target_x,           int target_y,           int target_width,           int target_height)</pre>
Description	Tiles the area enclosed by the rectangle with left-top position (source_x, source_y) and width and height respectively source_width and source_height into a rectangle with left top position (target_x, target_y) and width and height respectively target_width and target_height
Parameters	<pre>int source_x – the x coordinate of the source rectangle  int source_y – the y coordinate of the source rectangle  int source_width – the width of the source rectangle  int source_height – the height of the source rectangle  int target_x – the x position of the target rectangle  int target_y – the y position of the target rectangle  int target_width – width of the target rectangle  int target_height – height of the target rectangle</pre>
Returns	nothing

Function	<pre>void stretch(int x,               int y,               int width,               int height,</pre>
----------	--

	<pre>int new_x, int new_y, int new_width, int new_height)</pre>
Description	Stretches area from the image enclosed by the rectangle which left-top coordinate has the values (x, y) and respectively width and height. The rectangle is stretched according to the <code>new_width</code> and <code>new_height</code> values and put at position (new_x, new_y)
Parameters	<pre>int x           – the x coordinate of the rectangle int y           – the y coordinate of the rectangle int width       – the width of the rectangle int height      – the height of the rectangle int new_x       – the new x position int new_y       – the new y position int new_width   – new area width int new_height  – new area height</pre>
Returns	nothing

Function	<pre>void filter_blur(int radius100, int sigma100, int x, int y, int width, int height)</pre>
Description	Applies a <i>blur</i> effect on a rectangular area of the image with left-top position (x, y) and respectively width and height, sigma and radius stuff
Parameters	<pre>int radius100 – the radius of the pixel not counting the center int sigma100  – represents deviation in pixels int x         – the x position int y         – the y position int width     – the width of the rectangle int height    – the height of the rectangle</pre>

Returns	nothing
Function	<pre>void filter_sharpen(int radius100,                     int sigma100,                     int x,                     int y,                     int width,                     int height)</pre>
Description	Applies a <i>sharpen</i> effect on a rectangular area of the image with left-top position (x, y) and respectively width and height, sigma and radius stuff
Parameters	<p>int radius100 – the radius of the pixel not counting the center</p> <p>int sigma100 – represents deviation in pixels</p> <p>int x – the x position</p> <p>int y – the y position</p> <p>int width – the width of the rectangle</p> <p>int height – the height of the rectangle</p>
Returns	nothing

Function	<pre>void filter_display()</pre>
Description	This function tells the interpreter that the filters invoked after this function call will be applied on the whole virtual screen
Parameters	none
Returns	nothing

Function	<pre>void filter_rectangle(int x,                      int y,                      int width,                      int height)</pre>
Description	This function tells the interpreter that the filters invoked after this function call will be applied on rectangle with left top corner position of (x, y) and defined width and height
Parameters	int x – the rectangle's left-top corner x position

	<pre>int y      - the rectangle's left-top corner y position  int width  - the rectangle width  int height - the rectangle height</pre>
Returns	nothing

Function	<b>void filter_noise(int noise_type)</b>
Description	<p>Adds noise to the image. There are 5 types of noise</p> <p>0 - Uniform Noise,  1 - Gaussian Noise,  2 - Multiplicative Gaussian Noise,  3 - Impulse Noise,  4 - Laplacian Noise,  5 - Poisson Noise</p>
Parameters	int noise_type - one of the types above
Returns	nothing

Function	<b>void filter_charcoal(int radius100, int sigma100)</b>
Description	Applies <i>charcoal</i> filter
Parameters	<pre>int radius100 - the radius of the pixel not counting the center  int sigma100  - represents deviation in pixels</pre>
Returns	nothing

Function	<b>void filter_colorize(int opacity_red, int opacity_green, int opacity_blue, int red, int green, int blue)</b>
Description	Applies <i>colorize</i> filter
Parameters	int opacity_red - red color opacity in %

	<pre>int opacity_green – green color opacity in % int opacity_blue – blue color opacity in % int red – red color quantum int green – blue color quantum int blue – green color quantum</pre>
Returns	nothing

Function	<b>void filter_contrast(int sharpen)</b>
Description	Applies <i>contrast</i> filter
Parameters	int sharpen – sharpen level
Returns	nothing

Function	<b>void filter_cycle_colormap(int amount)</b>
Description	Applies <i>cycle colormap</i> filter
Parameters	int amount – amount of cycling the image colormap
Returns	nothing

Function	<b>void filter_despeckle()</b>
Description	Applies <i>despeckle</i> filter (reduce speckle noise)
Parameters	none
Returns	nothing

Function	<b>void filter_emboss(int radius100, int sigma100)</b>
Description	Applies <i>emboss</i> filter
Parameters	int radius100 – the radius of the pixel not counting the center int sigma100 – represents deviation in pixels

Returns	nothing

Function	<code>void filter_enhance()</code>
Description	Applies <i>enhance</i> filter (minimize noise)
Parameters	none
Returns	nothing

Function	<code>void filter_equalize()</code>
Description	Applies <i>equalize</i> filter (histogram equalization)
Parameters	none
Returns	nothing

Function	<code>void filter_flip_vertical()</code>
Description	Flips the image vertically
Parameters	none
Returns	nothing

Function	<code>void filter_flip_horizontal()</code>
Description	Flips the image horizontally
Parameters	none
Returns	nothing

Function	<code>void filter_gamma(int red100, int green100, int blue100)</code>
Description	Gamma corrects the image
Parameters	int red100 - red gamma correct value  int blue100 - blue gamma correct value

	<code>int green100</code> - green gamma correct value
Returns	nothing

Function	<code>void filter_gaussian_blur(int width100, int sigma100)</code>
Description	Apply a <i>Gaussian</i> blur filter.
Parameters	<code>int width100</code> - pixel blur width  <code>int sigma100</code> - represents deviation in pixels
Returns	nothing

Function	<code>void filter_implode(int factor100)</code>
Description	Applies <i>implode</i> filter (implode image pixels about the center)
Parameters	<code>int factor100</code> - represents the implode factor value
Returns	nothing

Function	<code>void filter_modulate(int brightness100, int saturation100, int hue100)</code>
Description	Modulate percent hue, saturation, and brightness of an image
Parameters	<code>int brightness100</code> - brightness factor value  <code>int saturation100</code> - saturation factor value  <code>int hue100</code> - hue factor value
Returns	nothing

Function	<code>void filter_negate()</code>
Description	Negate colors in image
Parameters	none
Returns	nothing

Function	<code>void filter_normalize()</code>
Description	Normalize image (increase contrast by normalizing the pixel values to span the full range of color values)
Parameters	none
Returns	nothing

Function	<code>void filter_oil_paint(int radius)</code>
Description	Applies <i>oil paint</i> filter
Parameters	<code>int radius</code> - the radius of influence of the filter per pixel
Returns	nothing

Function	<code>void filter_reduce_noise()</code>
Description	Reduce noise in image using a noise peak elimination filter
Parameters	none
Returns	nothing

Function	<code>void filter_roll(int offset_x, int offset_y)</code>
Description	Roll image (rolls image vertically and horizontally) by specified number of columns and rows
Parameters	<code>int offset_x</code> - horizontal offset in pixels <code>int offset_y</code> - vertical offset in pixels
Returns	nothing

Function	<code>void filter_rotate(int degrees100)</code>
Description	Rotates image
Parameters	<code>int degrees100</code> - degrees to rotate

Returns	nothing
---------	---------

Function	<code>void filter_solarize(int factor100)</code>
Description	Solarize image (similar to effect seen when exposing a photographic film to light during the development process)
Parameters	<code>int factor100</code> - solarize factor value
Returns	nothing

Function	<code>void filter_spread(int amount)</code>
Description	Spread pixels randomly within image by specified ammount
Parameters	<code>int amount</code> - amount value
Returns	nothing

Function	<code>void filter_swirl(int degrees100)</code>
Description	Swirl image (image pixels are rotated by degrees)
Parameters	<code>int degrees100</code> - degrees to rotate the pixels
Returns	nothing

Function	<code>void filter_wave(int amplitude100, int wavelength100)</code>
Description	Map image pixels to a sine wave
Parameters	<code>int amplitude100</code> - the amplitude of the sine wave  <code>int wavelegth100</code> - sine wave length
Returns	nothing

## Text Operations

Includes functions that are used to set font properties, drawing properties and to draw text on the screen

### Display

Function	<code>void draw_text(int x,                   int y,                   string text)</code>
Description	Draws <code>text</code> on positions (x, y) relatively to the left-top corner of the virtual screen
Parameters	<code>int x</code> - x coordinate  <code>int y</code> - y coordinate  <code>string text</code> - the text to be drawn
Returns	nothing

### Font Selection

Function	<code>int select_font(string font)</code>
Description	Loads the font that will be used when calling the <code>draw_text</code> function. Loading another font will replace the current one
Parameters	<code>string font</code> - a string representing a file path of the TTF (true type font) relative to the path of the <code>.cgi</code> script. This path must not point to a directory that is on a higher level from the directory where the <code>.cgi</code> script is
Returns	on success -returns 1 on failure -returns -1 if the file cannot be opened for reading

Function	<code>void font_size(int size)</code>
Description	Sets font size in <code>points</code> which will be used when calling the <code>draw_text</code> function
Parameters	<code>int size</code> - size in points
Returns	nothing

Function	<code>void set_kerning(int kerning)</code>
Description	Sets text kerning value (the distance between letters, in pixels). Could be less than zero.
Parameters	<code>int kerning</code> – kerning value
Returns	nothing

Function	<code>void line_spacing_offset(int offset)</code>
Description	Sets text line spacing offset value (changes the default distance between the lines with some <code>offset</code> , in pixels). Could be less than zero too.
Parameters	<code>int offset</code> – offset value
Returns	nothing

### Decoration

Function	<code>void text_color(int red, int green, int blue)</code>
Description	Sets the text color that will be used on the next call of <code>draw_text</code> function
Parameters	<code>int red</code> – red color value  <code>int green</code> – green color value  <code>int blue</code> – blue color value
Returns	nothing

Function	<code>void antialiasing(int type)</code>
Description	Sets antialiasing mode. The default value is 1, i.e. enabled antialiasing mode 1.
Parameters	<code>int type</code> – takes values 0, 1, 2, where 0 means disabled
Returns	nothing

Function	<code>void simple_antialiasing_factor(int factor)</code>
Description	Sets different factors of antialiasing. The default antialiasing factor value is 2. Note that bigger <code>factor</code> value you have the slower the images will be displayed.
Parameters	<code>int factor</code> – takes values from 1 to 32
Returns	nothing

Function	<code>void font_underline(int on)</code>
Description	Enables/Disables underline mode. The default value is Disabled.
Parameters	<code>int on</code> – when <code>on</code> is non zero the font underline is enabled
Returns	nothing

### Positioning

Function	<code>void max_text_width(int width)</code>
Description	Limits the text width to the value represented by the <code>width</code> variable. If <code>width</code> is 0 the text has a default for the font width
Parameters	<code>int width</code> – desired text width
Returns	nothing

Function	<code>void max_text_height(int height)</code>
Description	Limits the text height to the value represented by the <code>height</code> variable. If <code>height</code> is 0 the text has a default for the font height
Parameters	<code>int height</code> – desired text height
Returns	nothing

Function	<code>void align_x_center()</code>
Description	Sets the text alignment to <i>centered horizontally</i> . This alignment will be used in the next call of <code>draw_text</code>

Parameters	None
Returns	nothing

Function	<b>void align_y_center()</b>
Description	Sets the text alignment to <i>centered vertically</i> . This alignment will be used in the next call of <code>draw_text</code>
Parameters	None
Returns	nothing

Function	<b>void align_left()</b>
Description	Sets the text alignment to <i>left</i> . This alignment will be used in the next <code>draw_text</code> function call
Parameters	none
Returns	nothing

Function	<b>void align_right()</b>
Description	Sets the text alignment to <i>right</i> . This alignment will be used in the next call of <code>draw_text</code>
Parameters	none
Returns	nothing

Function	<b>void align_top()</b>
Description	Sets the text alignment to <i>top</i> . This alignment will be used in the next call of <code>draw_text</code>
Parameters	none
Returns	nothing

Function	<b>void align_bottom()</b>
Description	Sets the text alignment to <i>bottom</i> . This alignment will be used in the next call of <code>draw_text</code>
Parameters	none

Returns	nothing
---------	---------

## Cropping

### Clipping and Wrapping

Function	<code>void no_clip()</code>
Description	Disables wrapping mode (default is <code>wrapping</code> )
Parameters	none
Returns	nothing

Function	<code>void word_clip()</code>
Description	Enables word clipping mode. When a text has to be printed and there are words that go out of the virtual screen area (to the left or to the right) these words are not drawn
Parameters	none
Returns	nothing

Function	<code>void char_clip()</code>
Description	Enables char clipping mode. When a text has to be printed and there are chars that go out of the virtual screen area (to the left or to the right) these chars are not drawn
Parameters	none
Returns	nothing

Function	<code>void word_wrap(int on)</code>
Description	Enables/Disables word wrapping. But before calling the <code>draw_text</code> function there must be called the <code>max_text_width</code> function
Parameters	<code>int on</code> – when <code>on</code> is non zero the line wrap is disabled
Returns	nothing

Function	<code>void line_wrap(int on)</code>
Description	Enables/Disables line wrapping. This means that if you have text to be drawn and the last line (if drawn) will have only its top part visible, so this function disables this line “cut” and doesn’t draw it. But before calling the <code>draw_text</code> function there must be called the <code>max_text_height</code> function
Parameters	<code>int on</code> – when <code>on</code> is non zero the line wrap is disabled
Returns	nothing

### Accessing Metrics

Function	<code>int text_width(string text)</code>
Description	Returns the text width in pixels
Parameters	<code>string text</code> – the text which we want to measure
Returns	Integer value representing the text width in pixels

Function	<code>int text_height(string text)</code>
Description	Returns the text height in pixels
Parameters	<code>string text</code> – the text which we want to measure
Returns	Integer value representing the text height in pixels

### String Manipulation

Includes functions applies various image effects

Function	<code>int strlen(string text)</code>
Description	Returns the text characters’ count
Parameters	<code>string text</code> – a string
Returns	Integer value representing the length of the text

Function	<code>string itoa(int number)</code>
----------	--------------------------------------

Description	Converts an <i>Integer</i> value to a <i>String</i> one
Parameters	<code>int number</code> – number to be converted
Returns	String value

Function	<code>int atoi(string text)</code>
Description	Converts ASCII string to integer.
Parameters	<code>string text</code> – a string
Returns	Integer value representing the length of the text. Returns 0 if the text doesn't represent a number

Function	<code>string sub_string(string text, int from, int count)</code>
Description	Gets a sub string from a specified string
Parameters	<code>string text</code> – a string  <code>int from</code> – from character index (the least value is 0)  <code>int count</code> – how many characters to get from the <code>from</code> position on
Returns	The substring value

Function	<code>string toupper(string text)</code>
Description	Converts a text to UPPERCASE
Parameters	<code>string text</code> – the text to be converted
Returns	The converted text

Function	<code>string tolower(string text)</code>
Description	Converts a text to lowercase
Parameters	<code>string text</code> – the text to be converted

Returns	The converted text

## Mathematics

Includes functions for mathematical operations

Function	<code>int math_pow(int base, int degree)</code>
Description	Returns the <i>base</i> over the power of <i>degree</i>
Parameters	int base - the base int degree - the degree
Returns	Integer representing the result of $base^{degree}$

Function	<code>int math_sin(int rad100)</code>
Description	Calculates the sine function
Parameters	int rad100 - the value is in radians, multiplied by 100
Returns	An integer value, multiplied by 100 (because some values may be with floating point, but Imagen supports only integer values)

Function	<code>int math_asin(int sin_value100)</code>
Description	Calculates the arcus sine function
Parameters	int sin_value - the actual sine value, multiplied by 100
Returns	Integer, representing the angle (in radian units) that has the sine value of <code>sine_value</code>

Function	<code>int math_cos(int rad100)</code>
Description	Calculates the cosine function
Parameters	int rad100 - the value is in radians, multiplied by 100
Returns	An integer value, multiplied by 100

Function	<code>int math_acos(int cos_value100)</code>
Description	Calculates the arcus cosine function
Parameters	<code>int cos_value</code> – the actual cosine value, multiplied by 100
Returns	Integer, representing the angle (in radian units) that has the cosine value of <code>cosine_value</code>

Function	<code>int math_tan(int rad100)</code>
Description	Calculates the tanges function
Parameters	<code>int rad100</code> – the value is in radians, multiplied by 100
Returns	An integer value, multiplied by 100

Function	<code>int math_atan(int tan_value100)</code>
Description	Calculates the arcus tangens function
Parameters	<code>int tan_value</code> – the actual tanges value, multiplied by 100
Returns	Integer, representing the angle (in radian units) that has the tanges value of <code>tan_value</code>

## Simple Graphics

Includes functions for drawing basic shapes onto the screen

Function	<code>void fill_color(int red,                   int green,                   int blue)</code>
Description	Sets the fill color (used for drawing rectangles, ellipses, circles)
Parameters	<code>int red</code> – red color value  <code>int green</code> – green color value  <code>int blue</code> – blue color value
Returns	nothing

Function	<code>void stroke_color(int red, int green, int blue)</code>
Description	Sets the border color for the rectangle, circle, and also is used for color to draw a single line
Parameters	<p><code>int red</code> - red color value</p> <p><code>int green</code> - green color value</p> <p><code>int blue</code> - blue color value</p>
Returns	nothing

Function	<code>void rectangle(int x, int y, int width, int height)</code>
Description	Draws a rectangle with a top-left corner placed on position (x, y) and the given width and height
Parameters	<p><code>int x</code> - the x position</p> <p><code>int y</code> - the y position</p> <p><code>int width</code> - width of rectangle</p> <p><code>int height</code> - height of rectangle</p>
Returns	nothing

Function	<code>void circle(int x, int y, int width, int height)</code>
Description	Draws a circle which center is placed on position (x, y) and the given width or height as radius (the bigger one is taken)
Parameters	<p><code>int x</code> - the x position</p> <p><code>int y</code> - the y position</p> <p><code>int width</code> - radius</p>

	<code>int height - radius</code>
Returns	nothing

Function	<code>void line(int start_x,           int start_y,           int end_x,           int end_y)</code>
Description	Draws a line starting from position (start_x, start_y) and ends in (end_x, end_y)
Parameters	<code>int start_x</code> - the start x position  <code>int start_y</code> - the start y position  <code>int end_x</code> - the start x position  <code>int end_y</code> - the start y position
Returns	nothing

Function	<code>void ellipse(int x,              int y,              int width,              int height,              int arc_start,              int arc_end)</code>
Description	Draws an ellipse which center is placed on position (x, y) and given width and height. The arc_start and arc_end parameters are used for specifying start and end arc degrees
Parameters	<code>int x</code> - the x position  <code>int y</code> - the y position  <code>int width</code> - radius  <code>int height</code> - radius  <code>int arc_start</code> - arc start degrees  <code>int arc_end</code> - arc end degrees
Returns	nothing

Function	<code>int get_pixel_red(int x, int y)</code>
Description	Gets the decimal red value of the pixel positioned on coordinates (x, y)
Parameters	int x            - the x position  int y            - the y position
Returns	integer value representing the red pixel value

Function	<code>int get_pixel_green(int x, int y)</code>
Description	Gets the decimal green value of the pixel positioned on coordinates (x, y)
Parameters	int x            - the x position  int y            - the y position
Returns	integer value representing the green pixel value

Function	<code>int get_pixel_blue(int x, int y)</code>
Description	Gets the decimal blue value of the pixel positioned on coordinates (x, y)
Parameters	int x            - the x position  int y            - the y position
Returns	integer value representing the blue pixel value

Function	<code>void set_pixel_red(int x, int y)</code>
Description	Sets a decimal red value of the pixel positioned on coordinates (x, y)
Parameters	int x            - the x position

	<code>int y</code> – the <code>y</code> position
Returns	nothing

Function	<code>void set_pixel_green(int x, int y)</code>
Description	Sets a decimal green value of the pixel positioned on coordinates (x, y)
Parameters	<code>int x</code> – the <code>x</code> position  <code>int y</code> – the <code>y</code> position
Returns	nothing

Function	<code>void set_pixel(int x, int y, int red, int green, int blue)</code>
Description	Sets the color of the pixel positioned on coordinates (x, y)
Parameters	<code>int x</code> – the <code>x</code> position  <code>int y</code> – the <code>y</code> position  <code>int red</code> – the <code>red</code> color value  <code>int green</code> – the <code>green</code> color value  <code>int blue</code> – the <code>blue</code> color value
Returns	nothing

## Virtual Display Management

These functions affect the virtual screen

Function	<code>void display_size(int new_width, int new_height)</code>
----------	---

Description	Sets new size for the virtual display screen
Parameters	<pre>int new_width    - new width value int new_height   - new height value</pre>
Returns	nothing

## Debug Facilities

Includes functions used to provide debug information

Function	<b>void info(string text)</b>
Description	Dumps message (lowest level of warning notifications). Could be configured where to dump the message. Doesn't exit after the message dump.
Parameters	<code>string text</code> – the text to be printed
Returns	nothing

Function	<b>void note(string text)</b>
Description	Dumps message (higher level of warning notifications than “info”). Could be configured where to dump the message (see “Imagen Script Interpreter Configuration Files”). Doesn't exit after the message dump.
Parameters	<code>string text</code> – the text to be printed
Returns	nothing

Function	<b>void warning(string text)</b>
Description	Dumps message (higher level of warning notifications than “note”). Could be configured where to dump the message (see “Imagen Script Interpreter Configuration Files”). Doesn't exit after the message dump.
Parameters	<code>string text</code> – the text to be printed

Returns	nothing

Function	<code>void error(string text)</code>
Description	Dumps message and exits.
Parameters	<code>string text</code> – the text to be printed
Returns	nothing

## Document TODO