

Skulker Version 2 - Administrator Guide

Version: 1.3
Date: 10th January, 2009.

Version History

| Version | Date | Author | Changes |
|---------|----------------------------------|---------------|---------------------------------|
| 1.0 | 12 th December, 2006. | Simon Edwards | Original. |
| 1.1 | 8 th June, 2007. | Simon Edwards | Improvement documentation. |
| 1.2 | 20 th December, 2007. | Simon Edwards | Support and use HTML reporting. |
| 1.3 | 10 th January, 2009. | Simon Edwards | Use of --var setting. |

Contents

| | | |
|----------|--|-----------|
| 1 | INTRODUCTION..... | 3 |
| 1.1 | WHAT IS SKULKER? | 3 |
| 1.2 | PURPOSE OF ADMINISTRATION GUIDE | 3 |
| 2 | BASIC COMMAND EXECUTION | 4 |
| 2.1 | ENVIRONMENT CONFIGURATION | 4 |
| 2.2 | DISPLAY USAGE OPTIONS | 4 |
| 2.3 | OVERVIEW OF COMMAND ARGUMENT SYNTAX | 6 |
| 3 | SUMMARY OF SUPPORTED ACTIONS..... | 7 |
| 3.1 | AVAILABLE ACTIONS..... | 7 |
| 3.2 | ORDER OF ACTIONS PERFORMED | 8 |
| 4 | SUPPORTED COMMAND LINE ARGUMENTS..... | 9 |
| 4.1 | MANDATORY ARGUMENTS | 9 |
| 4.2 | OPTIONAL ARGUMENTS | 9 |
| 4.3 | COMMONLY USED OPTIONS AND EXAMPLES | 12 |
| 4.3.1 | <i>Generating a HTML Report.....</i> | <i>12</i> |
| 4.3.2 | <i>Listing Files that will be processed.....</i> | <i>12</i> |
| 5 | UNDERSTANDING SKULKER OUTPUT | 13 |
| 5.1 | THE RULE PARSING OUTPUT..... | 13 |
| 5.2 | THE LIST RULES OUTPUT..... | 13 |
| 5.3 | THE SKULK ACTION OUTPUT | 14 |
| 5.4 | THE REPORT ACTION OUTPUT | 15 |
| 5.5 | THE SUMMARY ACTION LOG | 16 |
| 5.6 | THE TOTALS SUMMARY ACTION LOG..... | 16 |
| 5.7 | THE “HTML REPORT” OPTION OUTPUT..... | 17 |
| 5.7.1 | <i>Rule Details</i> | <i>17</i> |
| 5.7.2 | <i>Runs over 31 Days.....</i> | <i>17</i> |

1 Introduction

1.1 What is Skulker?

“Skulker” is a tool that has been designed to automatically manage many files that either grow in size or number over a period of time whilst the Operating System runs and applies a series of actions against these files to ensure the disk space they occupy is automatically managed.

Skulker works by applying a series of rules that define the files to apply a particular rule again, the frequency that a rule can be run and optional logic that must be matched for a particular file is considered a candidate for application by the rule.

Skulker is entirely written in Perl and has been written to work across any number of UNIX variants, including HP-UX, AIX, Solaris and Linux. It supports multiple levels of logging, accurately evaluates (and reports on) space saved, and offers a simply syntax based on XML files.

Skulker also provides a “preview” mode allowing the administrator peace of mind regarding what files will be impacted by the default rules. The environment is extensible - the administrator can change the existing rules or add further rules if they wish.

1.2 Purpose of Administration Guide

This document aims to provide a summary of the command line options that Skulker supports and typical methods of running the utility. It also describes possible automation frequencies and how to interpret the output generated by the various options.

Since Skulker provides statistics reporting it describes not only how to view this information, (and reset it), but also common things to look for which might indicate problems.

This document does not describe the process of installing Skulker - that information can be found in the document “Skulker Version 2 - Installation Guide”. Further this document does not describe how the software environment can be configured - that can be found in the document “Skulker Version 2 - Configuration Guide”.

The power of this utility comes from the handling of rules. Although the default rules ensure that the standard log files for a support UNIX variant are managed additional rules to manage application logs/files for a particular host are recommended. For information on rule configuration the document “Skulker Version 2 - Rule Configuration Guide” should be consulted.

2 Basic Command Execution

2.1 Environment Configuration

Typically Skulker is invoked using the complete command path, for example:

```
spbe903a# /opt/skulker2/bin/skulkerv2
```

Unless a suitable environment is configured the output generated will likely be similar to the following;

```
2007/12/19 18:03:12 LOG      Successfully loaded internal module logging
2007/12/19 18:03:12 LOG      Successfully loaded internal module skulkercfg
2007/12/19 18:03:12 LOG      Successfully loaded internal module match_parse
2007/12/19 18:03:12 LOG      Successfully loaded internal module skulkerrules
2007/12/19 18:03:12 LOG      Successfully loaded internal module skcompression
2007/12/19 18:03:12 LOG      Successfully loaded internal module stats
2007/12/19 18:03:12 LOG      Successfully loaded internal module histstats
2007/12/19 18:03:13 LOG      Successfully loaded internal module histgraphs
2007/12/19 18:03:13 LOG      Successfully loaded internal module skutils
2007/12/19 18:03:13 ERROR    No --cfg argument and SKULKERV2_CFG not defined.
2007/12/19 18:03:13 ERROR    Please set either one to a valid Skulker v2
2007/12/19 18:03:13 ERROR    configuration file.
```

Hence for Skulker to function it must understand the configuration file that is to be used. There are two ways of defining this - via the command line and via the SKULKERV2_CFG environment variable. To use the command line option use;

```
# /opt/skulker2/bin/skulkerv2 \
  --cfg /opt/skulker2/cfg/cfg.xml
```

If you intend to use Skulker on the command line rather than just via schedules it is recommended that the environment variable is set automatically during login. For example the following instead could have been added to a “.profile” file (or equivalent):

```
SKULKERV2_CFG=/opt/skulker2/cfg/cfg.xml
export SKULKERV2_CFG
```

For the rest of the example commands it is assumed that the “SKULKERV2_CFG” environment variable has been set as shown to reduce the size of the command lines.

2.2 Display Usage Options

Once Skulker understands where to find the configuration file it should no longer complain. However unless any meaningful arguments are given on the command line it will not actually do anything!

Due to the number of command line arguments a brief argument summary screen is available. To access this screen, use either of the following commands;

```
# /opt/skulker2/bin/skulkerv2 -h
```

or;

```
# /opt/skulker2/bin/skulkerv2 -?
```

Due to the way Skulker works prior to the actual help message some log output may be shown (this can be ignored). As of the current version, the help screen looks similar to the following;

```
Usage: skulkerv2 [--cfg=F] [--logfile=LF] [--loglevel=N] [rulesfile=F]
      [--rulesdir=D] [--statusfile=F] [--actions=A[,...]]
      [--ignoreintervals] [--testfiles=F] [--historyfile=F]
      [--historykeep] [--historydays D] [--reporttitle T]
      [--reportname X] [--top N] [--days N] [--reportdir X]
      [--var name=value ...]

skulkerv2(1m) is an extensible and flexible temporary and log file management
tool designed for UNIX platforms. It is typically used on a nightly basis to
run a series of rules to manage storage. The rules specify minimum intervals
between processing to ensure certain actions do not occur too often.

The tool also supports statistic gathering and reporting for the last and all
runs of the current rule set, which in practise is especially useful.

All command line arguments are optional - though in most cases at one or two
tend to be given to influence how the tool runs.

--cfg          The path to the configuration file if the default in the
               environment variable SKULKERV2_CFG is not set or the user
               wishes to override it.
--var          Creates a variable and sets the value as specified. This can
               then be used in rules to add further flexibility. This
               can be used any number of times to set variables.
--logfile      The name of the file to log information and errors to.
               If it is defined here it overrides the value in the
               configuration file.
--loglevel     The amount of logging to produce, ranging from 0 ... 5.
               0 is no logging apart from errors and warnings to 4 which
               is maximum verbosity. A log level of 5 produces even more
               information, but makes no actual changes - like a 'dry run'
               function.
--rulesfile    The name of the rules file to process if the user wishes to
               override the value in the configuration file.
               No directory component is necessary.
--rulesdir     The directory where the rules files can be found. If is
               given separately since it is used by rules files to include
               other rules files.
--functionsdir The directory to use to look for functions to load,
               overriding config file value.
--functionslist A comma-separated list of functions that should be loaded -
               others in the directory will be ignored.
--generatorsdir The directory to use to look for generators to load,
               overriding config file value.
--generatorslist A comma-separated list of generators that should be loaded -
               others in the directory will be ignored.
--statusfile   The path to the status file to use. The file generated or
               maintained typically two files depending on the platform.
--actions      The list of actions to perform. This is given as a
               comma-separated list. Supported actions include:
               * parse      Parse the rules files, (pre-requisite for most
                           activities).
               * generate   Generate additional statistic entries for missing
                           rules. If this is not present only rules with
                           existing statistics entries will be handled.
               * list      Simply list the files that will be impacted by
                           the rules to do - do not perform any changes.
               * reset     Reset space statistics.
               * skulk     Actually perform the work in the rules being
                           evaluated.
               * test      Perform the system self test - usually given as
                           the only action if used.
               * report    Report the results of the last run of the rules
                           specified.
```

```

* summary    Produce a summary report of the last run of the
              rules specified.
* tsummary   Produce a summary report of the totals of all
              runs of the rules specified.
* htmlreport Generates a html report [with graphs for the
              rules which save the most disk space].
--ignoreintervals Usually a rule can only be run only after a rule specified
                  period of time has passed. This ignores the intervals for
                  all rules defined to be run.
--testfiles    When running in 'test' mode it allows just specified tests
                  to be performed. Typically only used when a new version
                  is installed. The test files can be found in the 'tests'
                  sub-directory.
--historykeep  A flag to turn on history recording if it is not enabled
                  in the configuration file.
--historyfile  The file where to keep the history - defaults are in the
                  main configuration file.
--historydays The number of days of history to keep - older items will
                  be removed.
--reportname   The prefix to use for the reportname in the given directory.
--reportdir    The directory to create the report and associated graphics.
--reporttitle  The title to put down on the report.
--top          Maximum number of rules to report on defaulting to 10 if
                  not specified.
--days        Number of days to check over - will default to 30 if
                  not specified.

```

Written by Simon Edwards, (C) 2005-2009, proprius.co.uk

2.3 Overview of Command Argument Syntax

Skulker uses the GNU style of argument handling - that is all arguments use “long” names (rather than just single characters) to indicate the option. Each of these long options is prefixed with two dashes, for example;

```
--ignoreintervals
```

If multiple arguments are required they should be white space separated. Since some of the argument names are long another feature that is example is “shortest unique” prefix support. In this case the complete argument is not required just enough characters to uniquely identify the argument in question. For example all of the below are equivalent.

```
-i
--i
--ig
--ignore
--ignoreintervals
```

Note - if shortening the name to a single character (if unique), then only a single dash is actually necessary, though two are still supported.

Some arguments expect a value (such as “--loglevel”) and this can be specified in either of the following ways;

```
--loglevel 3
--loglevel=3
```

Of course “--loglevel” could be shortened to “--logl” if necessary (since there is also a “--logfile” option).

3 Summary of Supported Actions

The Skulker command can work in various ways, and the actual work it performs is determined by the "--action" argument. Although the actions are described in the help text they are shown here in more detail.

3.1 Available Actions

There are a significant number of options, and if more than one is required (which is common), then they should be comma-separated.

- **parse** Simply parse the rules specified and exit with a true status (return code 0) if all rules are valid or false (return code > 0) if any of the rules contain a syntax error.

When running in this manner it is recommended that a verbosity level of at least three is chosen to provide context information for any failures.
- **generate** Skulker must keep information about each rule that is run, and it does this using a "statistics" file. If this file does not contain anything for a particular rule that is scheduled to run Skulker will abort.

When this option is specified it will automatically add entries to the statistics file if deemed necessary.
- **list** List the files that would be processed by each of rules that have been defined. This "list" option does not omit rules that are not scheduled to run as yet, but for each rule summary information does include the frequency, time of last run, and how much space last running the rule saved.

The list of files shown adheres to any pattern specified and any attribute logic that has been defined for the rule.
- **reset** As indicated Skulker keeps statistics about each of the rules that it manages. The statistics cover the time it was last run and the totals of all runs of the rule.

The "reset" actions "zeros" the totals for the "all runs" details. Typically unlikely to be used, but handy following significant application or file system changes.
- **skulk** Actually perform the work specified in the rules file. Each rule has a frequency which means that not all rules configured might run depending on the current time/date and when the rules were last processed.
- **test** Run Skulker in self-test mode. It is strongly recommended that this be used at least once prior to automating Skulker to run on a regular basis.

The self-test is not exhaustive; however it does give a good degree of confidence that Skulker is working as expected on the current platform.
- **report** Produce a detailed report of the last run of the rules specified. The output is shown to the log file - not the screen (unless that is configured)

as the log file!).

The information is quite detailed and is meant to be human readable, though should be straightforward to parse by other tools if necessary.

- **summary** Produce a summary report for the rules specified. As for the “report” option please note that the report is shown to the log device - so it will only appear on the screen if the log device is set appropriately.

This action displays a single line for each rule, showing the number of files processed the last time the rule was run, and also the amount of space saved for that particular run.

- **tsummary** This provides a report in the same format as “summary” - but shows the details of the rules against all rule runs. As stated above this information can be zeroed using the “reset” action if desired.
- **htmlreport** This will generate a HTML-based report including graphics of the amount of disk space saved. This functionality is optional and depends on additional Perl modules not distributed with Skulker - see the installation guide for details.

3.2 Order of Actions Performed

When several actions are specified the order they are given in on the command line is not important - Skulker will always perform them in the order it expects. Thus for example, “summary” is always performed after a “skulk”, if “skulk” is provided as an action.

It should also be noted that “parse” is always performed - since none of the other actions can actually occur until that step has been completed.

The reporting options always occur at the end of the run - to take account of any changes that might just have run.

4 Supported Command Line Arguments

Skulker supports many command line arguments, though many of them are only necessary in certain conditions. It should be noted that Skulker has been designed to run as an automated process rather than interactively and this means that running it on the command line might result in a significant amount of typing to get it to work in the expected manner!

4.1 Mandatory Arguments

Skulker has no mandatory arguments! However unless the following are included it will only parse the configuration and do nothing else.

| Argument | Purpose |
|----------|--|
| --cfg | Actually this is only mandatory if the environment variable SKULKERV2_CFG is not defined. It is used to determine the location of the configuration file. It can be an absolute pathname or relative to the current directory, for example; <code>--cfg=../cfg/cfg.xml</code> |
| --action | Must be specified if you want Skulker to do anything apart from parse the specified configuration. An example might be; <code>--action=generate,list</code> If more than a single action is given each must be separated by a comma - not white space! |

4.2 Optional Arguments

| Argument | Purpose |
|------------|--|
| --logfile | The location where the logs should be written to. If not specified the information from the configuration file is used (which is typically set to a file in "/var/adm" for example. Another possible value is "-" which indicates the log should be written to standard output. An example might be: <code>--logfile="/var/adm/skulker-%D.log"</code> In the above case "%D" will be replaced by the current date in format "YYYYMMDD". The details of the available "%" macros can be found in the "Skulker Rules" document. |
| --loglevel | The amount of logging to perform. If not specified the setting in the configuration file will be used. However the default logging level, "5", does not actually perform the work - just shows the work it would perform. Typically this is set to "3" which gives a reasonable amount of output. Setting it to "0" will produce no output - apart from any warnings or errors. An example usage might be; <code>--loglevel=3</code> |

| Argument | Purpose |
|-------------------|---|
| --rulesfile | <p>The file containing the rules to process. If not specified it will take the details from the configuration file specified. This is the file name in the “rules directory” - and is not expected to contain a path element - see “--rulesdir” entry for that setting.</p> <p>Hence an example of this setting might be;</p> <pre>--rulesfile=test.xml</pre> <p>[The extension must be specified if present, though the file does not need to be suffixed with “.xml”]</p> <p>The default and recommended setting for performing all rules is “MAIN.xml”.</p> |
| --rulesdir | <p>Overrides the directory where rules files can be found compared to the setting in the configuration file. This option is useful when the ruleset in question references other rule sets (via the “include” statement).</p> <p>However in most invocations this does not need to be specified.</p> |
| --rules | <p>A list of actual rules that should be processed. If multiple rules are required then they should be comma-separated. For example:</p> <pre>--rules hpux-tmplogs:1020,1030</pre> |
| --statusfile | <p>The path to the status file (which records details of when each rule was last run and the current and historical statistics). As usual this will override the similar setting in the main configuration file.</p> <p>The value specified is relative to the current working directory, and so relative names can be used if desired, for example;</p> <pre>--statusfile=“../var/status.info”</pre> <p>The file name in question will be created if it does not exist. It will actually create two files - with “.dir” and “.pag” extensions. This extension is omitted when specifying the name on this argument.</p> |
| --ignoreintervals | <p>Usually when a series of rules are to be performed the time the rule was last run will be checked. If less time has elapsed then as specified in the rule frequency or the date/time settings are not met, then the rule is silently ignored.</p> <p>This option will always run the specified rules ignoring the time they were last run.</p> |
| --testfiles | <p>When running in test mode this will indicate which test files should be run. The values specified should be comma separated (if more than one) and be the complete file names with any path component. An example might be;</p> <pre>--testfiles delete.0001.test,tail.0001.test</pre> |
| --historykeep | <p>This will force history to be kept and will override a “keep=0” setting in the main configuration file.</p> |
| --historyfile | <p>The name of the file to store history information in. Again this will override the settings in the main configuration file.</p> |
| --historydays | <p>The maximum age of any history statistics to keep - as indicated this is a number of days. Older entries will be deleted from the history file.</p> |
| --reportname | <p>The name of the report to generate - this should not include a directory component - see the next argument for that. This argument is mandatory if the</p> |

| Argument | Purpose |
|------------------|---|
| | series of actions to run include the “htmlreport” option. If this does not include a “.html” extension one will be added automatically. The action will be aborted if a file with this name already exists. |
| --reportdir | The directory to generate the files in for the “htmlreport” action. Please note several files will be generated here - one for the main html report and one for each graph that is added to the report. All graphs are prefixed with the name of the main report file. |
| --reporttitle | The title to put at the top of the report and in the title of the window when displaying in a web browser. This is optional - a suitable default will be used if not present. |
| --top | The number of rules to show details for in the “htmlreport” action. If not specified it defaults to 10. |
| --days | The number of days over which to gather the history to work out which rules have saved the most space - if not specified it defaults to 30. |
| --var name=value | This can be used to set a variable which can be used in file handling rules ... either setting new values, or overriding the values of variables that are predefined in the used configuration file. This can be used any number of times to set as many variables as necessary. A variable should begin with a letter and consist of letters and numbers. |

Note: The separate use of directory and file options for rules is necessary it helps to determine the location for rules files that are referenced via the support “include” statement.

4.3 Commonly Used Options and Examples

The most common invocation of Skulker is probably the following;

```
/opt/skulker2/bin/skulkerv2 \  
  --cfg /opt/skulker2/cfg/cfg.xml \  
  --loglevel=3 --actions=generate,skulk,summary \  
  --rulesfile=MAIN.xml
```

Typically such a line is scheduled automatically to occur at least once a day; commonly during the early hours of the morning or any other time where the server in question is expected to be least busy.

Skulker can be run at any time and some sites run it every six hours - which rules get run obviously is determined by the frequency given for the rules compared to the time when each particular rule was run.

In the above the logging level is set since the default of “5” would not actually make any changes - essentially it would work as a “preview mode”!

The configuration file is referenced as a full path; as is the name of the program itself, since it is unlikely it will exist in the standard PATH setting for a scheduling tool.

By specifying the actions “generate,skulk,summary” the invocation will automatically add new rules to the statistics database if necessary, actually run the rules and generate a summary report of what changes have been performed.

4.3.1 Generating a HTML Report

A typical command line to generate a html report might be:

```
/opt/skulker2/bin/skulkerv2 \  
  --cfg /opt/skulker2/cfg/cfg.xml --loglevel=3 --actions=htmlreport \  
  --rulesfile=MAIN.xml \  
  --reportdir=/tmp --reportname=skulker_report --reporttitle "Test Report"
```

4.3.2 Listing Files that will be processed

The list option is very useful to check the impact of a change to a particular rule:

```
/opt/skulker2/bin/skulkerv2 \  
  --cfg /opt/skulker2/cfg/cfg.xml --loglevel=3 --actions=list \  
  --rulesfile=MAIN.xml \  
  --rulesfile=MAIN.xml
```

5 Understanding Skulker Output

All samples should here are generated when "--loglevel" set to 3 - which is the recommended setting since it provides plenty of feedback without excessive levels of "noise". In most case the output should be self-explanatory of course.

5.1 The Rule Parsing Output

The rules parsing can generate significant amounts of output since if it needs to default a missing setting it will say so. It will also indicate any changes that it has made to any patterns since it assumes a shell-based pattern has been specified and changes are necessary to convert this to a Perl equivalent.

Some sample output lines might be:

```
2006/05/08 02:20:01 WARN Added $ to end of pattern "/var/adm/cron/log".
2006/05/08 02:20:01 WARN Added ^ to start of file pattern "/var/adm/cron/log$".
```

The above two lines are the most common entries. It will add "^" to the start of patterns and "\$" to the end - this will ensure that only the file named "log" in the specified directory is matched. Without it "slog" and "blog" and "logs" would also be matched if the existed in the directory.

```
2006/05/08 02:20:01 WARN Added . to prefix '*' at start of file pattern
"/tmp/.../*.Z$".
```

A comma shell pattern to match is "*.something" to match any file with a ".something" extension. Unfortunately this is not a valid Perl pattern - and this warning has indicating that it has added a "." prefix to approximate the pattern you might expect to use from the shell.

```
2006/05/08 02:20:01 WARN Rotate "format" defaulted to "%d/%f.%N"
2006/05/08 02:20:01 WARN Rotate "keep" defaulted to "0"
```

Lines similar the above are shown whenever a rule requires certain parameters that have not been provided. Where possible defaults from the current configuration file are used.

5.2 The List Rules Output

When the "list" action is used then following the "parse" log messages each of the rules specified will be shown along with the files that match the rule in question.

In this instance (since nothing is actually being changed) the interval setting is ignored - all rule details are shown. The output generated for each rule is formatted as follows;

```
2007/03/10 00:06:42 LOG Rule linux-cups:10 Rule action : tail handleopen keep=5000
olderto=%d/%f.gz
2007/03/10 00:06:42 LOG Match type : file
2007/03/10 00:06:42 LOG Match pattern : /var/log/cups/^.*_log$
2007/03/10 00:06:42 LOG Match by : ALWAYS
2007/03/10 00:06:42 LOG Last run : 2007/03/02 23:31
2007/03/10 00:06:42 LOG Run interval : 1D
2007/03/10 00:06:42 LOG [Files: 0, Duration:
00:00:00, Space: 0 Kb]
2007/03/10 00:06:42 LOG /var/log/cups/error_log
2007/03/10 00:06:42 LOG Matched 1 file in 1 directory.
```

The “rule action” line shows the type of action to be performed (“delete” in the above example), and any arguments that are to be used (“handleopen” in this case). The “Match type” is typically “file” - i.e. only regular files are processed by the rule other file system objects are ignored.

The “Match pattern” is the pattern that is used to generate matches. This is the pattern after the parser has processed the output and so it might be slightly different than that which was entered as part of the rule definition.

The “Match by” string is an expression that Skulker uses to check the attributes of each file that matches the pattern. The Skulker document “Skulker version 2 - Rule Configuration Guide” describes the available syntax for this option.

The last run is the date (format is “YYYY/MM/DD”) and time when the rule was actually last processed via the “skulk” action. The rule interval is the minimum time that must elapse between two “skulk” actions against this rule being performed.

Finally the last line of the header is a terse summary of the characteristics of the last run of the rule, for example;

```
[Files: 4, Duration: 00:00:00, Space: 422.2 Kb]
```

The above indicates that on the last run the rule processed 4 files in less than a second (time is HH:MM:SS format) and is reclaimed 422 Kb.

The output generated for each rule also includes the filenames of all files matched - and so can be very lengthy if a rule matches many files.

5.3 The Skulk Action Output

The “Skulk” output shows summary details of each rule that is processed (ignoring those for which the elapsed time between invocations has not been met). The summary output is the same as the “list” action shown above.

After that, assuming that loglevel 3 is in use each type of rule will produce some output regarding any file that is processed. Examples of the more common rule types are shown below.

5.3.1.1 Command output

```
2006/05/09 02:20:11 LOG      Running commands : [1]
/opt/mbnatools/skulker2/utils/startcron
2006/05/09 02:20:14 LOG      Command: /opt/mbnatools/skulker2/utils/startcron RC=0
```

For each command the command line is shown along with the return code of each command.

5.3.1.2 Rotate output

```
2006/05/09 02:20:06 LOG      Rotated [copy/truncated] file
/opt/hpws/apache/logs/error_log successfully.
2006/05/09 02:20:06 LOG      Rotated [copy/truncated] file
/opt/hpws/apache/logs/access_log successfully.
```

For rotate the names of the files that have been rotated are given. Optionally the lines may contain the text [“copy/truncated”] which is a method Skulker uses when the rule indicates that it should work with the possibility of the files being open for writing.

5.3.1.3 Compress output

```
2006/05/09 02:20:06 LOG      Compress: /tmp/sh/.msanders.1092 -> /tmp/sh/.msanders.1092.Z
[ 56.02%].
```

When compressing files it shows details for each file - the original name and the newly compressed name. The figure in brackets the relative size of the file compared to the original - so 56% means the compressed file is just over half the size of the uncompressed one.

5.3.1.4 Delete output

```
2006/05/09 02:20:07 LOG      Delete: Truncated - /var/adm/btmp [user:group]
2006/05/09 02:20:07 LOG      Delete: Removed - /tmp/xxwewe [user:group]
```

The messages shown for delete will include “Truncated” if the file has not actually been removed and just reduced to 0 bytes. This occurs if the “truncate” or “handleopen” flags have been specified.

5.3.1.5 Tail output

```
2006/05/09 12:55:24 LOG      /tmp/881/files/xx2 - 15 lines removed.
```

When tail removes data from a file a line similar to the above is shown - indicating the number of lines that were removed. If no lines need to be removed that will also be indicated.

5.3.1.6 Organise output

```
2006/05/09 12:55:17 LOG      Moved file '/tmp/files/xx' to '/tmp/files/2006-04-07'.
```

Organise is a facility that moves files to specified directories based on the characteristics of each file. When a file is moved output similar to the above is shown - which indicates the name of the directory the particular file was moved to.

5.4 The Report Action Output

The “report” action gives a long listing for each of the rules specified showing all statistics available for each rule. The format of the output generated is;

```
2006/05/09 14:18:10 LOG      Rule hostonly-stbe905a:1100  Rule action          : compress
handleopen type=compress
2006/05/09 14:18:10 LOG                Match type           : file
2006/05/09 14:18:10 LOG                Match pattern        :
/a/b/.../*.*(log|spdslog)$
2006/05/09 14:18:10 LOG                Match by             : MTIME OLDER THAN 1D
2006/05/09 14:18:10 LOG                Last run              : 2006/05/09 12:54
2006/05/09 14:18:10 LOG                Run interval         : 1D
2006/05/09 14:18:10 LOG                Last run Info        : [Files: 0, Duration:
00:00:00, Space: 0 Kb]
2006/05/09 14:18:10 LOG                Total run count      : 37
2006/05/09 14:18:10 LOG                Total files processed : 1138
2006/05/09 14:18:10 LOG                Total time spent     : 00:04:42
2006/05/09 14:18:10 LOG                Total space saved    : 16464.3
```

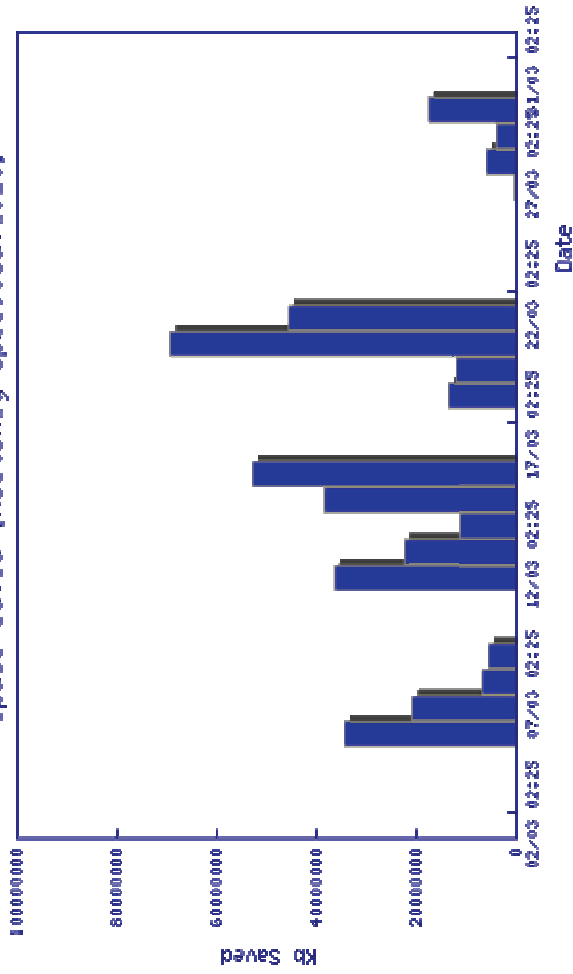
Much of the output is shown as part of the “list” and “skulk” actions - the additional “Total” settings are the details for all runs of this particular rule - including how many times it has been run, how many files it has processed, the time spent processing those files, and the total disk space saved (in Kb).

5.7 The "Html Report" Option Output

The html report option allows the user to generate a graphical output showing the largest space saving rules over a period of time. This is only possible if historical information is being kept. A typical output will have entries similar to the following:

Top 10 Skulker Rules for December 2007

Space saved [hostonly-spbbe903a:1020]



5.7.1 Rule Details

Rule action : saswork handleopen
Match type : directory
Match pattern : /sas/work/saswork/.../ ^SAS_work.*stbe905a\$
Match by : ALWAYS
Last run : 2007/12/20 02:25
Run interval : 6H
[Files: 49, Duration: 00:00:00, Space: 3221380.5 Kb]

5.7.2 Runs over 31 Days

Runs : 4
Files: 84
Space: 4,115,729 Kb
Spent: 00:00:00