Message Rules

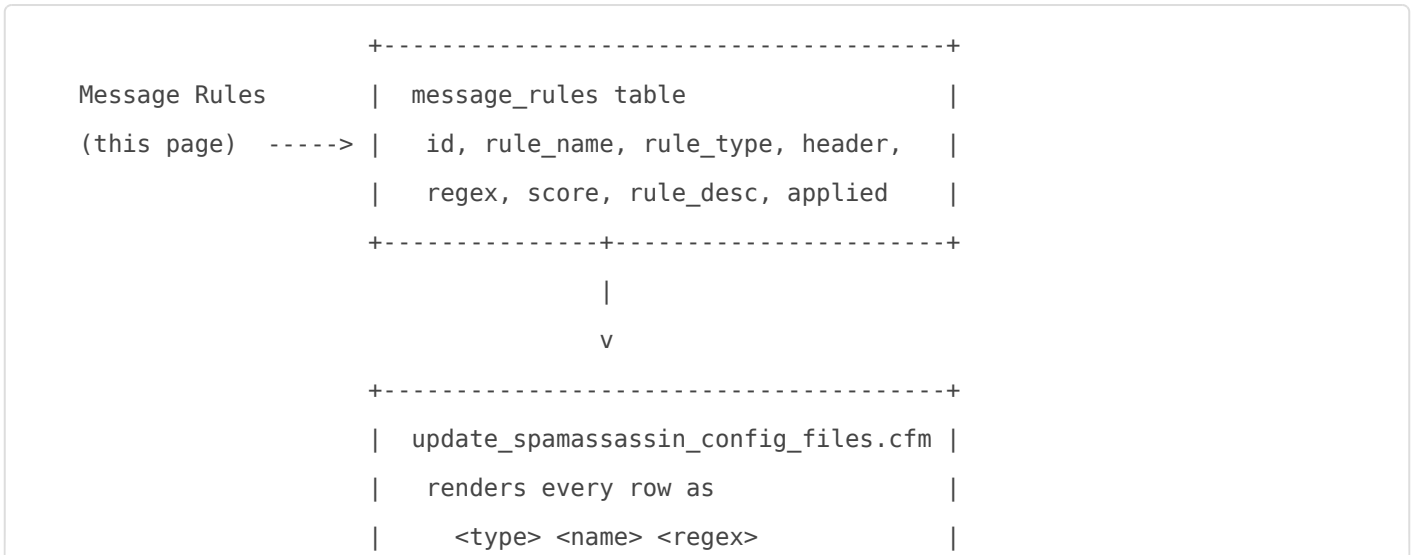
Message Rules

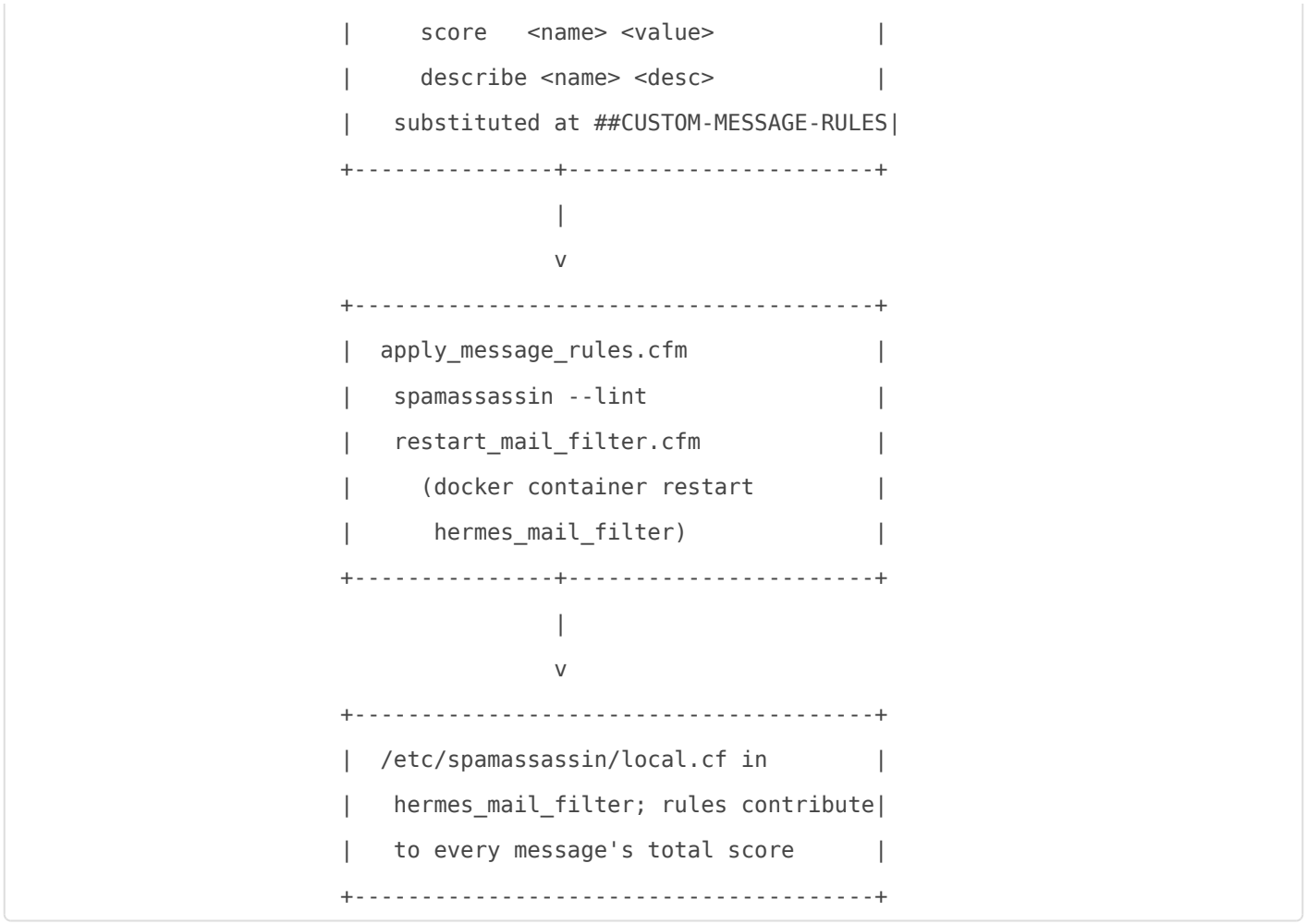
Admin path: **Content Checks > Message Rules** (`view_message_rules.cfm`, `inc/get_message_rules.cfm`, `inc/apply_message_rules.cfm`, `inc/update_spamassassin_config_files.cfm`, `inc/restart_mail_filter.cfm`).

This page maintains a catalogue of **custom SpamAssassin rules** that score against a regex match in a specific part of the message (header, body, raw body, full message, or URI). Every rule a row on this page produces is appended verbatim to SpamAssassin's `local.cf` as a `<type> <name> <regex>` line, paired with a `score <name> <value>` line, and (optionally) a `describe <name> <text>` line. SpamAssassin then runs the rule on every message that reaches the SpamAssassin pass inside Amavis. The cutoff that turns a final score into a `tag` / `quarantine` action is set globally on Anti-Spam Settings; this page only writes the rules themselves.

Message Rules is the body/header equivalent of what File Extensions does for attachment names. Both ride into `local.cf` / `50-user` on save, both are validated with `spamassassin --lint` before the mail filter restarts, but File Extensions matches the trailing extension of an attachment filename while Message Rules matches arbitrary regex against text inside the message.

Where Message Rules sits





A row added here only affects the SpamAssassin pass — it does not reject at SMTP-time, it does not modify headers directly, and it does not bypass content filtering for any recipient. It just adds or subtracts from the final score, and whether that final score crosses a quarantine threshold is decided by the recipient's SVF Policy.

Rule types

Type	What it matches	Cost
`header`	A specific message header (`Subject`, `From`, `Return-Path`, ...) or any header when `ALL` is set	Very cheap; runs against parsed header values
`body`	The decoded plain-text body	Cheap
`rawbody`	The raw/HTML body before SpamAssassin decodes it (good for catching CSS tricks, hidden text, encoded payloads)	Cheap
`full`	The entire raw message including all MIME parts and headers	Most expensive; use sparingly

Type	What it matches	Cost
`uri`	URIs extracted from the message body	Cheap; ideal for catching suspicious link patterns

The Page Guide on the page calls out `full` as resource-intensive because SpamAssassin runs the regex against the whole raw blob; a greedy or expensive regex in a `full` rule can noticeably slow every scan. Prefer `body`, `rawbody`, or `uri` where they cover the case.

Score semantics

The `score` value behaves identically to a Score Overrides weight, except this page creates the rule from scratch instead of overriding a shipped rule:

Score	Effect
Positive (`5`, `20`, etc.)	Adds to the spam score on match. Higher values push the message toward `tag` / `quarantine`
`0`	Rule still runs but contributes nothing — useful for keeping the rule in place during a tuning pass without firing it
Negative (`-3`, `-10`)	Subtracts from the score on match — effectively whitelists messages matching the pattern

The validation accepts any numeric value in the range `-999 .. 999` (per the input's `step="0.01"` and `min`/`max` attributes). The SVF policy assigned to the recipient determines what `total score` threshold triggers tag / quarantine — see SVF Policies.

The page

A Page Guide callout, a collapsible **Regex Helper** card (three tools: rule builder, common-pattern picker, regex tester — all client-side JavaScript that just populates the Add form), an Add Message Rule card, and an Existing Message Rules DataTable.

Add Message Rule card

Field	Stored as	Notes

Rule Name	`message_rules.rule_name`	Required. Letters, numbers, dashes, underscores only — no spaces. Must be unique. SpamAssassin uses this as the rule identifier in logs (`X-Spam-Status` header reports rule names that fired)
Rule Type	`message_rules.rule_type`	Required. One of `header`, `body`, `rawbody`, `full`, `uri`
Header	`message_rules.header`	Required when Rule Type is `header`. Letters, numbers, dashes, underscores only. Datalist suggests common headers (`Subject`, `From`, `Return-Path`, ...) plus the special `ALL` token to match any header. For non-header rules the field is force-cleared on save
Regex Pattern	`message_rules.regex`	Required. A SpamAssassin-format regex like `/keyword/i`. For `header` rules a `~` prefix is auto-added on save (this is the SpamAssassin `=~` operator notation `header_name =~ /pattern/`) and stripped on display so the operator sees only the regex
Score	`message_rules.score`	Required, numeric, `-999 .. 999`
Description	`message_rules.rule_desc`	Optional. Surfaced into the rendered `local.cf` as a `describe` line, which feeds the rule into SpamAssassin's "why was this scored" explanations

The handler validates each field in order and returns to the page with `session.m_rules = <code>` for the first failure (form values are preserved through `session.form_*` so the operator doesn't re-type). Successful insert sets `applied = 2` (pending) before the apply chain runs, then bulk-updates all rows to `applied = 1` once `spamassassin --lint` and the restart succeed.

Regex Helper card

Pure client-side, no server roundtrip:

Tool	What it does
Build a Rule	Pick "match in body / header / raw / URIs," choose Contains / Exact / Starts / Ends / Any-of, type the text, click Build. JavaScript escapes regex metacharacters and assembles a `/pattern/i` string
Quick Select Common Patterns	A `<select>` of pre-built rules for typical spam patterns ("Subject contains lottery winner", "URI: URL shortener", "HTML: hidden text"). Picking one populates the Add form below

Tool	What it does
Test a Pattern	Paste a `/regex/flags` and a sample string; the helper runs JavaScript's `RegExp` against it and reports Match / No match / Invalid regex

This is operator convenience — none of it touches the database or SpamAssassin. The pattern that lands in `message_rules.regex` is exactly what the operator submits, even if it came from one of these helpers.

Existing Message Rules DataTable

Column	Source
(checkbox)	Selection for bulk Delete Selected
Rule Name	`message_rules.rule_name`
Type	`message_rules.rule_type` rendered as a coloured badge per type
Header	`message_rules.header` for `header` rules; `N/A` otherwise
Regex	`message_rules.regex` (with the auto-prefixed `~` stripped for `header` rules so the display matches what the operator typed)
Score	`message_rules.score`
Description	`message_rules.rule_desc`
Actions	Per-row Edit and Delete buttons

Edit reuses the same validation as Add. Rule Name is shown read-only in the modal — to rename, delete and re-add (renaming would orphan any `X-Spam-Status` historical correlation anyway).

Save and apply flow

```
1. View page submits action="add_rule" | "edit_rule" |
   "delete_rule" | "bulk_delete"
2. Action handler validates input, INSERT/UPDATE/DELETE on
   message_rules (applied flag set to '2' = pending on
   add/edit; no applied flag manipulation on delete)
3. cfinclude apply_message_rules.cfm:
     a. cfinclude update_spamassassin_config_files.cfm:
          - Read /opt/hermes/conf_files/local.cf.HERMES (template)
          - Substitute USE-DCC, USE-PYZOR, USE-RAZOR2, USE-BAYES,
```

```
            BAYES-AUTO-LEARN, BAYESAUTOLEARN-SPAM, BAYESAUTOLEARN-HAM
            from spam_settings (the Anti-Spam Settings rows)
          - Append per-rule score overrides
            (#CUSTOM-TESTS placeholder, from spam_settings
            rows where spamfilter=1)
          - Append every message_rules row as
            "<type> <name> <regex>"+"score <name> <value>"
            [+"describe <name> <desc>" if non-blank]
            (#CUSTOM-MESSAGE-RULES placeholder)
          - Back up /etc/spamassassin/local.cf ->
            local.cf.HERMES.BACKUP, move rendered file into place
     b. Write a temp shell script wrapping
          docker exec hermes_mail_filter \
            /usr/bin/spamassassin --lint 2>/dev/null
          exit 0
        (stderr redirected to /dev/null and trailing `exit 0` —
        Lucee otherwise throws on stderr warnings; the lint return
        code is captured into lintOutput)
     c. cfinclude restart_mail_filter.cfm:
          docker container restart hermes_mail_filter
     d. UPDATE message_rules SET applied = '1'
        (mark every row as live)
4. session.m_rules = 1|2|3 -> green alert
5. cflocation back to view_message_rules.cfm
```

A few things worth knowing about this chain:

- **Lint failures do not stop the restart.** The lint output is captured into `lintOutput` but the include does not branch on it; the next step (`restart_mail_filter.cfm`) runs unconditionally. An invalid regex in a new rule will surface in the restarted container's logs (Amavis will refuse to load SpamAssassin, or SpamAssassin will skip the broken rule depending on the failure mode), not in the green alert on the page.
- **`applied = '1'` is set for every row, not just the new one.** The flag tracks "has every rule been pushed to the running SpamAssassin?" rather than "is this specific rule live?" — after the restart, every row is by definition live.
- **The full container is restarted** (`docker container restart hermes_mail_filter`), not `force-reload`d. This is the same restart that Anti-Spam Settings does; outbound mail queues briefly during the restart (typically a few seconds) and Postfix retries.

Failure semantics

Alert	Trigger
`m_rules = 1`	Add Rule succeeded; SpamAssassin validated and reloaded
`m_rules = 2`	Delete (single or bulk) succeeded; SpamAssassin validated and reloaded
`m_rules = 3`	Edit Rule succeeded; SpamAssassin validated and reloaded
`m_rules = 10`	Rule Name is empty
`m_rules = 11`	Rule Name contains characters other than letters, numbers, dashes, underscores
`m_rules = 12`	A rule with that name already exists
`m_rules = 13`	Header field is empty for a `header` rule
`m_rules = 14`	Header field contains invalid characters
`m_rules = 15`	Regex/Pattern is empty
`m_rules = 16`	Score is empty
`m_rules = 17`	Score is not numeric
`m_rules = 18`	Rule Type is not one of `header`, `body`, `rawbody`, `full`, `uri`

The validation order is sequential — the first failure wins and the rest of the validation does not run. Form values are preserved into the next page render via `session.form_*` so the operator sees their submission intact when the error renders.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_message_rules.cfm`	`hermes_commandbox`	The page (validation + Add / Edit / Delete / Bulk Delete + Regex Helper)
`config/hermes/var/www/html/admin/2/inc/get_message_rules.cfm`	`hermes_commandbox`	Loads the full rules list and a count of `applied = 2` (pending) rows
`config/hermes/var/www/html/admin/2/inc/apply_message_rules.cfm`	`hermes_commandbox`	Orchestrates the render + lint + restart + mark-applied chain
`config/hermes/var/www/html/admin/2/inc/update_spamassassin_config_files.cfm`	`hermes_commandbox`	Renders `local.cf` from template, appends every `message_rules` row and every `spamfilter=1` `spam_settings` row
`config/hermes/var/www/html/admin/2/inc/restart_mail_filter.cfm`	`hermes_commandbox`	`docker container restart hermes_mail_filter`

Path	Owner	Role
`config/hermes/opt/hermes/conf_files/local.cf.HERMES`	template (read) -> `hermes_mail_filter` (live `/etc/spamassassin/local.cf`)	Receives the rendered rules at the `#CUSTOM-MESSAGE-RULES` placeholder
`/etc/spamassassin/local.cf.HERMES.BACKUP`	`hermes_mail_filter`	Pre-write backup of the prior live `local.cf`, refreshed each save
`message_rules` table	`hermes_db_server` (`hermes` DB)	Source of truth for every rule on this page
`hermes_mail_filter` container	--	Hosts SpamAssassin under Amavis; full container restart on every save

Related

- SVF Policies -- the per-recipient policy that decides what threshold a rule's score has to push the running total above before Amavis tags or quarantines. A rule scored at `5` does nothing on a recipient whose SVF policy has `spam_kill_level = 12`
- Score Overrides -- overrides for the weights of SpamAssassin's shipped rules; this page **creates** rules, that page **reweights** existing ones. Both ride into `local.cf` on the same save chain
- Anti-Spam Settings -- engine-wide toggles (Bayes, DCC, Razor, Pyzor) and the global `final_*_destiny` quarantine actions. Subject tagging (`sa_spam_subject_tag`) is also set here
- Antivirus Settings -- the ClamAV pass runs before SpamAssassin in the same Amavis call; a virus verdict pre-empts any spam score this page contributes
- File Extensions -- the attachment-name equivalent of this page; both write to `hermes_mail_filter` on save and share the lint-then-restart pattern (File Extensions uses `force-reload` instead of full restart because no SpamAssassin state is touched)
- File Rules -- bundles extensions into named rulesets that bind to SVF policies; orthogonal to message rules but lives in the same Amavis pass
- Perimeter Checks -- nothing on this page matters for messages rejected at SMTP-time; rules here only see the traffic that already passed the perimeter
- Message History -- a quarantined message surfaces the matched rule names in the SpamAssassin verdict details, which is the canonical way to see whether a custom rule fired
- System Logs -- the `amavis[...]` line for a scored message reports `tests=` followed by every rule that fired with its weight, including custom rules from this page

Revision #48
Created 2026-05-31 12:52:28 UTC by Dino Edwards
Updated 2026-06-20 13:33:16 UTC by Dino Edwards