

Autotest

Autotest application

Documentation

July 14, 2001. Document Revision 2

Netclime Inc.

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

email: imagen@netclime.com

Overview

The Automatic QA System goals are:

- ◆ To provide sample tests and expected results for easy verification of Imagen operation.
- ◆ To provide scripts for automatic testing of Imagen with the sample tests and to check if the expected results are returned.

This system is very important when modifying the source code of the program. Although it cannot fully guarantee Imagen's consistence, it can help finding most of the problems in the program.

Test environment

Tests are organized into groups. Each group consists of set of configuration files and set of scripts stored in single directory. Some of the scripts may have additional file containing one or more sets of arguments to be passed to the script (one set per line). If there is no such file, the script will be tested once with no arguments.

Testing a group consists of executing Imagen with each configuration file for each script giving all script parameters. For each execution the standard output and error log (if generated) are compared with sample ones for exact match. If differences are found, the test is considered failed.

The testing can be done recursively: Each subdirectory that contains at least one configuration and script files is considered test group, and is then tested. The directories are tested in alphabetical order.

Test Groups

The different test groups are described here. The groups should be tested in the given order. The test environment is setup to meet this criterion.

Preconditions for all test levels

- ◆ Disable volatile info in output messages
 - ◆ Disable display of current time and environment
 - ◆ Disable dumping of internal program scopes

Begin: Basic configuration, interpretation and logging tests

These are the first tests to be executed. They test if Imagen can handle simple configuration and script files and if it can report errors at all.

This is the place where debug external routines are tested too as they will be used by the tests in next test groups.

Tests

- ◆ Test debug output external routines

Scan: Preprocessor tests

Parsing of lexemes (identifiers, numbers, string literals and operators) is taken into full detail here. The preprocessor is also responsible for ensuring the scripts versions are correct and for including other scripts.

- ◆ Version and include directives operation
- ◆ Comments
- ◆ Syntax errors at lexeme level

Lang: Language constructs tests

All language constructs are tested here. That includes:

- ◆ `if` statement with all possible kinds of constant expressions as conditions (all subsequent tests need working operator `if` in order to work)
- ◆ Variable allocation and assignment.
- ◆ Full if, while, do-while tests
- ◆ Function definitions and calling
- ◆ Scope resolution (for variables) and function calls
- ◆ Handling of parameters passed to scripts

External: External routines

Each documented external routine needs tests covering normal operation, boundary/critical/special cases and invalid cases (invalid input parameters).

Auto-Tester Scripts Reference

This section describes the usage of the auto testing modules. Currently there is only one implementation written as bash shell script to be launched from Linux boxes.

Quick Start

The script is named `autotest` and resides in the `autotest` directory of the Imagen distribution. To perform the tests go to the `autotest` directory and start:

```
autotest -r -i <<directory-containing-imagen>>
```

Where `<<directory-containing-imagen>>` is the directory containing the Imagen executable you wish to test. You should take into account that the tests temporary override the Imagen configuration file contained into that directory. Thus you shouldn't test Imagen instances that are currently in use (for example don't directly test Imagen executable placed into your `cgi-bin` directory. Instead first copy it into temporary empty folder, and try there).

When launched, the script displays one line for each test it makes, and also some helper lines (like what directory or configuration files are used). If the test fails, `[BAD LOG]` and/or `[BAD OUT]` appear next to the test stating that the log or the output differs from the expected one. At the end of its execution the total number of bad logs and outputs is printed.

Tests Organization

The `autotest` directory of the Imagen distribution contains the tester script (named `autotest`) and the scripts and configuration files arranged into subdirectories. Testing a directory consists of enumerating all Imagen configuration files there (they must have `.cfg` extension) and for each of them executing Imagen with all scripts (with `.img` extension). If, for given script there is no arguments file (file with the same name as the script, but with `.arg` prefix), the script will be executed once (per configuration file), with no arguments. If the arguments file is provided, the script will be executed with arguments passed from that file: Each line must contain the parameters, as they have to be given to Imagen through the web (but without the script name). For example:

```
param1=14&param2="string"
param1=7&param2="another string"
```

For each execution of Imagen, its output, and the log file it may have produced are recorded and compared to sample ones. The samples are into directory `.ok`, inside the test directory. The sample logs have extension `.log` and the sample outputs have extension `.out`. The names of the samples consist of the name of the script followed by optional configuration suffix and optional parameter suffix. The configuration suffix is appended when used configuration file different from `default.cfg` and consists of a hyphen followed by the name of the configuration file (without the extension).

The parameter suffix is added whenever an arguments file is provided and consists of `.p` followed by the line number of the arguments in the arguments file.

When differences are found, the generated by Imagen log and/or output file is recorded into the directory `.diff` (relative to the test dir) by following the same naming convention.

Note that if Imagen does not produce log file, but such one is expected, and error is signaled.

Restrictions in the configuration files

The following parameters cannot be setup by the test scripts (they are predefined during the tests): `dump_header`, `log_enabled`, `log_path`, `profiling_enabled` and `mailing_enabled`

All other parameters must be provided (see Imagen Configuration Files Reference for full description)

Creating New Tests

The `autotest` script allows some automation of the development of test scripts. When new test script is added, executing of `autotest` will state that the new test is invalid (since there are no sample results in the `.ok` directories) and will copy the generated results in the `.diff` directories. After you make sure the results returned from Imagen are correct (by inspecting the files in the `.diff` directories), you can promote the results into the `.ok` directories by using the `-p` command line option (See *Command Line Reference* below). This command simply copies the files found under the `.diff` directories into the `.ok` ones, **overwriting** existing ones! This makes `autotest` think that these are the expected results and future tests will return success only if Imagen continues to return these results.

Note that the entire `.diff` directories are copied, so it's best to be sure that only the right files are stored into these directories. To simplify this task the `-z` command line option is

provided: It deletes everything it finds in the `.diff` directories, making way for their clean generation at next invocations.

If you are unhappy with the content of the `.ok` directories you can destroy them with the `-d` option.

Advise: Do not use these facilities of `autotest` from inside the `autotest` directory because you can easily damage the tests placed there (or at least the sample results in the `.ok` directories). It's better to create new directory for your tests and to operate there only.

Command Line Reference

Please note that `autotest` searches for test data (scripts, configuration files and argument files) into its current directory (and subdirectories if the `-r` command line option is specified). This enables you to enter some subdirectory of the tests directory and thus to perform only some specific subset of the tests given in the `autotest` or another directory.

Usage

```
autotest [options] [-i ImagenDir]
```

Options

<code>-I ImagenDir</code>	Give Imagen location to <code>autotest</code>
<code>-r</code>	Recursive: Work on subdirectories too
<code>-z</code>	Zap differences (see Actions below)
<code>-p</code>	Promote (Note: Destructive! See Actions below)
<code>-d</code>	Destroy results (Note: Destructive! See Actions below)
<code>-h</code>	Print short help screen and exit
<code>-H</code>	Print longer help screen and exit
<code>-v</code>	Print version and exit

Actions

`autotest` performs one of several actions according to command line parameters. If you only need to use `autotest` to ensure Imagen's proper operation, you won't need to change the action from its default (Test).

The other actions can be usable if developing your own tests. The *Creating New Tests* section above describes this into more detail.

Actions reference

◆ Test (the default)

`autotest` performs its tests by executing Imagen for each config/script pair (and optionally arguments list) and checking the results with given ones in a results dir. All encountered differences are displayed and recorded into the differences directory.

◆ Promote

All differences are copied to the result directories by copying the files in the `.diff` directories into the `.ok` ones.

NOTE: This action can destroy some of the result files (by overriding with the new ones). Make sure this is exactly what you want to do!

- ◆ Zap differences

This command removes all previously recorded differences (empties the `.diff` directories)

- ◆ Destroy results

All results (`.ok` directories) are destroyed. You will have to regenerate them in order to make `autotest` work properly again.

Syntax of `.arg` files

The `.arg` files must only contain lines with arguments passed *without* any useless blank lines. An ending“`\n`” is required at the end of the file.

Document TODO

Autotest for Windows