WritingRulesAdvanced

Here's some advanced information on what happens behind the scenes of different rule types.

Body rules: body RULENAME /foo/

Let's assume we have this message content, after any normal decoding of base64/quoted-printable and HTML to text rendering:

> *Subject: This is subject clause.**\n***
> ***\n***
> *First clause of body.   Second clause in first paragraph.**\n***
> *Third clause in first paragraph.**\n***
> ***\n***
> *First clause of second paragraph. Etc.**\n***

(Line breaks / newlines are shown as **\n** for extra clarity)

Body rules are processed in *paragraphs* (blocks of text separated by atleast two newlines), normalized into single lines.

- All paragraphs are whitespace normalized, any single or consecutive whitespace (newline/space/tab/etc) is turned into a single space.
- Newline that ends a paragraph will be preserved, if it exists (message is not required to end in a newline).
- Space at beginning or end of a normalized paragraph is not removed, i.e. using pure anchoring /**^**foo/ might not match " foo".
  - Common practice is to use word boundary matching (/\bfoo/), unless there's a reason to anchor to the start of a *paragraph(!)*.
- Note that if the resulting normalized line is larger than 2048 bytes, it will be split into smaller individual lines, from nearest space boundary if possible.

These three strings (paragraphs turned into lines) will be *individually* tested against the body pattern /foo/, until a match is found.

1)

> *This is subject clause.**\n***

2)

> *First clause of body. Second clause in first paragraph. Third clause in first paragraph.**\n***

3)

> *First clause of second paragraph. Etc.**\n***

Since the matching is separate, pattern like /first paragraph.*Etc/s won't work. You would need to split it into separate subrules:

- body __FOO1 /first paragraph/
- body __FOO2 /Etc/
- meta FOO (__FOO1 && __FOO2)

Note that Subject header is considered a part of the body to simplify rule writing, it will always be the first string to be tested.

- Subject string will always contain an ending newline (**\n**).
- Even if Subject header is missing or empty, the first string will be a single newline (**\n**).
- Starting from SpamAssassin 3.4.3, it's possible to use **tflags nosubject** to skip matching the Subject completely.

When using **tflags multiple**, process is exactly the same, all lines are searched for all matches (until maxhits=x, if defined):

- Pattern /clause/ would result in 5 rule hits.
- Pattern /^./ would result in 3 rule hits.

When using rules with extended characters / diacritics, you should always use both ISO-8859-1 / UTF-8 encodings.
Body content can be different depending on normalize_charset setting. If matching "fügen", see these examples:

- body FOO /fügen/   (**BAD**)
  - Does not work when normalize_charset 1 and mail is converted from ISO-8859-1 to UTF-8
- body FOO /fÃŒgen/   (**BAD**)
  - Does not work when normalize_charset 0 and mail is ISO-8859-1
- body FOO /f(?:\xfc|\xc3\xbc)gen/
  - Works for both encodings, and file is also now very portable and not encoding dependent
  - You can use UTF-8 / ascii table tools found with google, or try perl for hex convert:
    - perl -MEncode -e 'print unpack("H*",encode("UTF-8","ü"))."\n"'
    - perl -e 'print unpack("H*","ü")."\n"'
- You can also try some replace_tags found in default ruleset, that match different variations:
  - body FOO /f<U>gen/

    - replace_tags FOO

As body is processed in raw bytes, **Unicode-regex features like \p{} can not currently be used.**

Rawbody rules: rawbody RULENAME /foo/

Rawbody rules are processed similarly as body, but these are the main differences:

- Instead of paragraphs/lines, text is split into **1-4kB chunks** (depending on SA version and whitespace boundaries found).
- All textual mime parts of the message are used, HTML is not rendered.
- No normalization of text is done, aside from the normal base64/quoted-printable decoding.

It's important to remember that chunks of text are again *individually* tested against a pattern.

- When using anchoring (/^foo/), it will only match the start of a *chunk*.
    - I.e. it's not possible to match a beginning of part 100% accurately, if it's larger than 1-4kB.

Header rules: header RULENAME Header =~ /foo/

If there are multiple headers named "Header", the matched string contains each of the headers, newline separated, starting from first (topmost).

- If **Header:raw** is used, all whitespace and newlines are preserved. Again multiple headers are concatted in the same matching string.
- When using anchors (/^foo/), use m-modifier if any of the duplicate Headers should match. Without, only the first header (line) will match.