



VERSION 2.0.1

Remarks

Documentation Version 12, June 18, 2008.

Creator

Douglas J. Robar, Percipient Studios.

Copyright

Copyright © 2006-2008 Percipient Studios. All Rights Reserved.

Overview

No one wants to open Photoshop to edit every image before putting it online.

Do you need to create photo galleries, screenshot flip books, staff profile or directory pages? Would you like to automatically create graphical text from any font on your server for headlines or menus? Do you need to protect your site's images with a watermark? Don't waste your valuable time manually creating, resizing, or editing each image in Photoshop.

ImageGen is an easy-to-use application for your ASP.NET 2.0 website that automatically resizes photos, screenshots, and images from icon to thumbnail to full-screen sizes. *ImageGen* can create text graphics as well as overlay text on images, which is particularly useful for making image-based navigation with styled dynamic text on top. And *ImageGen* can automatically protect and brand your images with watermarks and corporate logos.

With more than 30 options, *ImageGen* can create hundreds of thousands of variations to meet virtually any need. *ImageGen* is easy to install and use on your website. *ImageGen* creates extremely high-quality images quickly. And *ImageGen*'s advanced caching means images are served immediately for fast and responsive websites.

With *ImageGen*, you can upload and go!

Feature Summary

	ImageGen Basic	ImageGen Professional
Runs on all ASP.NET 2.0 sites, including umbraco-based sites	✓	✓
Compatible with <i>ImageGen</i> 1.x for easy upgrades	✓	✓
Highest quality output in PNG, JPG, GIF, and BMP formats	✓	✓
Server caching of generated images	✓	✓
Browser caching improved with ETags and content expiration headers	✓	✓
Resize images to a specified height, width, or both	✓	✓
Keep image proportions when resizing	✓	✓
Create text graphics using any font, style, color, and size	✓	✓
Wrap long text into multiple lines to fit image width	✓	✓
Anti-alias text to avoid “jaggies”	✓	✓
Place borders around text and images (any color or size)	✓	✓
Specify the background color	✓	✓
Select JPG compression to maximize image quality or minimize file size	✓	✓
Create transparent GIF and PNG images	✓	✓
Display an alternate image if the primary image is unavailable (such as a generic “photo not available” image in a staff directory list)	✓	✓
Display a different image depending on the visitor’s browser (an easy way to handle the transparent PNG bug in IE6, for instance)		✓
Rotate text and images		✓
Overlay watermarks and logo images (PNG alpha channels supported)		✓
Prevent images from being size larger than a maximum height or width		✓
Prevent images from being scaled up beyond their original size		✓
Mirror and Flip images horizontally, vertically, or both		✓
Crop images		✓
Convert color images to grayscale		✓
Intelligent “Save As” filenames		✓
Simplify urls and increase security with pre-defined classes		✓
Automatically apply <i>ImageGen</i> settings to all images in a specified folder		✓
	Free!	199 USD per domain¹

¹ Academic and registered Non-Profit organizations may be eligible for discounts. Discounts for multi-domain purchases are available.

Installing ImageGen

UPGRADING FROM PRIOR VERSIONS OF IMAGEGEN

For users who have an earlier version of *ImageGen* installed, upgrading is simply a matter of deleting any `Cached` folders on the web server and then following the regular installation instructions to update your files.

ImageGen now uses an `.ashx` file rather than the previous `.aspx` file. An `ImageGen.aspx` file is provided for backward compatibility with existing sites, but you will enjoy maximum performance by using the `ImageGen.ashx` file instead. If possible, update your scripts, macros, templates, or pages to use the new `ImageGen.ashx` file.

INSTALLING IN UMBRACO WEBSITES

Install *ImageGen* in your umbraco-based website by installing the *ImageGen 2.0 Package* from the *Developer* section of the umbraco administration interface.

INSTALLING IN ASP.NET WEBSITES

Extract the files from the ImageGen 2.0 Package zip file as shown below. You may place `ImageGen.ashx` and `ImageGen.aspx` in any suitable folder on your website, but whatever folder you choose, you would use its folder name in place of `/umbraco` in the examples shown in the rest of this document.

1. Copy `ImageGen.dll` to the `/bin` folder
2. Copy `ImageGen.ashx` to the `/umbraco` folder
3. *Optional:* copy `ImageGen.aspx` to the `/umbraco` folder
(for backward compatibility with *ImageGen 1.x*)
4. Copy the sample `ImageGen.sample.config` file to the `/config` folder

ACTIVATING PROFESSIONAL FEATURES

In order activate *ImageGen's* professional features you will need to purchase a registration key for your domain and then update two configuration files as follows:

1. Rename or copy the sample `/config/ImageGen.sample.config` file to `/config/ImageGen.config`
2. Enter the registration Key for your domain(s), similar to that shown below, in the `ImageGen.config` file:

```
<ImageGenConfiguration>
  <Registration>
    <Key domain="example.com">C90FE21F1447D822BB4D834BC46F7D89A1376272</Key>
    <Key domain="sample.com">55FD8FD97A199AB3082BC7DC11AE5F743B092C55</Key>
  </Registration>
  ...
</ImageGenConfiguration>
```

3. Update the `/web.config` file by adding the following entries to the `<configuration>` section:

```
<configuration>
  ...
  <configSections>
    <!-- ImageGen Professional -->
    <section name="ImageGenConfiguration" type="ImageGen.ImageGenConfigurationHandler,ImageGen"/>
  </configSections>

  <!-- ImageGen Professional -->
  <ImageGenConfiguration configSource="config/ImageGen.config"/>
  ...
</configuration>
```

VERIFYING INSTALLATION

```
http://localhost/umbraco/ImageGen.ashx?version
```

ImageGen will show the version installed as well as any domains for which you have a valid registration key. The output will be similar to the following:

```
ImageGen version 2.0.xxxx.xxxx
(Professional features enabled for localhost)
(Professional features enabled for example.com)
(Professional features enabled for sample.com)
```

Notice that the professional features are always enabled when running from `localhost`. This allows you to try the professional features on your local machine without needing to buy a registration key first. You will need to purchase a registration key for your domain(s) in order to use the professional features from the internet or from other machines on your local network.

Using ImageGen

When *ImageGen* is called, it compares the requested parameters with those of previous requests (saved in `index.xml`) and immediately sends a previously generated image from its cache, if available. If no prior request has the same parameters as those requested, or if the source image has changed (different file size or modification date), or if the cached image has been deleted, *ImageGen* will dynamically generate the requested image, send it to the client, and save it to its cache for future re-use. This greatly reduces the processor burden on the web server, keeping the site responsive to visitors.

ImageGen will place its cached images in a `Cached` folder beneath the folder containing the image requested. Text graphics are always saved to the `/data/Cached` folder. These folders must be writable by the owner of the application pool for your website.

You may safely delete any `Cached` folders and associated `xml` files at any time; *ImageGen* will create a new `Cached` folder and `xml` files and generate new images for the cache when the next request occurs.

BASIC EXAMPLES

Hello, World!

If no parameters are specified, *ImageGen* will display a simple "Hello, World!" image:

```
http://localhost/umbraco/ImageGen.ashx
```

A large, bold, black text graphic that reads "Hello, world!". The text is centered and occupies most of the width of the light gray background box.

Create a Text Graphic with Unicode Characters

Unicode characters are supported, including right-to-left languages, with automatic wrapping (provided the selected font supports those characters):

```
http://localhost/umbraco/ImageGen.ashx?text=בראשית  
-or-  

```

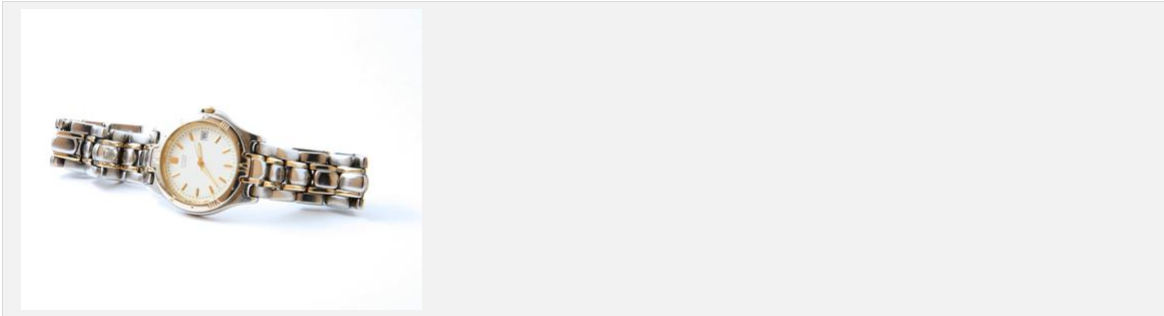
A large, bold, black text graphic that reads "בראשית" (Bereshit) in Hebrew. The text is centered and occupies most of the width of the light gray background box.

Create a Thumbnail Image

To create a 200 pixel wide thumbnail of a large image stored in `/media/watch.jpg`:

```
http://localhost/umbraco/ImageGen.ashx?image=/media/watch.jpg&width=200  
-or-  

```

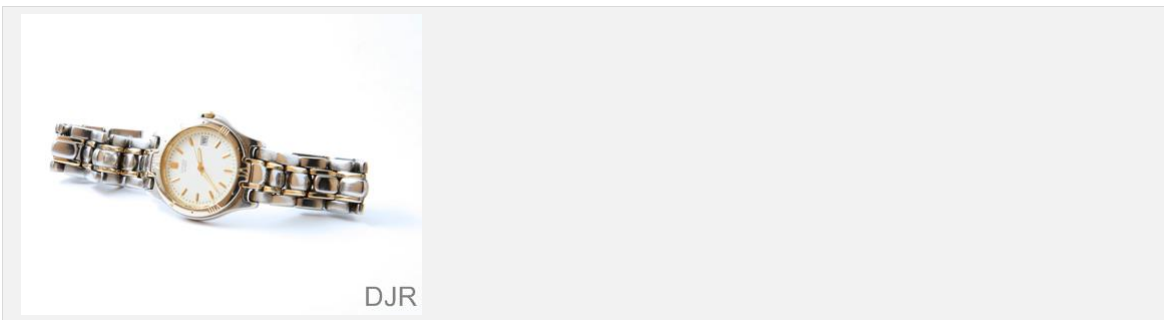


Place Text on an Image

To brand a thumbnail image with the photographer's initials in the bottom-right corner of the thumbnail:

```
http://localhost/umbraco/ImageGen.ashx?image=/media/watch.jpg&width=200 < ⤴  
&text=DJR&fontsize=10&fontcolor=gray&align=right&valign=bottom  
-or-  

```



ADVANCED EXAMPLES

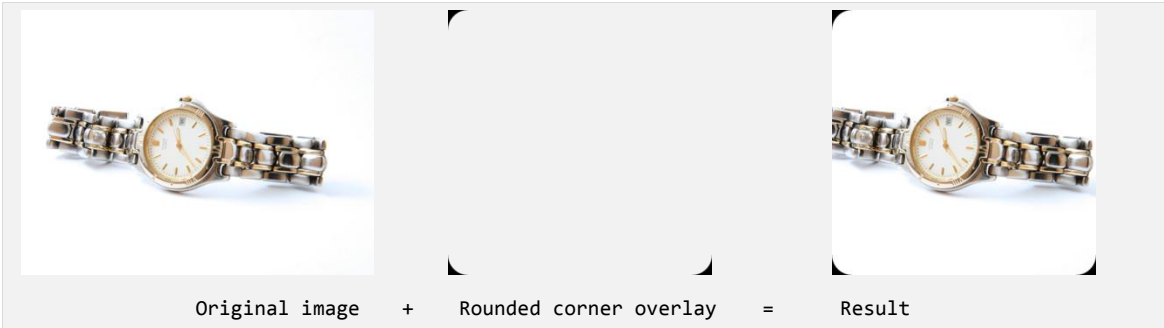
Rounded Corners

To round the corners of your image, create an overlay image with rounded corners the same color as the background of your website and the center transparent. For this example, the overlay rounds three of the corners and leaves one square.

```
http://localhost/umbraco/ImageGen.ashx?image=/media/watch.jpg&width=200&height=200 < ⤴  
&overlayimage=/media/rounded-corners.png&align=middle&valign=center&crop=resize  
-or-
```

```

```



When placed on a black background, the image will appear to have three rounded corners.



Grunge Borders

Create an overlay image with a transparent center. This is best done in Photoshop or a similar program.²

```
http://localhost/umbraco/ImageGen.ashx?image=/media/watch.jpg&width=200&height=200 ←  
&overlayimage=/media/grunge-border.png&align=middle&valign=center&crop=resize  
-or-  

```



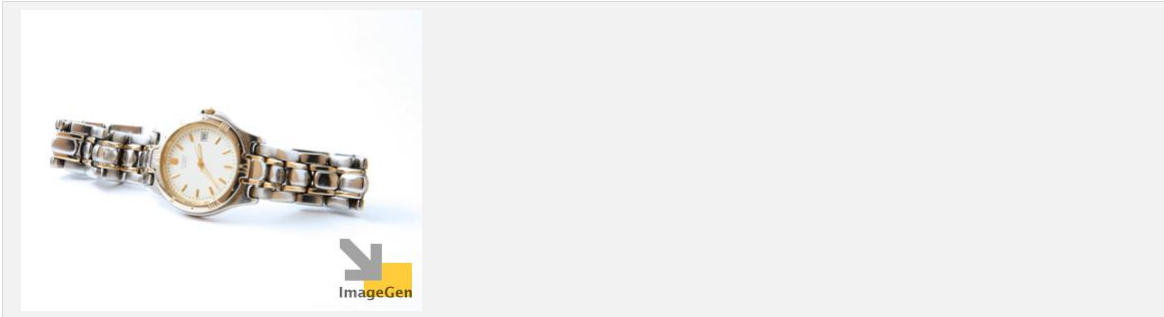
Watermark your Images

To brand a thumbnail image with a logo in the bottom-right corner of the thumbnail:

² For instruction and inspiration, http://andrearusky.deviantart.com/gallery/#_browse/resources and <http://www.640pixels.com/articles/free-photo-borders-for-photoshop.aspx>


```
http://localhost/umbraco/ImageGen.ashx?image=/media/big.jpg&width=100 ↵  
&overlayimage=/media/logo.png&align=right&valign=bottom  
-or-  

```

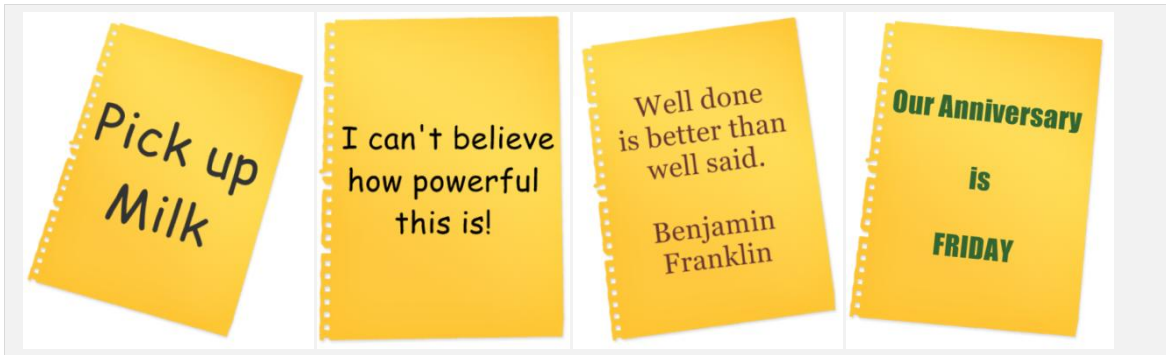


Random Notes

Because you can easily place auto-wrapping text on top of an image, as well as rotate the image and text, you can create a wide variety of effects. Here we'll place text on an image of paper torn from a notebook. By creating multiple images with differing text and rotating each a random amount you could easily create a collage effect.

```
http://localhost/umbraco/ImageGen.ashx?image=/media/paper.png&width=400 ↵  
&align=middle&valign=center&rotate=17&text=Pick+up+Milk  
-or-  

```



This effect can be enhanced by setting the output format to PNG and making the background color transparent, and then using CSS to position the images, even overlapping them slightly.

UMBRACO TEMPLATE AND XSLT EXAMPLES

Create a Text Graphic of the Page's Title (in a template)

In this example, we want to create a text image of the page's title on every page of our web site. In an Umbraco Template, enter the following

```
" ↵
alt=<?UMBRACO_GETITEM field="pageTitle"/> />
```

Create a Thumbnail (via XSLT)

To create a 100 pixel wide thumbnail of a large image via XSLT, code such as the following would be used, possibly within an `<xsl:for-each` loop of all the images in a portion of the media or content sections.

```
<img>
  <xsl:attribute name="src">
    <xsl:text>/umbraco/ImageGen.ashx?image=</xsl:text>
    <xsl:value-of select="$image"/>
    <xsl:text>&width=100</xsl:text>
  </xsl:attribute>
</img>
```

Create a Thumbnail with a Link to the Original Image (via XSLT)

To create a 100 pixel wide thumbnail that is also a link to the original image via XSLT, code such as the following would be used, possibly within an `<xsl:for-each` loop of all the images in a folder.

```
<a>
  <xsl:attribute name="href">
    <xsl:value-of select="$image"/>
  </xsl:attribute>
  <img>
    <xsl:attribute name="src">
      <xsl:text>/umbraco/ImageGen.ashx?image=</xsl:text>
      <xsl:value-of select="$image" />
      <xsl:text>&width=100</xsl:text>
      <xsl:text>&height=100</xsl:text>
      <xsl:text>&constrain=true</xsl:text>
    </xsl:attribute>
  </img>
</a>
```

ImageGen Reference

Align The horizontal position of text-only image if `width` is also specified. The horizontal position of text or an overlay image atop an image, even if `width` is not specified.

Class Notation: `<Align>Left</Align>`

Options: `Left, Center, Right`

Default: `Left`

Also See: `VAlign`



AllowUpsizing Allow or disallow an image to be resized larger than its original width and height. If `AllowUpsizing` is set to false, the image produced by *ImageGen* will not exceed its original width or height, even if the `width` or `Height` parameters specified are greater than the dimensions of the original image.

Class Notation: `<AllowUpsizing>True</AllowUpsizing>`

Options: `True, False`

Default: `True`

Also See: `MaxHeight, MaxWidth`



AltBrowser Return `AltImage` if the user's browser matches the specified `AltBrowser` criteria. This is especially useful for displaying a .GIF image to older browsers but a .PNG to newer browsers. `AltBrowser` has no effect if `AltImage` is not specified. Limited regular expression matching is supported. Separate multiple `AltBrowser` parameters with a semi-colon. Note that using `AltBrowser` will slow down the performance of *ImageGen* slightly.

Class Notation: `<AltBrowser>Browser:IE</AltBrowser>`

Examples: `Browser:Gecko`
`Browser:IE;MajorVersion:^[1-6]$`
`Type:IE5;Platform:MacPPC`

Note: Common parameters include:

	Firefox 2.0	IE7	IE6
Id	mozillafirefox	ie6to9	ie6to9
Type	Firefox	IE7	IE6
Browser	Gecko	IE	IE
Version	1.8	7.0	6.0
MajorVersion	1	7	6
MinorVersion	0.8	0	0
Platform	WinXP	WinXP	WinNT
EcmaScriptVersion	1.5	1.2	1.2

Cookies	True	True	True
VBScript	False	True	True
JavaScript	True	True	True

AltImage	<p>If <code>Image</code> is not found on the server, <code>AltImage</code> is used instead. This is especially useful for displaying a generic “photo not available” image in a staff directory list, for instance. The <code>AltImage</code> can also be activated by using the <code>AltBrowser</code> parameter. Specify the full path and filename on the web server to an alternate image.</p> <p>Class Notation: <code><AltImage>waterfall.png</AltImage></code> Also See: <code>AltBrowser</code>, <code>Class (ImageBaseDir property)</code></p>
-----------------	---

AntiAlias	<p>Remove “jaggies” around the edges of text. The <code>BgColor</code> specifies the color against which the text is anti-aliased.</p> <p>Class Notation: <code><AntiAlias>True</AntiAlias></code> Options: <code>True</code>, <code>False</code> Default: <code>True</code> Also See: <code>BgColor</code></p>
------------------	--

BgColor	<p>The background color to place behind the text or image. <code>BgColor</code> is only visible around an image if <code>Pad=True</code> or the image is rotated. Colors can be specified by their hex value (without the # symbol), or by name. The named colors can be found at: http://msdn.microsoft.com/en-us/library/system.drawing.knowncolor.aspx</p> <p>Class Notation: <code><BgColor>FFFFFF</BgColor></code> Default: <code>white (FFFFFF)</code>. Also See: <code>AntiAlias</code>, <code>Transparent</code></p>
----------------	--

Border	<p>The width of a border to place around the edge of an image or text graphic.</p> <p>Class Notation: <code><Border>0</Border></code> Default: <code>0</code></p>
---------------	--

BorderColor	<p>The border color. Colors can be specified by their hex value (without the # symbol), or by name. The named colors can be found at: http://msdn.microsoft.com/en-us/library/system.drawing.knowncolor.aspx</p> <p>Class Notation: <code><BorderColor>000000</BorderColor></code> Default: <code>black (000000)</code> Also See: <code>Border</code></p>
--------------------	---

Automatically apply multiple *ImageGen* parameters by specifying a single `Class` parameter on the querystring. Class settings are defined in the `ImageGen.config` file. Any querystring parameter can also be specified in a class definition, as well as some parameters that can only be specified in a class for security reasons.

Using classes is especially useful for obfuscating the location of your images, ensuring that watermarked or branded images are not modified, and otherwise protecting your site from malicious users attempting to alter the querystring and gain access to images you wish to protect. While no users have reported such incidents in the wild, it is best to protect one's assets and server resources.

An example `ImageGen.config` with a default, named, and folder class is shown below:

```
<ImageGenConfiguration>
...
<Class Name="default" OverridesQueryString="true">
  <ImageBaseDir>/media</ImageBaseDir>
  <AllowUpsizing>>false</AllowUpsizing>
  <MaxHeight>800</MaxHeight>
  <MaxWidth>800</MaxWidth>
  <OverlayImage>watermark.png</OverlayImage>
</Class>
<Class Name="Thumbnail" OverridesQueryString="true">
  <InheritFromClass>default</InheritFromClass>
  <Width>200</Width>
  <Height>200</Height>
  <Constrain>true</Constrain>
</Class>
<Class Folder="/products" OverridesQueryString="true">
  <Height>100</Height>
  <Border>2</Border>
</Class>
...
</ImageGenConfiguration>
```

Note that all XML tags in the `ImageGen.config` file are case sensitive, though their values are not.

Consider the following example that demonstrates the benefits of using classes:

```
http://localhost/umbraco/ImageGen.ashx?image=big.jpg&class=thumbnail
-or-

```

Without classes, you would need the following:

```
http://localhost/umbraco/ImageGen.ashx?image=/media/big.jpg &
&overlay=/media/watermark.png&width=200&height=200&constrain=true
-or-

```

Without classes the command string is much longer and it reveals important information about the image (such as its original location at /media/big.jpg). A malicious website visitor could use this information to request the original image. Or he could change the settings on the querystring to override your preferred settings, even removing the watermark overlay. With classes and the `OverridesQueryString="true"` setting, the location of the original file is not revealed, the watermark cannot be removed, nor can the size altered.

It is best practice to specify strict values for all parameters in the `Default` class. This effectively limits *ImageGen* to a minimum behavior unless a named class is explicitly requested, which will also have very specific parameters. This protects your assets.

If a `Default` class exists, these settings are applied to all *ImageGen* operations, even if no `Class` parameter appears on the querystring. That is, the `Default` class is always applied unless another class is explicitly specified.

If a named class is specified on the querystring and the class also exists in the config file, its settings are used. If the requested class does not exist in the config file then *ImageGen* will behave as though no class were specified, even applying the `Default` class (if defined).

Folder classes are applied automatically when the first characters of the `Folder=` property exactly match the first characters of the `image=` querystring parameter. You may specify an entire folder (and all images contained in that folder) with `Folder="/screenshots"` or a single file, such as `Folder="logo.gif"`. Wildcard characters are not supported in folder classes. Folder classes do not apply to sub-folders; you must have a class for each folder.

A named class's properties take precedence over a folder class, which in turn takes precedence over the default class.

The `ImageBaseDir` parameter (can only be set in `ImageGen.config`) specifies a path fragment that is silently prepended to `Image=`, `AltImage=`, and `OverlayImage=` querystring parameters. This prevents the full path of the image from being revealed on the querystring, as well as shortening the querystring.

The `InheritFromClass` parameter (can only be set in `ImageGen.config`) lets you cascade settings from the specified class, allowing you to override any inherited parameter without repeating all the parameters of the parent class.

An inherited class parameter can be unset in a class that inherits a parameter by specifying an empty parameter, such as `<OverlayImage></OverlayImage>`.

The `OverridesQueryString` parameter (can only be set in `ImageGen.config`; default value is `true` if not specified) ensures that any settings specified in the class will be applied, even if different values are specified on the `queryString`. If `OverridesQueryString` is set to `false`, parameter values on the `queryString` will be used even if a parameter is also specified in the class. Avoid setting `OverridesQueryString` to `false`, because this eliminates protection against potentially malicious users.



ColorMode

Convert a color image to grayscale. Note that black and white or grayscale images cannot be colorized by setting `ColorMode` to `Color`. Converting images to grayscale is a processor-intensive operation, particularly for large images. Therefore, it is recommended that you convert the original image to grayscale if possible rather than relying on the `ColorMode` parameter.

Class Notation: `<ColorMode>Color</ColorMode>`

Options: `Color, Grayscale`

Default: `Color`

Compression

For JPEG images, `Compression` specifies the file size vs. quality trade-off. `0` provides minimum file size (and minimum quality) while `100` provides maximum quality (and maximum file size). The default value produces a good quality image of reasonable size.

Class Notation: `<Compression>80</Compression>`

Options: `0 to 100`

Default: `80`

Constrain

Retain the proportions of the original image when resizing to fit within the specified `Height` and `Width`. Thus, one of the dimensions of the resized image will be exactly that specified by the `Width` or `Height` parameter and its other dimension will be smaller. If `Constrain` is set to `false`, the resized image will fill the specified `Height` and `Width` without regard for the original image's proportions. `Constrain` is only used when both `Width` and `Height` are specified and `Pad` is set to `false`; otherwise it is ignored.

Class Notation: `<Constrain>False</Constrain>`

Options: `True, False`

Default: `False`

Also See: `Pad`



Crop the image to the specified `Height` and `Width`, either before or after resizing. If `resize` is specified, the image is first resized and any portion of the image that extends beyond the specified dimensions is cropped off, resulting in an image of the exact dimensions specified by the `Height` and `Width` parameters. This shows the maximum amount of the original image. If `noresize` is specified, the original, un-resized image is cropped to the dimensions specified and no resizing takes place. The portion of the original image that remains visible is based on the `Align` and `VAlign` settings.

Class Notation: `<Crop>Resize</Crop>`
Options: `Resize`, `Noresize`
Also See: `Constrain`, `Pad`, `Align`, `VAlign`



Mirror the image horizontally, vertically, or both. Using `Flip` and `OverlayImage` you can create that cool “web 2.0” reflection that is so popular.

Class Notation: `<Flip>X</Flip>`
Options: `X`, `Y`, `XY`
Also See: `Rotate`

Font

The font used for rendering text. Specify the font name of any font installed on your web server, or the path and filename of a TrueType or Open Type font located on your web server. If the specified font cannot be found, the default font is used. Note that Open Type support is limited to those fonts with TrueType outlines rather than Postscript outlines. This is a limitation of the .NET Framework. Also note that individual font files contain a single font style (regular, bold, italic, bold-italic). Therefore, if you specify a font by filename you must also specify the `FontStyle` to match the style contained in the font file.

Class Notation: `Arial`
Default: `Arial`
Examples: `Font=Verdana` (for installed fonts)
`Font=/fonts/myfont.ttf` (for font files not installed on the server)

FontColor

The color of the text created by *ImageGen*. Colors can be specified by their hex value (without the # symbol), or by name. The named colors can be found at:

<http://msdn.microsoft.com/en-us/library/system.drawing.knowncolor.aspx>

Class Notation: `<FontColor>000000</FontColor>`
Default: `black (000000)`

FontSize	<p>The size of the text created by <i>ImageGen</i>. If the <code>Text</code> and <code>FontSize</code> parameters generate text too large for the specified <code>width</code> and <code>Height</code>, no text will appear. If the <code>Height</code> is not specified, text will automatically wrap to fit the specified <code>width</code>.</p> <p>Class Notation: <code><FontSize>32</FontSize></code> Default: 32</p>
FontStyle	<p>Style the font. Any combination of <code>FontStyle</code>'s can be used, but must be separated by plus signs (%2B is the URL-escaped plus sign for use with querystrings). Not all styles are supported by all fonts, especially those fonts which are not installed on the web server.</p> <p>Class Notation: <code><FontStyle>Bold+Italic</FontStyle></code> Options: Regular, Bold, Italic, Underline, Strikeout Default: Regular Examples: <code>FontStyle=Bold</code> <code>FontStyle=Bold%2BItalic%2BUnderline</code></p>
Format	<p>The image format for the created or resized image.</p> <p>Class Notation: <code><Format>GIF</Format></code> Options: JPEG, JPG, GIF, PNG, BMP Default: The format of the original image when resizing, or GIF for text-only graphics</p>
Height	<p>The height of the created or resized image. The image will resize proportionally to the desired height unless <code>width</code> is also specified. If both <code>Height</code> and <code>width</code> are specified, the image is resized to those dimensions exactly. If <code>Constrain</code> or <code>Pad</code> is specified in addition to <code>Height</code> and <code>width</code>, the image is resized proportionally and padded to fill any extra space.</p> <p>Class Notation: <code><Height>100</Height></code> Default: The height of the original image when resizing, or to the height required to contain the specified text graphic Also See: <code>Constrain</code>, <code>Pad</code>, <code>Width</code></p>
Image	<p>The full path and filename on the web server to an image to resize. <i>ImageGen</i> can also automatically select the {first} image or a {random} image from a folder of images.</p> <p>Class Notation: <code><Image>portrait.jpg</Image></code> Examples: <code>/media/big.gif</code> <code>/media/5206/big.jpg</code> <code>/images/screenshots/{first}</code> <code>/images/{random}</code> Also See: <code>AltBrowser</code>, <code>ImageBaseDir</code>, <code>Class</code> (<code>ImageBaseDir</code> property)</p>

 **MaxHeight**

The maximum height of any image. If the `Height` parameter is larger than the `MaxHeight` value, the `Height` will be treated as though it were equal to the `MaxHeight` setting. This is an extremely useful parameter for use in classes to avoid the creation of inordinately large images.

Class Notation: `<MaxHeight>800</MaxHeight>`

Also See: `AllowUpsizing`

 **MaxWidth**

The maximum width of any image. If the `width` parameter is larger than the `MaxWidth` value, the `width` will be treated as though it were equal to the `MaxWidth` setting. This is an extremely useful parameter for use in classes to avoid the creation of inordinately large images.

Class Notation: `<MaxWidth>800</MaxWidth>`

Also See: `AllowUpsizing`


NoCache

Generate a new image, even if a previously-cached image exists with the same properties. If a previously-cached image exists with the same parameters requested, `NoCache` will force generation of a new image and replace the previously cached file with a new one.

Class Notation: `<NoCache>False</NoCache>`

Options: `True, False`


Default: `False`

 **OverlayImage**

The full path and filename to an image on the web server to overlay on top of the `Image` or `Text`. As with text atop images, `Align` and `VAAlign` parameters specify the position of the `OverlayImage` with respect to the underlying image. Best results are achieved with transparent PNG overlay images utilizing the alpha transparency channel. Note that `OverlayImage`'s are neither resized nor rotated, but are placed at original size on top of the already-resized `Image`. You may want several different sizes of your watermark or logo for use with the various sizes of images you create with *ImageGen*.

Class Notation: `<OverlayImage>watermark.png</OverlayImage>`

Also See: `OverlayMargin`, `Class (ImageBaseDir property)`

 **OverlayMargin**

The distance the `OverlayImage` will appear from the edge specified by the `Align` and `VAAlign` parameters.

Class Notation: `<OverlayMargin>5</OverlayMargin>`

Also See: `Align`, `VAAlign`, `OverlayImage`

Pad

Fill the specified `width` and `Height` with the `BgColor` and place the (proportionally) resized image in the center of the available space. `Pad` is only useful when both `Height` and `width` are specified; otherwise it is ignored.

Class Notation: `<Pad>False</Pad>`

Options: `True, False`

Default: `False`

Also See: `BgColor, Constrain, Transparent`



Rotate

The number of degrees to rotate the image in a clockwise direction. As the image is rotated, the `BgColor` will fill any unused space. Note that any `OverlayImage` is not rotated, though `Text` is rotated, even if placed on top of an image.

Class Notation: `<Rotate>90</Rotate>`

Default: `0`

Also See: `BgColor, Transparent`



Text

The text to be rendered. If the text is wider than would fit in the width, the text is automatically wrapped to multiple lines. Place `\r\n` in the text string to force a line-break at that location.

Class Notation: `<Text>Hello, World!</Text>`

Examples: `Lorem%20ipsum`

`Lorem\r\nipsum`

Note: Special characters must be URL-encoded. These include:

<code><space></code>	Space	<code>%20</code>
<code>"</code>	Double quote	<code>%22</code>
<code>#</code>	Pound character	<code>%23</code>
<code>\$</code>	Dollar sign	<code>%24</code>
<code>%</code>	Percent character	<code>%25</code>
<code>&</code>	Ampersand	<code>%26</code>
<code>'</code>	Single quote	<code>%27</code>
<code>@</code>	At symbol	<code>%40</code>
<code>`</code>	Back tick/Grave accent	<code>%60</code>
<code>+</code>	Plus	<code>%2B</code>
<code>,</code>	Comma	<code>%2C</code>
<code>/</code>	Forward slash/Virgule	<code>%2F</code>
<code>:</code>	Colon	<code>%3A</code>
<code>;</code>	Semi-colon	<code>%3B</code>
<code><</code>	Less than symbol	<code>%3C</code>
<code>=</code>	Equals	<code>%3D</code>
<code>></code>	Great than symbol	<code>%3E</code>

?	Question mark	%3F
[Left square bracket	%5B
\	Backslash	%5C
]	Right square bracket	%5D
^	Caret	%5E
{	Left curly brace	%7B
	Vertical bar/Pipe	%7C
}	Right curly brace	%7D
~	Tilde	%7E

If passing values from a form, URL-encoding should happen automatically. Unicode characters are fully supported (with Unicode fonts) but may require URL-encoding.

Transparent Make the `BgColor` transparent for GIF and PNG images. Text rendered with `AntiAlias=True` calculates the anti-aliasing based on the `FontColor` and `BgColor`. The `BgColor` is then made transparent. Though primarily useful for text graphics, images can also make use of the `Transparent` parameter. Note that transparent GIFs may not retain their transparency when resized because of the potential for color shifting associated with the limited 8-bit GIF palette. The use of PNG images is recommended.

Class Notation: `<Transparent>True</Transparent>`
Options: `True, False`
Default: `True`

VAlign The vertical position of text-only image if `Height` is also specified. The vertical position of text or an overlay image atop an image, even if `Height` is not specified.

Class Notation: `<VAlign>Middle</VAlign>`
Options: `Top, Middle, Bottom`
Default: `Middle`
Also See: `Align`

Version Show the version of *ImageGen*, if `Version` is the only parameter specified. Otherwise, the parameter it is ignored.

Example: `http://localhost/ImageGen.ashx?version`

Width The Width of the created or resized image. The Image will resize proportionally to the desired width unless `Height` is also specified. If both `Height` and `width` are specified, the image is resized to those dimensions exactly. If `Pad` or `Constrain` is specified in addition to `Height` and `width`, the image is resized proportionally and padded to fill any extra space.

Class Notation: `<Width>100 </Width>`

Default: The width of the original image when resizing, or to the width required to contain the specified text graphic

Also See: Constrain, Height, Pad

License

The key points of the license are:

- You can install *ImageGen* on as many computers as you wish.
- You agree not to decompile, reverse engineer or disassemble, modify or create derivative works.
- You agree not embed or otherwise include *ImageGen* in your applications.

FULL LICENSE DETAILS

Percipient Studios, of Louisville, Kentucky is the owner, developer and sole copyright holder of this product, which is licensed—not sold—to you on a non-exclusive basis. *ImageGen* is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.

You may use the free version of *ImageGen* for as long as you like without ever having to acquire a registration key. However, it is recommended that you join the Percipient Studios mailing list so you can be kept informed of bug reports, usage tips and new releases as they become available.

The free version of *ImageGen* can be upgraded to enable additional features over and above those of the free product.

Each registration key covers a single second-level domain (e.g., example.com) and *ImageGen* can be installed on any machine or machines within that domain. You need a unique registration key for each second-level domain for which you want the full features enabled.

By installing and/or using *ImageGen*, you agree NOT to:

- (a) Decompile, reverse engineer or disassemble, modify or create derivative works based on *ImageGen* or the documentation in whole or in part.
- (b) Remove any copyright or other Percipient Studios proprietary notices.
- (c) Distribute any registration key for *ImageGen* to anyone other than the legally registered end user.
- (d) Rent or lease *ImageGen* to any other party.
- (e) Use a registration key that was not obtained directly from Percipient Studios.

You may transfer your *ImageGen* registration key to another person when transferring your domain only after receiving written authorization from Percipient Studios and only if the recipient agrees to be bound by the terms of this agreement.

Percipient Studios reserves the right to cancel the registration key(s) of any user who Percipient Studios determines is in violation of this agreement.

THE WARRANTIES IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE IS PROVIDED "AS IS" AND PERCIPIENT STUDIOS DISCLAIMS AND EXCLUDES ALL OTHER WARRANTIES. IN NO EVENT WILL PERCIPIENT SUTDIOS BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, EVEN IF WE HAVE KNOWLEDGE OF THE POTENTIAL LOSS OR DAMAGE.

If you are located in the United States this agreement is subject to the laws of the state of Kentucky. If you are located outside the United States local law may apply. Some states do not allow the exclusion of warranties, so the above exclusion may not apply to you.

You agree that you will not export or re-export *ImageGen* to any country, person, entity or end user subject to U.S.A. export restrictions.

Version History

Version	Date	Comments
2.0.1	June 18, 2008	<ul style="list-style-type: none">• Fixed bug with temporary xml files not being deleted.
2.0	May 26, 2008	<p><i>ImageGen Basic</i> and <i>Professional</i> released. Many new and improved features, such as:</p> <ul style="list-style-type: none">• Added superimposing images on top of one another for watermarking and branding purposes, with alpha transparency support.• Added <code>ImageGen.config</code> classes to increase security and shorten URLs.• Maximum image size can be set.• Upsizing of images can be disabled.• Color images can be converted to grayscale output.• Rotate, crop, and flip images.• Text wrapping.• Improved error handling for busy sites.
1.6	August 9, 2007	Added <code>AltBrowser</code> parameter.
1.5	February 28, 2007	Fixed edge interpolation bug.
1.4	December 14, 2006	Added <code>version</code> parameter. Added error handling in the event the <code>index.xml</code> file becomes corrupted.
1.3	July 1, 2006	Added <code>constrain</code> parameter to image resizing.
1.2	May 12, 2006	Minor bug fixes. Not released.
1.1	January 23, 2006	Changed default location of cached text graphics to the <code>/data</code> folder. Fixed bug attempting to render apostrophe or ampersand in text.
1.0	January 13, 2006	Initial release.