Mailbox Rules

Mailbox Rules

Admin path: **Email Server > Mailbox Rules** (`view_sieve_rules.cfm`, `inc/sieve_rule_actions.cfm`, `inc/sieve_helpers.cfm`, `inc/generate_sieve_global.cfm`, `inc/get_sieve_rule_json.cfm`).

This page manages **global Sieve rules** — server-side filters that run on every message delivered to every mailbox **before** any user-defined Sieve script. Sieve is the IETF mail filtering language (RFC 5228); Dovecot's `sieve` plugin executes it at LMTP delivery time, after Amavis content scanning and just before the message lands in the user's mailbox.

This page is the **admin** side. Mailbox users get a parallel UI in the user portal (`/users/2/view_sieve_rules.cfm`, `scope='user'`) where they can manage their own rules. Global rules always run first and **cannot be overridden** by user rules — they are the right place for organization-wide policy (compliance archiving, mandatory quarantine routing, blanket discards of known-noise patterns).

How Sieve fits the delivery pipeline

```
inbound SMTP -> Postfix -> Amavis (spam/virus) -> Postfix
                                                    |
                                                    v
                                            Dovecot LMTP (port 24)
                                                    |
                                                    v
                                  sieve_before  =  /srv/sieve/global/before.sieve
                                                    |   (this page)
                                                    v
                                  user .sieve scripts (per-mailbox)
                                                    |
                                                    v
                                            final mailbox delivery
```

`sieve_before` is the Dovecot Pigeonhole convention for scripts that run **before** the user's personal script. Hermes wires that to `/srv/sieve/global/before.sieve` (mounted from `/mnt/data/sieve/global/`). The user-portal page writes per-mailbox scripts to `/mnt/data/sieve/<user>/` which run after the global script — and only if the global script does not `discard` or `reject` the message first.

Configuration storage

Each rule is split across three tables to support multi-condition / multi-action rule definitions:

Table	Role
`sieve_rules`	One row per rule. `scope='global'` for admin rules; `scope='user'` (with `username`) for per-mailbox rules. Carries `rule_name`, `rule_order` (top-to-bottom evaluation order), `enabled` (0/1), `is_system` (0/1 — system rules can be toggled but not deleted), `match_type` (`all` = `allof` / AND, `any` = `anyof` / OR).
`sieve_rule_conditions`	One row per condition for the rule. `condition_field` (`subject`, `from`, `to`, `cc`, `bcc`, `header`, `size`, `all`), `condition_type` (`contains`, `is`, `matches`, `not_contains`, `over`, `under`), `condition_value`, `condition_order`. Cascade-deletes when the parent rule is removed.
`sieve_rule_actions`	One row per action. `action_type` (`fileinto`, `discard`, `keep`, `redirect`, `flag_seen`, `reject`), `action_value`, `action_order`. Cascade-deletes with the parent.
`sieve_compile_log`	Append-only log of `sievec` compile errors keyed by `scope` / `username` / `rule_id`. Indexed on `(scope, username)` and `created_at` for the troubleshooting view.

The save handler wraps the child-row delete + re-insert in a single `cftransaction` so a mid-write failure doesn't leave a rule with partial conditions or actions.

How a rule becomes a compiled Sieve script

```
form submit  ──► sieve_rule_actions.cfm
                     |
                     |  validatePayload()   - field/type/value checks
                     |    - rule_name not blank, <= 255 chars
```

```
          |    - >= 1 condition, >= 1 action
          |    - "all" condition cannot coexist with others
          |    - size value matches ^\d+\s*[KMGkmg]?[Bb]?$
          |    - redirect action requires IsValid("email", v)
          |    - per-value length caps (500 cond, 255 act)
          |
          |  --- write DB ---
          |  INSERT/UPDATE sieve_rules
          |  cftransaction:
          |    DELETE child conds + acts for this rule_id
          |    INSERT every cond_field_<i> / cond_type_<i> / cond_value_<i>
          |    INSERT every act_type_<i>  / act_value_<i>
          |
          |  --- generate ---
          v
generate_sieve_global.cfm
    |
    |  read every enabled scope='global' rule (ordered by rule_order)
    |  build "require [...]" header based on action types used
    |    fileinto -> "fileinto", flag_seen -> "imap4flags",
    |    reject   -> "reject",   vacation  -> "vacation"
    |  for each rule:
    |    "## Rule: <name>"
    |    if (single cond):           if <cond> { <actions> }
    |    if (multi-cond, match all): if allof (<cond>, <cond>) { <actions> }
    |    if (multi-cond, match any): if anyof (<cond>, <cond>) { <actions> }
    |    if (all-messages):          (unconditional actions)
    |
    |  cffile write /mnt/data/sieve/global/before.sieve
    |  docker exec hermes_dovecot chown -R 1000:1000 /srv/sieve/global
    |
    v
docker exec hermes_dovecot sievec /srv/sieve/global/before.sieve
    |
    |  stderr non-empty? -> request.sieveCompileError set,
    |                       row inserted into sieve_compile_log,
    |                       session.m = 30 ("saved, but compile failed")
    |                       previous .svbin remains active
    |
    |  stderr empty?     -> session.m = 1/2/3/4 per action
```

```
                |
                v
        cflocation -> view_sieve_rules.cfm
```

The compile-and-keep-old-binary behavior is by design. A broken rule saved into the DB does **not** break delivery — Dovecot continues executing the previous good `.svbin`, and the admin sees the compile error inline in the next page render. Fix and re-save to clear it.

The condition vocabulary

`condition_field`	What it matches	`condition_type` options
`subject`	The `Subject:` header	`contains`, `is`, `matches`, `not_contains`
`from` / `to` / `cc` / `bcc`	The respective address header. **Uses Sieve's `address` test, not `header`** — extracts just the email address, ignoring display name and angle brackets.	`contains`, `is`, `matches`, `not_contains`
`header`	Custom header. Value field is `Header-Name: value` — the first colon splits name from value, so header values containing colons (`X-Custom: foo:bar`) are preserved.	`contains`, `is`, `matches`, `not_contains`
`size`	Message body size. Value accepts `10`, `10M`, `10 MB`, `10mb` — normalized at save time to `10M`.	`over`, `under`
`all`	All messages. Cannot be combined with other conditions in the same rule.	(no type)

`matches` uses Sieve's glob syntax (`*` and `?`), not full regex. Use it for filename-style patterns; use `contains` for substring matches.

The action vocabulary

`action_type`	Effect	Value required?
`fileinto`	Deliver into the named IMAP folder. Use `/` for nested folders (`Work/Projects`). Folder must exist — the global generator does **not** emit `:create` (admin rules don't create folders for users; only the user-side generator does).	Yes

`action_type`	Effect	Value required?
`discard`	Silently drop the message. No delivery, no bounce, no notification. **Irreversible.** Combine with the `all` condition only with extreme care.	
`keep`	Default delivery to INBOX. Useful when chained with `flag_seen` to deliver-and-mark-read.	
`redirect`	Forward the message to another address. **See the Forwarder-trust warning below.**	Yes — must validate as an email address
`flag_seen`	Adds the `\Seen` IMAP flag. Combine with `keep` or `fileinto` to deliver as already-read.	
`reject`	Bounce the message back to the sender with the supplied text. **Leaks that the address exists** — use sparingly.	Yes

The form refuses to save without at least one condition and one action; the action handler re-validates server-side regardless.

The Forwarder-trust warning (#229)

The Action row UI surfaces an explicit warning when `redirect` is selected, because forwarding from a server-side rule breaks all three of the receiver's sender-authentication signals:

Signal	Why it breaks
SPF	The receiver sees Hermes's IP, not an IP authorized by the original sender's SPF record. This break happens on **any** forward, regardless of body modification.
DKIM	If Hermes-side modifiers (external-sender banner, disclaimer, encryption) altered the body, the original sender's `DKIM-Signature` body hash no longer matches.
ARC	If the inbound message had an upstream ARC seal, the same body modification invalidates it. Hermes's own seal honestly records `cv=fail`.

With all three broken, the receiver applies the original sender's DMARC policy — `p=quarantine` or `p=reject` for strict domains means the forward lands in spam or is dropped outright. **Internal redirects** (to a mailbox Hermes itself hosts) are not affected because Hermes never re-evaluates

its own headers. For external destinations, the receiver must be configured to trust this gateway as an authorized forwarder (ARC sealer allow-list, internal-relay exception, etc.) for the redirect to survive DMARC enforcement.

This applies symmetrically to the Sieve `redirect` action on the user-portal side.

Dangerous-combination guards

The save form fires a JavaScript `confirm()` dialog before submitting two specific combinations:

Combination	Warning
`all` condition + `discard` action	"This rule will SILENTLY DELETE every incoming message that reaches a mailbox. This is irreversible. Are you absolutely sure?"
`all` condition + `reject` action	"This rule will REJECT every incoming message and bounce it back to the sender. Are you absolutely sure?"

The guards exist because the global script runs **before** every user's personal rules — a misclick here black-holes the entire mail server for every mailbox. The dialog cancels the submit and explicitly clears the page preloader (the global form-submit hook in `html_head.cfm` shows the preloader before this handler can decide to cancel).

System rules

Rules with `is_system = 1` are seeded by the installer or by future migrations. The UI surfaces them with a **System** badge and:

- The Delete button is **suppressed** in favor of the badge
- The Edit button is **suppressed** — system rules are read-only
- The Enable / Disable toggle still works — admins can turn a system rule off without deleting it
- The action handler's `delete_rule` branch re-checks `is_system` server-side and refuses with error 22 if a crafted POST tries to bypass the missing button

Reorder is allowed on system rules, so an admin can move a system rule above or below a custom rule when the order matters.

The Bcc caveat

The page calls this out explicitly: the `Bcc:` header is **stripped by the MTA before delivery** in almost every case (that is the entire purpose of Bcc). A condition matching the `Bcc` field will therefore rarely fire on incoming mail. The option exists for completeness and for the rare deployments where an upstream relay preserves the header, but rules built around it should not be considered reliable.

Failure semantics

What breaks	What happens
Rule name blank or > 255 chars	`session.m = 10`, no DB write
Zero conditions (or all conditions blank)	`session.m = 11`
Zero actions (or all actions blank)	`session.m = 12`
`size` value fails the `^\d+\s*[KMGkmg]?[Bb]?$` regex	`session.m = 13`
`redirect` action with an invalid email address	`session.m = 14`
`fileinto` or `reject` action with empty value	`session.m = 15`
Condition value > 500 chars or action value > 255 chars	`session.m = 16`
`all` condition combined with any other condition	`session.m = 17`
Delete attempted on a system rule	`session.m = 22`
`sievec` compile error	`session.m = 30`, warning banner with full stderr, **previous compiled script stays active**, error logged to `sieve_compile_log`
`sievec` not reachable (Dovecot container down)	Same path as a compile error — wrapped in `cftry`; `request.sieveCompileError` captures the exception text
Transaction rollback during child re-insert	Rule row UPDATE is rolled back too (the wrapping `cftransaction` covers both); page surfaces the underlying exception

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_sieve_rules.cfm`	`hermes_commandbox`	Page + Add/Edit/Delete modals + reorder/toggle forms
`config/hermes/var/www/html/admin/2/inc/sieve_rule_actions.cfm`	`hermes_commandbox`	Action handler — validate, write DB, regenerate, compile
`config/hermes/var/www/html/admin/2/inc/generate_sieve_global.cfm`	`hermes_commandbox`	Reads `sieve_rules` + children, writes `before.sieve`, runs `sievec`

Path	Owner	Role
`config/hermes/var/www/html/admin/2/inc/sieve_helpers.cfm`	`hermes_commandbox`	Shared condition/action string builders (used by global + user generators)
`config/hermes/var/www/html/admin/2/inc/get_sieve_rule_json.cfm`	`hermes_commandbox`	AJAX hydrator for the Edit modal
`/mnt/data/sieve/global/before.sieve`	`hermes_dovecot` (mounted from host)	Live global script — overwritten on every save
`/mnt/data/sieve/global/before.svbin`	`hermes_dovecot` (mounted from host)	Compiled binary that Dovecot actually executes
`/mnt/data/sieve/<user>/*.sieve`	`hermes_dovecot` (mounted from host)	Per-mailbox user scripts (managed by the user portal, not this page)
`sieve_rules`, `sieve_rule_conditions`, `sieve_rule_actions`, `sieve_compile_log`	`hermes_db_server`	The rule definition + compile-error log

`sievec` is the Pigeonhole compiler. It **must** run inside the Dovecot container because the resulting `.svbin` format is plugin-version-sensitive and tied to the `pigeonhole` build Dovecot loads at runtime. Running it on the host would produce a binary Dovecot can't load.

Related

- Mailboxes — global rules run against every mailbox on every domain. There is no per-mailbox or per-domain scoping at the global tier — use conditions on `to`, `from`, or a custom header to scope.
- Domains — `domains.allow_user_signatures` is the closest per-domain user-rule toggle Hermes has today. There is no separate per-domain toggle for user Sieve rules; the user-portal Sieve editor is always available to mailbox users.
- Settings — Dovecot's `sieve` plugin and the `sieve_before` directive are configured globally there. The per-rule pieces this page edits sit underneath that global wiring.
- Aliases — silent-discard aliases are an alternative to a Sieve `discard` rule when the goal is to nuke mail to a specific address rather than match on content.
- Shared Mailboxes — global Sieve runs on shared-mailbox delivery too. A `fileinto` rule referencing a shared mailbox path will work as long as the folder exists.
- Email Relay > Relay Recipients — relay recipients do **not** receive Dovecot LMTP delivery (mail is forwarded out via Postfix `smtp_*` instead), so global Sieve rules do not run against relay-bound mail. Use Amavis policies or the body milter for relay-side filtering instead.

Revision #14
Created 2026-05-31 12:52:15 UTC by Dino Edwards
Updated 2026-06-13 12:30:13 UTC by Dino Edwards