NAME

Mail::SpamAssassin::Conf − SpamAssassin configuration file

SYNOPSIS

```
# a comment

rewrite_subject                 1

full PARA_A_2_C_OF_1618         /Paragraph .a.{0,10}2.{0,10}C. of S. 1618/i
describe PARA_A_2_C_OF_1618     Claims compliance with senate bill 1618

header FROM_HAS_MIXED_NUMS      From =~ /\d+[a-z]+\d+\S*@/i
describe FROM_HAS_MIXED_NUMS    From: contains numbers mixed in with letters

score A_HREF_TO_REMOVE          2.0

lang es describe FROM_FORGED_HOTMAIL Forzado From: simula ser de hotmail.com
```

DESCRIPTION

SpamAssassin is configured using some traditional UNIX-style configuration files, loaded from the /usr/share/spamassassin and /etc/mail/spamassassin directories.

The # character starts a comment, which continues until end of line.

Whitespace in the files is not significant, but please note that starting a line with whitespace is deprecated, as we reserve its use for multi-line rule definitions, at some point in the future.

Paths can use ~ to refer to the user's home directory.

Where appropriate, default values are listed in parentheses.

USER PREFERENCES

require_version n.nn

Indicates that the entire file, from this line on, requires a certain version of SpamAssassin to run. If an older or newer version of SpamAssassin tries to read configuration from this file, it will output a warning instead, and ignore it.

version_tag string

This tag is appended to the SA version in the X-Spam-Status header. You should include it when modify your ruleset, especially if you plan to distribute it. A good choice for *string* is your last name or your initials followed by a number which you increase with each change.

e.g.

```
version_tag myrules1    # version=2.41-myrules1
```

whitelist_from add@ress.com

Used to specify addresses which send mail that is often tagged (incorrectly) as spam; it also helps if they are addresses of big companies with lots of lawyers. This way, if spammers impersonate them, they'll get into big trouble, so it doesn't provide a shortcut around SpamAssassin.

Whitelist and blacklist addresses are now file-glob-style patterns, so `friend@somewhere.com`, `*@isp.com`, or `*.domain.net` will all work. Specifically, `*` and `?` are allowed, but all other metacharacters are not. Regular expressions are not used for security reasons.

Multiple addresses per line, separated by spaces, is OK. Multiple `whitelist_from` lines is also OK.

The headers checked for whitelist addresses are as follows: if `Resent-From` is set, use that;

otherwise check all addresses taken from the following set of headers:

```
Envelope-Sender
Resent-Sender
X-Envelope-From
From
```

e.g.

```
whitelist_from joe@example.com fred@example.com
whitelist_from *@example.com
```

unwhitelist_from add@ress.com

Used to override a default whitelist_from entry, so for example a distribution whitelist_from can be overriden in a local.cf file, or an individual user can override a whitelist_from entry in their own `user_prefs` file. The specified email address has to match exactly the address previously used in a whitelist_from line.

e.g.

```
unwhitelist_from joe@example.com fred@example.com
unwhitelist_from *@example.com
```

whitelist_from_rcvd addr@lists.sourceforge.net sourceforge.net

Use this to supplement the whitelist_from addresses with a check against the Received headers. The first parameter is the address to whitelist, and the second is a domain to match in the Received headers. This domain does not allow globbing, and must be followed by a numeric IP address in brackets in the Received headers.

e.g.

```
whitelist_from_rcvd joe@example.com  example.com
whitelist_from_rcvd *@axkit.org      sergeant.org
```

unwhitelist_from_rcvd add@ress.com

Used to override a default whitelist_from_rcvd entry, so for example a distribution whitelist_from_rcvd can be overriden in a local.cf file, or an individual user can override a whitelist_from_rcvd entry in their own `user_prefs` file. The specified email address has to match exactly the address previously used in a whitelist_from_rcvd line.

e.g.

```
unwhitelist_from_rcvd joe@example.com fred@example.com
unwhitelist_from_rcvd *@axkit.org
```

blacklist_from add@ress.com

Used to specify addresses which send mail that is often tagged (incorrectly) as non-spam, but which the user doesn't want. Same format as `whitelist_from`.

unblacklist_from add@ress.com

Used to override a default blacklist_from entry, so for example a distribution blacklist_from can be overriden in a local.cf file, or an individual user can override a blacklist_from entry in their own `user_prefs` file.

e.g.

```
unblacklist_from joe@example.com fred@example.com
unblacklist_from *@spammer.com
```

whitelist_to add@ress.com
: If the given address appears in the `To:` or `Cc:` headers, mail will be whitelisted. Useful if you're deploying SpamAssassin system-wide, and don't want some users to have their mail filtered. Same format as `whitelist_from`.

  There are three levels of To-whitelisting, `whitelist_to`, `more_spam_to` and `all_spam_to`. Users in the first level may still get some spammish mails blocked, but users in `all_spam_to` should never get mail blocked.

more_spam_to add@ress.com
: See above.

all_spam_to add@ress.com
: See above.

required_hits n.nn (default: 5)
: Set the number of hits required before a mail is considered spam. `n.nn` can be an integer or a real number. 5.0 is the default setting, and is quite aggressive; it would be suitable for a single-user setup, but if you're an ISP installing SpamAssassin, you should probably set the default to be more conservative, like 8.0 or 10.0. It is not recommended to automatically delete or discard messages marked as spam, as your users **will** complain, but if you choose to do so, only delete messages with an exceptionally high score such as 15.0 or higher.

score SYMBOLIC_TEST_NAME n.nn [n.nn n.nn n.nn]
: Assign scores (the number of points for a hit) to a given test. Scores can be positive or negative real numbers or integers. `SYMBOLIC_TEST_NAME` is the symbolic name used by SpamAssassin for that test; for example, 'FROM_ENDS_IN_NUMS'.

  If only one valid score is listed, then that score is always used for a test.

  If four valid scores are listed, then the score that is used depends on how SpamAssassin is being used. The first score is used when both Bayes and network tests are disabled. The second score is used when Bayes is disabled, but network tests are enabled. The third score is used when Bayes is enabled and network tests are disabled. The fourth score is used when Bayes is enabled and network tests are enabled.

  Note that test names which begin with '`__`' are reserved for meta-match sub-rules, and are not scored or listed in the 'tests hit' reports.

  If no score is given for a test, the default score is 1.0, or 0.01 for tests whose names begin with 'T_' (this is used to indicate a rule under test).

rewrite_subject { 0 | 1 } (default: 0)
: By default, the subject lines of suspected spam will not be tagged. This can be enabled here.

fold_headers { 0 | 1 } (default: 1)
: By default, the X-Spam-Status header will be whitespace folded, in other words, it will be broken up into multiple lines instead of one very long one. This can be disabled here.

always_add_headers { 0 | 1 } (default: 1)
: By default, **X-Spam-Status**, **X-Spam-Checker-Version**, (and optionally **X-Spam-Level**) will be added to all messages scanned by SpamAssassin. If you don't want to add the headers to non-spam, set this value to 0. See also **always_add_report**.

always_add_report { 0 | 1 } (default: 0)
: By default, mail tagged as spam includes a report, either in the headers or in an attachment (report_safe). If you set this to option to `1`, the report will be included in the **X-Spam-Report**

header, even if the message is not tagged as spam. Note that the report text **always** states that the mail is spam, since normally the report is only added if the mail **is** spam.

This can be useful if you want to know what rules the mail triggered, and why it was not tagged as spam. See also **always_add_headers**.

spam_level_stars { 0 | 1 } (default: 1)
: By default, a header field called ''X-Spam-Level'' will be added to the message, with its value set to a number of asterisks equal to the score of the message. In other words, for a message scoring 7.2 points:

  X-Spam-Level: *******

  This can be useful for MUA rule creation.

spam_level_char { x (some character, unquoted) } (default: *)
: By default, the ''X-Spam-Level'' header will use a '*' character with its length equal to the score of the message. Some people don't like escaping *s though, so you can set the character to anything with this option.

  In other words, for a message scoring 7.2 points with this option set to .

  X-Spam-Level: ....

subject_tag STRING ... (default: *****SPAM*****)
: Text added to the `Subject:` line of mails that are considered spam, if `rewrite_subject` is 1. _HITS_ in the tag will be replace with the calculated score for this message. _REQD_ will be replaced with the threshold.

report_safe { 0 | 1 | 2 } (default: 1)
: if this option is set to 1, if an incoming message is tagged as spam, instead of modifying the original message, SpamAssassin will create a new report message and attach the original message as a message/rfc822 MIME part (ensuring the original message is completely preserved, not easily opened, and easier to recover).

  If this option is set to 2, then original messages will be attached with a content type of text/plain instead of message/rfc822. This setting may be required for safety reasons on certain broken mail clients that automatically load attachments without any action by the user. This setting may also make it somewhat more difficult to extract or view the original message.

  If this option is set to 0, incoming spam is only modified by adding some headers and no changes will be made to the body.

use_terse_report { 0 | 1 } (default: 0)
: By default, SpamAssassin uses a long report format, explaining what happened to the mail message, for newbie users. If you would prefer shorter reports, set this to `1`.

skip_rbl_checks { 0 | 1 } (default: 0)
: By default, SpamAssassin will run RBL checks. If your ISP already does this for you, set this to 1.

ok_languages xx [yy zz ...] (default: all)
: Which languages are considered OK to receive mail in. SpamAssassin will try to detect the language used in the message text.

  Note that the language cannot always be recognized reliably. In that case, no points will be assigned.

  The rule `UNDESIRED_LANGUAGE_BODY` is triggered based on how this is set.

  The following languages are recognized. In your configuration, you must use the language specifier located in the first column, not the English name for the language. You may also specify `all` if your language is not listed, or if you want to allow any language. The default setting is `all`.

af	afrikaans
am	amharic
ar	arabic
be	byelorussian
bg	bulgarian
bs	bosnian
ca	catalan
cs	czech
cy	welsh
da	danish
de	german
el	greek
en	english
eo	esperanto
es	spanish
et	estonian
eu	basque
fa	persian
fi	finnish
fr	french
fy	frisian
ga	irish gaelic
gd	scottish gaelic
he	hebrew
hi	hindi
hr	croatian
hu	hungarian
hy	armenian
id	indonesian
is	icelandic
it	italian
ja	japanese
ka	georgian
ko	korean
la	latin
lt	lithuanian
lv	latvian
mr	marathi
ms	malay
ne	nepali
nl	dutch
no	norwegian
pl	polish
pt	portuguese
qu	quechua
rm	rhaeto-romance
ro	romanian
ru	russian
sa	sanskrit
sco	scots
sk	slovak

sl	slovenian
sq	albanian
sr	serbian
sv	swedish
sw	swahili
ta	tamil
th	thai
tl	tagalog
tr	turkish
uk	ukrainian
vi	vietnamese
yi	yiddish
zh	chinese

examples:

```
ok_languages all         (allow all languages)
ok_languages en          (only allow English)
ok_languages en ja zh    (allow English, Japanese, and Chinese)
```

Note: if there are multiple ok_languages lines, only the last one is used.

ok_locales xx [yy zz ...] (default: all)

Which locales (country codes) are considered OK to receive mail from. Mail using **character sets** used by languages in these countries, will not be marked as possibly being spam in a foreign language.

If you receive lots of spam in foreign languages, and never get any non-spam in these languages, this may help. Note that all ISO-8859−* character sets, and Windows code page character sets, are always permitted by default.

Set this to `all` to allow all character sets. This is the default.

The rules `CHARSET_FARAWAY`, `CHARSET_FARAWAY_BODY`, and `CHARSET_FARAWAY_HEADERS` are triggered based on how this is set.

Select the locales to allow from the list below:

en Western character sets in general

ja Japanese

ko Korea

ru Cyrillic charsets

th Thai

zh Chinese (both simplified and traditional)

examples:

```
ok_locales all         (allow all locales)
ok_locales en          (only allow English)
ok_locales en ja zh    (allow English, Japanese, and Chinese)
```

Note: if there are multiple ok_locales lines, only the last one is used.

describe SYMBOLIC_TEST_NAME description ...

Used to describe a test. This text is shown to users in the detailed report.

Note that test names which begin with '__' are reserved for meta-match sub-rules, and are not scored or listed in the 'tests hit' reports.

tflags SYMBOLIC_TEST_NAME [{ net | nice | learn | userconf } ...]
Used to set flags on a test. These flags are used in the score-determination back end system for details of the test's behaviour. The following flags can be set:

net The test is a network test, and will not be run in the mass checking system or if **−L** is used, therefore its score should not be modified.

nice The test is intended to compensate for common false positives, and should be assigned a negative score.

userconf
The test requires user configuration before it can be used (like language- specific tests).

learn
The test requires training before it can be used.

report ...some text for a report...
Set the report template which is attached to spam mail messages. See the `10_misc.cf` configuration file in `/usr/share/spamassassin` for an example.

If you change this, try to keep it under 76 columns (inside the the dots below). Bear in mind that EVERY line will be prefixed with ''SPAM: '' in order to make it clear what's been added, and allow other filters to **remove** spamfilter modifications, so you lose 6 columns right there. Also note that the first line of the report must start with 4 dashes, for the same reason. Each `report` line appends to the existing template, so use `clear_report_template` to restart.

The following template items are supported, and will be filled out by SpamAssassin:

HITS: the number of hits the message triggered
REQD: the required hits to be considered spam
SUMMARY: the full details of what hits were triggered
VER: SpamAssassin version
HOME: SpamAssassin home URL

clear_report_template
Clear the report template.

unsafe_report ...some text for a report...
Set the report template which is attached to spam mail messages which contain a non-text/plain part. See the `10_misc.cf` configuration file in `/usr/share/spamassassin` for an example.

Each `unsafe−report` line appends to the existing template, so use `clear_unsafe_report_template` to restart.

clear_unsafe_report_template
Clear the unsafe_report template.

terse_report ...some text for a report...
Set the report template which is attached to spam mail messages, for the terse-report format. See the `10_misc.cf` configuration file in `/usr/share/spamassassin` for an example.

clear_terse_report_template
Clear the terse-report template.

spamtrap ...some text for spamtrap reply mail...
A template for spam-trap responses. If the first few lines begin with `Xxxxxx: yyy` where Xxxxxx is a header and yyy is some text, they'll be used as headers. See the `10_misc.cf` configuration file in `/usr/share/spamassassin` for an example.

clear_spamtrap_template
Clear the spamtrap template.

use_dcc (0 | 1) (default 1)
: Whether to use DCC, if it is available.

dcc_timeout n (default: 10)
: How many seconds you wait for dcc to complete before you go on without the results

dcc_body_max NUMBER
dcc_fuz1_max NUMBER
dcc_fuz2_max NUMBER
: DCC (Distributed Checksum Clearinghouse) is a system similar to Razor. This option sets how often a message's body/fuz1/fuz2 checksum must have been reported to the DCC server before SpamAssassin will consider the DCC check as matched.

  As nearly all DCC clients are auto-reporting these checksums you should set this to a relatively high value, e.g. 999999 (this is DCC's MANY count).

  The default is 999999 for all these options.

dcc_add_header { 0 | 1 } (default: 0)
: DCC processing creates a message header containing the statistics for the message. This option sets whether SpamAssassin will add the heading to messages it processes.

  The default is to not add the header.

use_pyzor (0 | 1) (default 1)
: Whether to use Pyzor, if it is available.

pyzor_timeout n (default: 10)
: How many seconds you wait for Pyzor to complete before you go on without the results.

pyzor_max NUMBER
: Pyzor is a system similar to Razor. This option sets how often a message's body checksum must have been reported to the Pyzor server before SpamAssassin will consider the Pyzor check as matched.

  The default is 5.

pyzor_add_header { 0 | 1 } (default: 0)
: Pyzor processing creates a message header containing the statistics for the message. This option sets whether SpamAssassin will add the heading to messages it processes.

  The default is to not add the header.

pyzor_options options
: Specify options to the pyzor command. Please note that only [A-Za-z0−9 −/] is allowed (security).

num_check_received { integer } (default: 2)
: How many received lines from and including the original mail relay do we check in RBLs (you'd want at least 1 or 2). Note that for checking against dialup lists, you can call check_rbl with a special set name of ''set-firsthop'' and this rule will only be matched against the first hop if there is more than one hop, so that you can set a negative score to not penalize people who properly relayed through their ISP. See dialup_codes for more details and an example

use_razor1 (0 | 1) (default 1)
: Whether to use Razor version 1, if it is available.

use_razor2 (0 | 1) (default 1)
: Whether to use Razor version 2, if it is available.

razor_timeout n (default 10)
: How many seconds you wait for razor to complete before you go on without the results

use_bayes (0 | 1) (default 1)
: Whether to use the naive-Bayesian-style classifier built into SpamAssassin.

rbl_timeout n (default 30)
: All RBL queries are started at the beginning and we try to read the results at the end. In case some of them are hanging or not returning, you can specify here how long you're willing to wait for them before deciding that they timed out

check_mx_attempts n (default: 2)
: By default, SpamAssassin checks the From: address for a valid MX this many times, waiting 5 seconds each time.

check_mx_delay n (default 5)
: How many seconds to wait before retrying an MX check.

dns_available { yes | test[: name1 name2...] | no } (default: test)
: By default, SpamAssassin will query some default hosts on the internet to attempt to check if DNS is working on not. The problem is that it can introduce some delay if your network connection is down, and in some cases it can wrongly guess that DNS is unavailable because the test connections failed. SpamAssassin includes a default set of 13 servers, among which 3 are picked randomly.

You can however specify your own list by specifying

dns_available test: server1.tld server2.tld server3.tld

Please note, the DNS test queries for MX records so if you specify your own list of servers, please make sure to choose the *one* (s) which has an associated MX record.

auto_whitelist_factor n (default: 0.5, range [0..1])
: How much towards the long-term mean for the sender to regress a message. Basically, the algorithm is to track the long-term mean score of messages for the sender (`mean`), and then once we have otherwise fully calculated the score for this message (`score`), we calculate the final score for the message as:

`finalscore` = `score` + (`mean` − `score`) ∗ `factor`

So if `factor` = 0.5, then we'll move to half way between the calculated score and the mean. If `factor` = 0.3, then we'll move about 1/3 of the way from the score toward the mean. `factor` = 1 means just use the long-term mean; `factor` = 0 mean just use the calculated score.

auto_learn (0 | 1) (default: 1)
: Whether SpamAssassin should automatically feed high-scoring mails (or low-scoring mails, for non-spam) into its learning systems. The only learning system supported currently is a naive-Bayesian-style classifier.

Note that certain tests are ignored when determining whether a message should be trained upon:
− auto-whitelist (AWL)
− rules with tflags set to 'learn' (the Bayesian rules)
− rules with tflags set to 'userconf' (user white/black-listing rules, etc)

Also note that auto-training occurs using scores from either scoreset 0 or 1, depending on what scoreset is used during message check. It is likely that the message check and auto-train scores will be different.

auto_learn_threshold_nonspam n.nn (default −2.0)
: The score threshold below which a mail has to score, to be fed into SpamAssassin's learning systems automatically as a non-spam message.

auto_learn_threshold_spam n.nn (default 15.0)
: The score threshold above which a mail has to score, to be fed into SpamAssassin's learning systems automatically as a spam message.

bayes_ignore_header

If you receive mail filtered by upstream mail systems, like a spam-filtering ISP or mailing list, and that service adds new headers (as most of them do), these headers may provide inappropriate cues to the Bayesian classifier, allowing it to take a ''short cut''. To avoid this, list the headers using this setting. Example:

```
bayes_ignore_header X-Upstream-Spamfilter
bayes_ignore_header X-Upstream-SomethingElse
```

SETTINGS

These settings differ from the ones above, in that they are considered 'privileged'. Only users running `spamassassin` from their procmailrc's or forward files, or sysadmins editing a file in `/etc/mail/spamassassin`, can use them. `spamd` users cannot use them in their `user_prefs` files, for security and efficiency reasons, unless allow_user_rules is enabled (and then, they may only add rules from below).

allow_user_rules { 0 | 1 } (default: 0)

This setting allows users to create rules (and only rules) in their `user_prefs` files for use with `spamd`. It defaults to off, because this could be a severe security hole. It may be possible for users to gain root level access if `spamd` is run as root. It is NOT a good idea, unless you have some other way of ensuring that users' tests are safe. Don't use this unless you are certain you know what you are doing. Furthermore, this option causes spamassassin to recompile all the tests each time it processes a message for a user with a rule in his/her `user_prefs` file, which could have a significant effect on server load. It is not recommended.

header SYMBOLIC_TEST_NAME header op /pattern/modifiers [if-unset: STRING]

Define a test. `SYMBOLIC_TEST_NAME` is a symbolic test name, such as 'FROM_ENDS_IN_NUMS'. `header` is the name of a mail header, such as 'Subject', 'To', etc.

'ALL' can be used to mean the text of all the message's headers. 'ToCc' can be used to mean the contents of both the 'To' and 'Cc' headers.

'MESSAGEID' is a symbol meaning all Message-Id's found in the message; some mailing list software moves the *real* Message-Id to 'Resent-Message-Id' or 'X-Message-Id', then uses its own one in the 'Message-Id' header. The value returned for this symbol is the text from all 3 headers, separated by newlines.

`op` is either `=~` (contains regular expression) or `!~` (does not contain regular expression), and `pattern` is a valid Perl regular expression, with `modifiers` as regexp modifiers in the usual style.

If the `[if-unset: STRING]` tag is present, then `STRING` will be used if the header is not found in the mail message.

Test names should not start with a number, and must contain only alphanumerics and underscores. It is suggested that lower-case characters not be used, as an informal convention. Dashes are not allowed.

Note that test names which begin with '__' are reserved for meta-match sub-rules, and are not scored or listed in the 'tests hit' reports. Test names which begin with 'T_' are reserved for tests which are undergoing QA, and these are given a very low score.

If you add or modify a test, please be sure to run a sanity check afterwards by running `spamassassin --lint`. This will avoid confusing error messages, or other tests being skipped as a side-effect.

header SYMBOLIC_TEST_NAME exists:name_of_header
: Define a header existence test. `name_of_header` is the name of a header to test for existence. This is just a very simple version of the above header tests.

header SYMBOLIC_TEST_NAME eval:name_of_eval_method([arguments])
: Define a header eval test. `name_of_eval_method` is the name of a method on the `Mail::SpamAssassin::EvalTests` object. `arguments` are optional arguments to the function call.

body SYMBOLIC_TEST_NAME /pattern/modifiers
: Define a body pattern test. `pattern` is a Perl regular expression.

  The 'body' in this case is the textual parts of the message body; any non-text MIME parts are stripped, and the message decoded from Quoted-Printable or Base-64−encoded format if necessary. All HTML tags and line breaks will be removed before matching.

body SYMBOLIC_TEST_NAME eval:name_of_eval_method([args])
: Define a body eval test. See above.

uri SYMBOLIC_TEST_NAME /pattern/modifiers
: Define a uri pattern test. `pattern` is a Perl regular expression.

  The 'uri' in this case is a list of all the URIs in the body of the email, and the test will be run on each and every one of those URIs, adjusting the score if a match is found. Use this test instead of one of the body tests when you need to match a URI, as it is more accurately bound to the start/end points of the URI, and will also be faster.

rawbody SYMBOLIC_TEST_NAME /pattern/modifiers
: Define a raw-body pattern test. `pattern` is a Perl regular expression.

  The 'raw body' of a message is the text, including all textual parts. The text will be decoded from base64 or quoted-printable encoding, but HTML tags and line breaks will still be present.

rawbody SYMBOLIC_TEST_NAME eval:name_of_eval_method([args])
: Define a raw-body eval test. See above.

full SYMBOLIC_TEST_NAME /pattern/modifiers
: Define a full-body pattern test. `pattern` is a Perl regular expression.

  The 'full body' of a message is the un-decoded text, including all parts (including images or other attachments). SpamAssassin no longer tests full tests against decoded text; use `rawbody` for that.

full SYMBOLIC_TEST_NAME eval:name_of_eval_method([args])
: Define a full-body eval test. See above.

meta SYMBOLIC_TEST_NAME boolean expression
: Define a boolean expression test in terms of other tests that have been hit or not hit. For example:

  meta META1 TEST1 && !(TEST2 || TEST3)

  Note that English language operators (''and'', ''or'') will be treated as rule names, and that there is no `XOR` operator.

meta SYMBOLIC_TEST_NAME boolean arithmetic expression
: Can also define a boolean arithmetic expression in terms of other tests, with a hit test having the value ''1'' and an unhit test having the value ''0''. For example:

  meta META2 (3 * TEST1 − 2 * TEST2) > 0

  Note that Perl builtins and functions, like `abs()`, **can't** be used, and will be treated as rule names.

  If you want to define a meta-rule, but do not want its individual sub-rules to count towards the final score unless the entire meta-rule matches, give the sub-rules names that start with '`__`' (two underscores). SpamAssassin will ignore these for scoring.

PRIVILEGED SETTINGS

These settings differ from the ones above, in that they are considered 'more privileged' — even more than the ones in the SETTINGS section. No matter what `allow_user_rules` is set to, these can never be set from a user's `user_prefs` file.

test SYMBOLIC_TEST_NAME (ok|fail) Some string to test against
: Define a regression testing string. You can have more than one regression test string per symbolic test name. Simply specify a string that you wish the test to match.

  These tests are only run as part of the test suite − they should not affect the general running of SpamAssassin.

razor_config filename
: Define the filename used to store Razor's configuration settings. Currently this is left to Razor to decide.

pyzor_path STRING
: This option tells SpamAssassin specifically where to find the `pyzor` client instead of relying on SpamAssassin to find it in the current PATH. Note that if *taint mode* is enabled in the Perl interpreter, you should use this, as the current PATH will have been cleared.

dcc_path STRING
: This option tells SpamAssassin specifically where to find the `dccproc` client instead of relying on SpamAssassin to find it in the current PATH. Note that if *taint mode* is enabled in the Perl interpreter, you should use this, as the current PATH will have been cleared.

dcc_options options
: Specify additional options to the *dccproc*(8) command. Please note that only [A-Z −] is allowed (security).

  The default is `−R`

auto_whitelist_path /path/to/file (default: ~/.spamassassin/auto-whitelist)
: Automatic-whitelist directory or file. By default, each user has their own, in their `~/.spamassassin` directory with mode 0700, but for system-wide SpamAssassin use, you may want to share this across all users.

bayes_path /path/to/file (default: ~/.spamassassin/bayes)
: Path for Bayesian probabilities databases. Several databases will be created, with this as the base, with `_toks`, `_seen` etc. appended to this filename; so the default setting results in files called `~/.spamassassin/bayes_seen`, `~/.spamassassin/bayes_toks` etc.

  By default, each user has their own, in their `~/.spamassassin` directory with mode 0700/0600, but for system-wide SpamAssassin use, you may want to reduce disk space usage by sharing this across all users. (However it should be noted that Bayesian filtering appears to be more effective with an individual database per user.)

timelog_path /path/to/dir (default: NULL)
: If you set this value, SpamAssassin will try to create logfiles for each message it processes and dump information on how fast it ran, and in which parts of the code the time was spent. The files will be named: `unixdate_messageid` (i.e 1023257504_chuvn31gdu@4ax.com)

  Make sure SA can write the log file; if you're not sure what permissions are needed, chmod the log directory to 1777, and adjust later.

auto_whitelist_file_mode (default: 0700)
: The file mode bits used for the automatic-whitelist directory or file.

  Make sure you specify this using the 'x' mode bits set, as it may also be used to create directories. However, if a file is created, the resulting file will not have any execute bits set (the umask is set to

111).

bayes_file_mode (default: 0700)

The file mode bits used for the Bayesian filtering database files.

Make sure you specify this using the 'x' mode bits set, as it may also be used to create directories. However, if a file is created, the resulting file will not have any execute bits set (the umask is set to 111).

bayes_use_hapaxes (default: 1)

Should the Bayesian classifier use hapaxes (words/tokens that occur only once) when classifying? This produces significantly better hit-rates, but increases database size by about a factor of 8 to 10.

bayes_use_chi2_combining (default: 0)

Should the Bayesian classifier use chi-squared combining, instead of Robinson/Graham-style naive Bayesian combining? Chi-squared produces more 'extreme' output results, but may be more resistant to changes in corpus size etc.

bayes_expiry_min_db_size (default: 100000)

What should be the minimum size of the Bayes tokens database? The database will never be shrunk below this many entries. 100000 entries is roughly equivalent to a 5Mb database file.

bayes_expiry_scan_count (default: 5000)

When expiring old entries from the Bayes databases, tokens which have not been read in this many messages will be removed (unless to do so would shrink the database below the bayes_expiry_min_db_size size).

user_scores_dsn DBI:databasetype:databasename:hostname:port

If you load user scores from an SQL database, this will set the DSN used to connect. Example: DBI:mysql:spamassassin:localhost

user_scores_sql_username username

The authorized username to connect to the above DSN.

user_scores_sql_password password

The password for the database username, for the above DSN.

user_scores_sql_table tablename

The table user preferences are stored in, for the above DSN.

user_scores_sql_field_username field_username

The field that the username whose preferences you're looking up is stored in. Default: username.

user_scores_sql_field_preference field_preference

The name of the preference that you're looking for. Default: preference.

user_scores_sql_field_value field_value

The name of the value you're looking for. Default: value.

user_scores_sql_field_scope field_scope

The 'scope' field. In Horde this makes the preference a single-module preference or a global preference. There's no real need to change it in other systems. Default: spamassassin.
Default:

{ ''dialups.mail-abuse.org.'' => ''127.0.0.3'',
For DUL + other codes, we ignore that it's on DUL
''rbl-plus.mail-abuse.org.'' => ''127.0.0.2'',
''relays.osirusoft.com.'' => ''127.0.0.3'' };

WARNING!!! When passing a reference to a hash, you need to put the whole hash in one line for the parser to read it correctly (you can check with spamassassin –D < mesg).

Set this to what your RBLs return for dialup IPs It is used by dialup-firsthop and relay-firsthop rules so that you can match DUL codes and compensate DUL checks with a negative score if the IP is a dialup IP the mail originated from and it was properly relayed by a hop before reaching you (hopefully not your secondary MX :−) The trailing ''−firsthop'' is magic, it's what triggers the RBL to only be run on the originating hop The idea is to not penalize (or penalize less) people who properly relayed through their ISP's mail server

Here's an example showing the use of Osirusoft and MAPS DUL, as well as the use of check_two_rbl_results to compensate for a match in both RBLs:

```
header RCVD_IN_DUL             rbleval:check_rbl('dialup', 'dialups.mail-abuse.o
describe RCVD_IN_DUL           Received from dialup, see http://www.mail-abuse.o
score RCVD_IN_DUL              4

header X_RCVD_IN_DUL_FH        rbleval:check_rbl('dialup-firsthop', 'dialups.mai
describe X_RCVD_IN_DUL_FH      Received from first hop dialup, see http://www.ma
score X_RCVD_IN_DUL_FH         -3

header RCVD_IN_OSIRUSOFT_COM   rbleval:check_rbl('osirusoft', 'relays.osirusoft.
describe RCVD_IN_OSIRUSOFT_COM Received via an IP flagged in relays.osirusoft.co

header X_OSIRU_SPAM_SRC        rbleval:check_rbl_results_for('osirusoft', '127.0
describe X_OSIRU_SPAM_SRC      DNSBL: sender is Confirmed Spam Source, penalizin
score X_OSIRU_SPAM_SRC         3.0

header X_OSIRU_SPAMWARE_SITE   rbleval:check_rbl_results_for('osirusoft', '127.0
describe X_OSIRU_SPAMWARE_SITE DNSBL: sender is a Spamware site or vendor, penal
score X_OSIRU_SPAMWARE_SITE    5.0

header X_OSIRU_DUL_FH          rbleval:check_rbl('osirusoft-dul-firsthop', 'rela
describe X_OSIRU_DUL_FH        Received from first hop dialup listed in relays.o
score X_OSIRU_DUL_FH           -1.5

header Z_FUDGE_DUL_MAPS_OSIRU  rblreseval:check_two_rbl_results('osirusoft', "12
describe Z_FUDGE_DUL_MAPS_OSIRU Do not double penalize for MAPS DUL and Osirusof
score Z_FUDGE_DUL_MAPS_OSIRU   -2

header Z_FUDGE_RELAY_OSIRU     rblreseval:check_two_rbl_results('osirusoft', "12
describe Z_FUDGE_RELAY_OSIRU   Do not double penalize for being an open relay on
score Z_FUDGE_RELAY_OSIRU      -2

header Z_FUDGE_DUL_OSIRU_FH    rblreseval:check_two_rbl_results('osirusoft-dul-f
describe Z_FUDGE_DUL_OSIRU_FH  Do not double compensate for MAPS DUL and Osiruso
score Z_FUDGE_DUL_OSIRU_FH     1.5
```

LOCALI[SZ]ATION

A line starting with the text `lang xx` will only be interpreted if the user is in that locale, allowing test descriptions and templates to be set for that language.

SEE ALSO

`Mail::SpamAssassin` `spamassassin` `spamd`