

Imagen

for Windows/Linux

Readme

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

Introduction

Imagen Netclime is a server-based image generation engine, designed to simplify the development of complex dynamic content sites and to reduce the time to market and increase the quality and flexibility of most web applications. Basically with Imagen you can load images, write any text over them, apply filters, resize/stretch/crop images and after that deliver the images in any web browser compatible format (GIF/JPG/PNG). All this is performed fully automatically on the server using a custom scripting language. You can pass parameters to scripts from HTML and thus control their behavior and reuse one and the same script for different applications. Imagen is equipped with a smart high performance cache, avoiding the multiple generation of one and the same image. Imagen works under both Windows / UNIX and can be installed as plugin to apache/IIS or can be used as a stand alone CGI-BIN application.

If you are a web site builder or a web engineer you will find the production of image rich web sites a lot easier using Imagen Netclime. If you are web production manager, your team can drastically increase the performance and quality of production, and add more flexibility to web site development process with Imagen Netclime. If you are a dynamic content management (DCM) developer, you can accomplish your DCM engine with a new powerful feature. All graphic elements can be controlled from your DCM system. If you are a web enthusiast, you can incorporate Imagen Netclime in your site and you will impress your friends with it.

Usage

Imagen Usage

Imagen is accessible through the following interfaces:

- ◆ Command Line
- ◆ CGI

These interfaces define the way Imagen retrieves the name of the script file that it has to process and the parameters that it has to pass to the script.

The output from Imagen is always in CGI form. Imagen dumps appropriate HTTP headers and after that dumps the content of the resulting image. Imagen can be configured to return error messages as plain text instead of image (or nothing since errors in most cases happen before the image is generated). Please refer to the Imagen Cache Usage

Imagen Cache Usage

As long as the Imagen Cache is a wrapper of Imagen, its usage syntax is equal to the Imagen one.

Imagen Cache provides additional features, which differ from the Imagen syntax:

Web-based Interface

Calling convention

- **Web server plugin:**
`http://my.server/path/script.img?param1=value1¶m2=value2...`

- **Standalone CGI:**
http://my.server/path/cache_executable?script.img¶m1=value1¶m2=value2...,
 where `cache_executable` is `imagenc.cgi` or `imagenc.exe`, depending on the platform.

Additional Commands

Command	Action
<code>cleanup</code>	Removes all cached files.
<code>lib_cmd=abc</code>	Sends the abc command to the used plugin through the Cache Plugin (Dispatcher) API.

For an example: <http://my.server/imagenc.cgi?cleanup>.

Command Line Interface

Argument	Action
<code>-cleanup</code>	Removes all cached files.
<code>-strip</code>	Perform smart cleanup which should free the specified percent of the max allowed disk space (see the Configuration section). This cleanup removes the most not accessed files. Note: You should not use this argument – it is used internally by Cache.
<code>-stats</code>	Shows cache statistics (number of files and disk space).
<code>query</code>	Executes the query just as it is passed through the web.

Imagen Cache Plugin (Dispatcher) Usage

Executing Imagen scripts is a resource-consuming process. It is a problem for the web server to generate a page with many Imagen images in it. The Dispatcher responsibility is to broadcast Imagen Cache requests to remote machines with installed Imagen in order to not slow down the web server.

The Dispatcher works as dynamic library under both Windows and Linux. As it is Imagen Cache plugin, it conforms the Cache Plugin API.

Here are the accepted commands through the Cache Plugin API (by the `lib_cmd` Cache argument):

Command	Description
<code>reset_rings</code>	Switch to ring 0.
<code>show_stats</code>	Show various statistics about the Dispatcher usage.

For example: http://server.com/folder/imagenc.img?lib_cmd=reset_rings. (`imagenc.img` file may not exist, if the web server configuration allows it)

Operation

Imagen Operation

During its operation Imagen passes through the following phases:

Startup Logging

Imagen stores one line containing the current time and invocation parameters in file named 'imagen.log' and located in the directory where the Imagen executable is located.

Load Configuration

The configuration file named 'imagen.cfg' located in the directory where the Imagen executable is located is parsed for configuration options.

Read script name and parameters

Imagen finds what interface is used for it invocation (command line or CGI) and reads its parameters from there.

Execute script

The script name is retrieved by the prescending phase. This phase loads the corresponding file and executes it. Note that if the script name represents relative path, the name becomes relative to the imagen executable. This is general rule: all relative paths are relative to this directory: images, fonts, relative paths read from the configuration. In other words Imagen works as its current directory is that directory.

The script should call the built in function `return_image` once to specify what is the resulting image.

Imagen Cache Operation

Here is a brief description of the operation flow of Imagen Cache, which is performed for each request:

- ◆ Get the query.
- ◆ If the file exists, it is outputted (cache hit).
- ◆ Otherwise (cache miss), the Imagen executable is called with the same query. The output is collected, stored in the file cache and outputted.

Error Handling section below for more information on this behavior.

Command-line Syntax

The command line syntax accepted by Imagen is as follows:

```
imagen script-name param1=value1 param2=value2 ... param3=value3
```

Where `script-name` is the name of the Imagen script to be executed and the `paramX=valueX` arguments specify optional variable assignments.

Note that the script name is relative to the directory where the Imagen executable is located, not the current directory.

A `param=value` pair is scanned for URL escape sequences which are translated back to the characters they represent. Then the right part of the '=' is evaluated as expression. New variable is created in the global scope with the name given at the left part of '=' and with the type of the expression. The result of the expression is then stored to the variable.

For more information about the structure of expressions and about scopes please refer to Imagen Script Language Reference.

For example the following invocation is valid in sh-compatible shells:

```
imagen script.img num=5 text=\"sample%20text\"
```

Note that the double quotation characters are escaped. Otherwise they will be interpreted by the shell and not passed to Imagen. The space is escaped by using URL escape sequence: '%20'. This is also sh trick since using space directly will split the parameter and will produce error.

Another equivalent invocation could be:

```
imagen script.img num=5 'text="sample text"'
```

URL Syntax

The URL Syntax that Imagen accepts is like the command line syntax, but with '&' used to separate parameters, as in regular CGI applications. For example the above command line invocation:

```
imagen script.img num=5 'text="sample text"'
```

Can be passed as URL this way:

```
http://server/path/imagen?script.img&num=5&text=%22sample%20text%22
```

The delimiter '&' can be passed in string constants only escaped as regular URL escape: %26.

Please note that depending on your platform and installation method `imagen` from the URL above may be something else. For example it could be `imagen.cgi` or `imagen.exe`.

Imagen Cache Plugin (Dispatcher) Operation

The Dispatcher operates at several rings. Each ring consists of several URLs to remote Imagens. By default, the Dispatcher uses ring 0. If there is no working machines at this ring, the next one is used. When there is no more rings, the local Imagen is used. The Dispatcher may be switched to ring 0 again by calling it with the appropriate argument.

Each URL in the ring has an integer number for its priority. Please refer to the *Algorithms* section for description of the priorities usage.

The Imagen on the remote machines may be installed in several ways – standalone Imagen Cache, Imagen Cache as a web server plugin, or Imagen without Imagen Cache. By this reason, there are two types of URL, which the Dispatcher accepts:

- URL for standalone generator. For example “`http://server.com/folder/imagenc.cgi`”. In this case, the Dispatcher uses the URL by the following way:
“`http://server.com/folder/imagenc.cgi?scriptname&arg1=val1&arg2=val2...`”
- URL for web server plugin. For example “`http://server.com/folder/`”. In this case, the Dispatcher uses the URL by the following way:
“`http://server.com/folder/scriptname?arg1=val1&arg2=val2...`”

Error Handling

Imagen Error Handling and Logging

Error Handling

Imagen recognizes several kinds of errors. They are classified in the following categories:

Severity code	Explanation
SEVERITY_INFO	Informational message (minimal severity level)
SEVERITY_NOTE	Note about something that is in almost all cases OK
SEVERITY_WARNING	Something in the script is valid, but probably is not the intent (may be logical error).
SEVERITY_ERROR	The script is invalid according to some rule (i.e. syntax error, or attempt to load image that does not exist).
SEVERITY_INTERNAL	Internal error. These are errors in Imagen. (maximal severity level)

From these SEVERITY_INFO and SEVERITY_NOTE are not surely error messages. They can be issued by scripts to take attention of some possible problem or to simply log script execution. This is up to the script developer. As consequence there are two kinds of error messages: *recoverable* and *fatal*. Fatal errors force Imagen to exit immediately (possibly storing the error message as described below) while recoverable errors allow execution to continue. Recoverable errors are those marked with the following severity codes:

SEVERITY_INFO, SEVERITY_NOTE and SEVERITY_WARNING. The script is capable of issuing recoverable messages directly through the runtime functions: `info()`, `note()` and `warning()`. The script can also issue messages with severity code of SEVERITY_ERROR by using the function `error()`. Scripts cannot issue SEVERITY_INTERNAL errors directly. These are issued by Imagen itself and correspond to errors in Imagen itself.

Error Logging

When error occurs Imagen can output the error to standard output with content type text/plain, it can also store in log file and send a mail. According to message severity the message can be sent to all possible locations, to some of them or to be discarded. Note that discarding fatal message means that the message will not be stored anywhere. Imagen still will refuse to continue execution when it encounters fatal error.

For full description of the possible options for error logging please refer to Imagen Configuration Files Reference.

Notes

Problem: Problem with finding “magic.mgk” configuration file

Synopsis: If you get a broken image while trying to load a .gif file, and clicking on “View Image” you got the following error message:

```
/path/script.img (XX): error: Magick: Unable to read magic configuration  
file (magic.mgk)
```

You probably use ImageMagick version 5.3.7 or later which requires “magic.mgk” to be copied in your `/path_to_apache/libexec` directory (or as an environment variable `MAGICK_HOME=/path/to/mgk_file`) in case you are using Imagen installed as Apache plugin. In other case put this configuration file in the directory where the Imagen executable is.

Problem: Rotate loses transparency of region

When a transparent region is rotated the transparency is lost but outside the rotated area there is transparency. Stupid workaround is to save the rotated image in another file (before saving set transparency) and to use this file.

Problem: ImageMagick 5.3.7 for Win and colors

There’s a problem with the 5.3.7 distribution of ImageMagick – the images generated by imagen are monochrome.

Problem: RedHat 7.0 and GCC 2.96

Imagen cannot be built with the non standart gcc-2.96 compiler found in the RedHat 7.0 distribution.