# File Rules

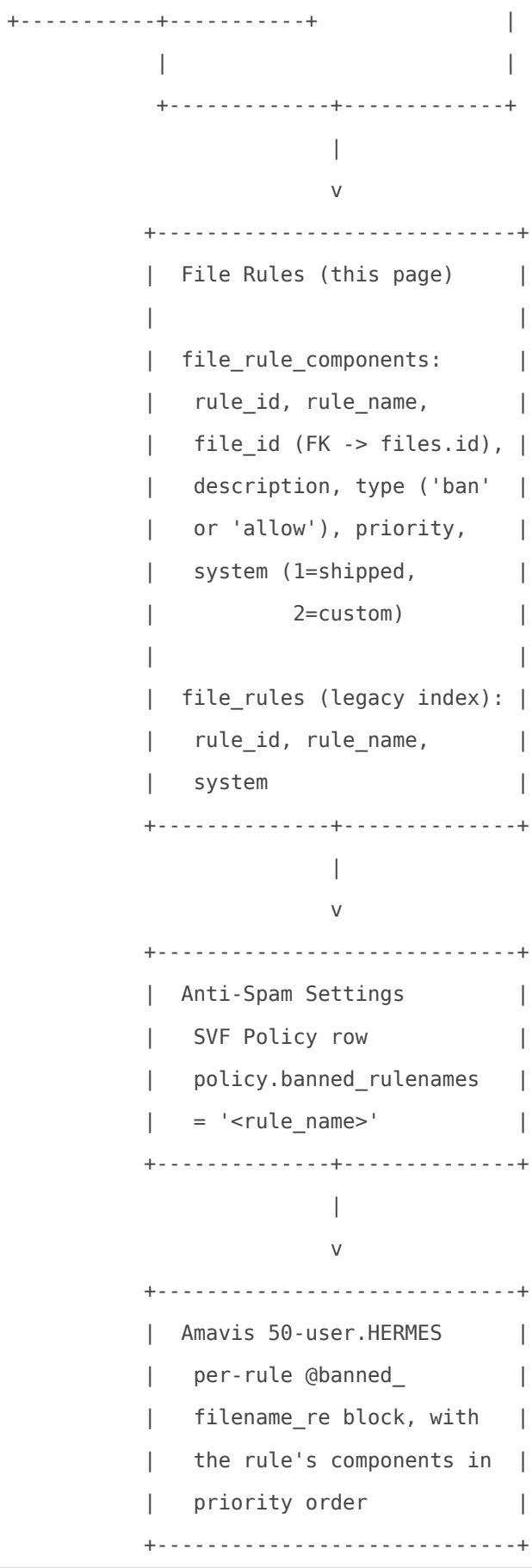
# File Rules

Admin path: **Content Checks > File Rules** (`view_file_rules.cfm`, `inc/get_file_rules.cfm`, `inc/update_amavis_config_files.cfm`).

This page is the **bundling layer** that turns the raw catalogues on File Extensions and File Expressions into named, prioritised rulesets that Amavis can actually enforce. A File Rule is a named group of file-type components (extensions, file types, MIME types, high-risk variants of each, and custom regex expressions) plus a default action (**Ban** or **Allow**) that the operator binds to recipient traffic via an SVF Policy under Anti-Spam Settings. Without a File Rule wrapping them, no row on the catalogue pages does anything to mail.

Hermes ships one system rule, **SYSTEM_DEFAULT**, populated with a broad ban list (executables, scripts, Windows-class-IDs, double-extension trap, archive formats, dangerous MIME types). It is read-only — it can be copied, but not edited or deleted. Every custom rule the operator creates lives alongside it in the same DataTable, marked **No** in the System Rule column.

# Where File Rules sits




A File Rule that is created but **not** bound to an SVF Policy is inert. The rule renders into Amavis's config (`50-user` carries every defined rule), but no recipient policy points at it, so nothing in

`@banned_filename_re` fires for traffic.

# The two backing tables

| Table | Role |
|---|---|
| `file_rule_components` | The real source of truth. One row per (rule, file-type) pair. Carries `rule_id`, `rule_name`, `file_id` (FK -> `files.id`), `description`, `type` (`ban` or `allow`), `priority`, `system` (1 = shipped, 2 = custom) |
| `file_rules` | A legacy index table holding only `rule_id`, `rule_name`, `system`. Hermes ships a single row in it (`SYSTEM_DEFAULT`, `system=1`) — the page's CRUD operations write to `file_rule_components` directly and the Delete handler also clears `file_rules` for the matching `rule_id`. New rules are NOT inserted into `file_rules`; rule existence is determined entirely by `DISTINCT rule_id` on `file_rule_components` |

The `system` value is the system / custom discriminator and is the guard for every modify path:

- `system = 1` -> shipped (SYSTEM_DEFAULT only). Read-only — attempting to edit or delete returns `m = 24`. The Copy button still works.
- `system = 2` -> operator-added. Editable and deletable, subject to the policy-binding guard.

The action column is named `type` (not `action`) on `file_rule_components` and is per-component: a single rule can mix **ban** and **allow** components, although the page's UI surfaces "Default Action" as a single radio button and assigns the same value to every component on save. Mixing `ban` and `allow` on the same rule is possible only by direct SQL.

# The page

A page guide callout, a single DataTable listing every rule (system and custom together), and three modals: Create Custom File Rule (`Add`), Edit File Rule, and Copy File Rule.

# File Rules DataTable

| Column | Source |
|---|---|
| Rule Name | `file_rule_components.rule_name` (distinct) |
| Type | Rendered from the first component's `type` — `<span class="badge bg-danger">Ban</span>` or `<span class="badge bg-success">Allow</span>` |

| Column | Source |
|---|---|
| File Types | Every component's `description` as a list of `bg-secondary` badges, each suffixed with `(ban)` or `(allow)` |
| System Rule | `Yes` (info badge, `system=1`) or `No` (warning badge, `system=2`) |
| Actions | Copy (always present) + Edit + Delete (only when `system=2`) |

Default sort is **System Rule asc, Rule Name asc**, so the shipped rule sinks below the custom ones once any exist (custom = `system=2` sorts above shipped = `system=1`? No — `2` > `1`, but the column order asc is intentional: shipped first, then custom alphabetised). The DataTable carries `stateSave: true`, so the operator's sort / search / page-size choices persist across page loads.

# Create Custom File Rule modal (Add)

| Field | Stored as | Notes |
|---|---|---|
| Rule Name | `file_rule_components.rule_name` | Regex-validated against `[^_a-zA-Z0-9-]` — letters, numbers, dashes, underscores only. No spaces, no punctuation. Max length 50. Duplicates across both system and custom rules are rejected (`m = 22`) |
| Default Action | `file_rule_components.type` on every inserted component | Radio: `ban` (default) or `allow` |
| File Type checkboxes | One INSERT per checked box into `file_rule_components` | Eight grouped cards: High Risk Extensions, High Risk File Types, High Risk MIME Types, File Extensions, File Types, MIME Types, Other Types, Custom Expressions. Each card has a "select-all" master checkbox and a scrollable list of every `files` row of that `type`. At least one file type must be selected (`m = 23`) |

The handler computes the next `rule_id` as `MAX(rule_id) + 1` (scoped across `file_rule_components`, not `file_rules`), assigns `priority` sequentially as components are inserted (1, 2, 3, ... in submission order), and marks each row `system = 2`.

# Edit File Rule modal

Opens preloaded with the current rule's name, default action, and checkbox selections — the JavaScript reads a `ruleComponents` map written into the page at render time and ticks the matching checkboxes across all eight category cards.

Save is destructive-then-rebuild: the handler **DELETEs** every `file_rule_components` row for the rule_id, then re-INSERTs from the new form selection. The same name / action / file-types validation as Add applies, plus:

- System rules (`system=1`) are refused with `m = 24`. The button is not even rendered for system rows, but the action handler still guards against forged POSTs.
- If the rule name changed, the handler also UPDATEs `policy.banned_rulenames` so any SVF Policy binding survives the rename. The cascade is name-keyed, not id-keyed — the policy table stores the name string, not the rule_id.

# Copy File Rule modal

The only path to derive a new rule from SYSTEM_DEFAULT. Asks for a new name (same `[a-zA-Z0-9_-]+` validation, same duplicate check, same 50-char max), then INSERTs a fresh set of `file_rule_components` rows under a new `rule_id` with all the source rule's `file_id`, `description`, `type`, and `priority` values preserved. The copy is always `system = 2` regardless of the source's flag — so a copy of SYSTEM_DEFAULT becomes a fully editable custom rule.

The default new-name in the modal is `<source>_copy`, so the operator can hit Copy on SYSTEM_DEFAULT and immediately get `SYSTEM_DEFAULT_copy` ready to edit.

# Policy-binding guard on delete

A custom rule cannot be deleted while any SVF Policy points at it. The Delete handler runs:

```
SELECT policy_name FROM policy
WHERE banned_rulenames = '<rule_name>'
```

If any row comes back, the delete is refused with alert `m = 25` and the policy name(s) are surfaced in the alert ("You cannot delete a file rule that is assigned to SVF Policy: **Default,Inbound-Strict**. Remove the assignment first under Content Checks > SVF Policies.").

This is the symmetric counterpart to the FK guard on File Extensions and File Expressions — those pages refuse to delete a row that is bundled into a rule; this page refuses to delete a rule that is bundled into a policy.

# Save and apply flow

```
1. View page submits action="add_rule" | "edit_rule" | "delete_rule"
                          | "copy_rule"
2. Validate name (non-empty, regex-clean, non-duplicate, non-system
   on edit/delete), validate file_ids (non-empty)
3. For Add / Edit / Copy:
     a. Determine rule_id (next MAX+1 for Add/Copy, form value for Edit)
     b. (Edit only) UPDATE policy.banned_rulenames if rule_name changed
     c. (Edit only) DELETE existing file_rule_components for rule_id
     d. INSERT one file_rule_components row per checked file_id, with
        priority assigned sequentially (1..N) and system='2'
   For Delete:
     a. DELETE FROM file_rules WHERE rule_id = :id
     b. DELETE FROM file_rule_components WHERE rule_id = :id
4. update_amavis_config_files.cfm:
     - Read /opt/hermes/conf_files/50-user.HERMES (template)
     - Substitute SERVER/destiny/DKIM/MySQL-credential placeholders
     - Loop every DISTINCT rule_id in file_rule_components
       and emit a per-rule @banned_filename_re block in
       priority order, using each component's allow or ban
       regex from files.allow / files.ban
     - Back up /etc/amavis/conf.d/50-user -> 50-user.HERMES,
       move rendered file into place
5. docker exec hermes_mail_filter /etc/init.d/amavis force-reload
   (60-second timeout - longer than the catalogue pages because
    every rule re-renders)
6. session.m = 1 (add) | 2 (edit) | 3 (delete) | 4 (copy)
              | 10 (reload error) | 20-25 (validation refusals)
```

Amavis is reloaded with `force-reload` rather than restarted. If the reload itself fails, the rule rows are already committed — alert `m = 10` ("Configuration Error") fires but the DB is not rolled back. The next successful save (or a manual `force-reload`) will re-render.

# Failure semantics

| Alert | Trigger |
|---|---|
| `m = 1` | Rule created. The alert also nudges the operator to assign the rule to a policy under SVF Policies — without that binding the rule is inert |

| Alert | Trigger |
|---|---|
| `m = 2` | Rule updated; Amavis reloaded |
| `m = 3` | Rule deleted; Amavis reloaded |
| `m = 4` | Rule copied. Same nudge as `m = 1` — the copy is inert until bound to an SVF Policy |
| `m = 10` | Amavis reload error — the DB write succeeded but `force-reload` returned non-zero. Open Anti-Spam Settings and save once to re-trigger the render + reload, or restart `hermes_mail_filter` manually |
| `m = 20` | Rule name field empty |
| `m = 21` | Rule name contains characters outside `[a-zA-Z0-9_-]` (spaces, dots, slashes, etc.) |
| `m = 22` | Duplicate rule name — checked against both system and custom rules |
| `m = 23` | No file types selected — at least one checkbox across the eight category cards is required |
| `m = 24` | Attempted to edit or delete a system rule (`system=1`) — refused. The operator's path is to Copy first, then edit the copy |
| `m = 25` | Delete refused — the rule is bound to one or more SVF Policies (policy names surfaced in the alert) |

# Files and containers touched

| Path | Owner | Role |
|---|---|---|
| `config/hermes/var/www/html/admin/2/view_file_rules.cfm` | `hermes_commandbox` | The page (CRUD + Copy + DataTable + three modals + Amavis reload) |
| `config/hermes/var/www/html/admin/2/inc/get_file_rules.cfm` | `hermes_commandbox` | Loads the rule list + every `files` row grouped by `type` for the modal cards (`get_files_ext_high`, `get_files_file_high`, ..., `get_files_custom_expr`) |
| `config/hermes/var/www/html/admin/2/inc/update_amavis_config_files.cfm` | `hermes_commandbox` | Renders `50-user` from template + every File Rule's components |
| `config/hermes/opt/hermes/conf_files/50-user.HERMES` | `hermes_commandbox` (read) -> `hermes_mail_filter` (live `/etc/amavis/conf.d/50-user`) | Canonical Amavis template; receives the per-rule `@banned_filename_re` blocks |
| `/etc/amavis/conf.d/50-user` | `hermes_mail_filter` | Live Amavis config; reloaded with `force-reload` on every save |

| Path | Owner | Role |
|---|---|---|
| `file_rule_components` table | `hermes_db_server` (`hermes` DB) | The real rule store — one row per (rule, file-type) pair |
| `file_rules` table | `hermes_db_server` (`hermes` DB) | Legacy index — only SYSTEM_DEFAULT lives here; custom rules are NOT mirrored. Cleared on delete for the matching `rule_id` |
| `files` table | `hermes_db_server` (`hermes` DB) | Source of the file-type checkboxes; FK target of `file_rule_components.file_id` |
| `policy` table, `banned_rulenames` column | `hermes_db_server` (`hermes` DB) | Where SVF Policies record their rule binding; renamed in step with rule renames, checked by the delete guard |
| `hermes_mail_filter` container | — | Hosts Amavis; receives `force-reload` (not restart) on every change |

# Operational consequences

- **A rule with no policy binding is inert.** Creating a rule does not block anything by itself — Amavis renders the rule into `50-user` but no recipient policy points at it. The "Please assign the rule to a policy under Content Checks > SVF Policies" nudge in `m = 1` and `m = 4` is the operational reminder. Until the binding is in place the rule exists for the operator's benefit only.
- **Edit is destructive-then-rebuild.** Saving an edit DELETEs and re-INSERTs every component for the rule. Priorities are reassigned 1..N in checkbox-submission order, which is the page render order, not the order the operator originally added them. An edit that only adds one new file type will reshuffle the priority numbers of every existing component on that rule. Functionally invisible (`@banned_filename_re` evaluation is any-match), but visible if anyone reads the table directly.
- **Renames cascade through `policy.banned_rulenames`.** The page joins on name, not id — when the rule name changes, the policy row is updated in the same transaction. If a policy binding exists, the operator does not need to re-open the SVF Policy page after a rename.
- **Copy is the only path off SYSTEM_DEFAULT.** The shipped rule is hard-locked (`m = 24` on any edit / delete attempt). Operators who want to tighten the defaults (add `.iso`, remove `.rtf`, swap MIME types) make a copy, edit the copy, and bind the copy to the policy in place of SYSTEM_DEFAULT.
- **The Type badge shows the first component's action only.** A hand-mixed rule (`ban` and `allow` components on the same rule) will display whichever was inserted at priority 1. The DataTable does not flag mixed rules — the File Types column shows each component's `(ban)` / `(allow)` suffix, which is the only place the mix is surfaced. The UI itself only writes uniform rules.

- **Amavis reload timeout is 60s here, vs 30s on the catalogues.** Re-rendering every rule's `@banned_filename_re` block can take longer than re-rendering a single allow/ban regex for an added extension. If the reload times out, the page shows `m = 10` and the rule write still succeeded.

# Related

- File Extensions — the plain-extension half of the file-type catalogue; rows here become checkboxes in this page's modals under "High Risk Extensions" and "File Extensions"
- File Expressions — the regex half of the file-type catalogue; rows there become checkboxes under "Custom Expressions"
- Message Rules — the body / header equivalent of File Rules; binds SpamAssassin rules to scope rather than Amavis filename patterns
- Anti-Spam Settings — where File Rules are bound to recipient traffic via SVF Policies ( `policy.banned_rulenames`) and where `final_banned_destiny` (the action on a match) is set
- Antivirus Settings — ClamAV runs in the same Amavis pass; a virus verdict on the same attachment overrides the banned-rule result
- Score Overrides — sibling Amavis tuning page; both write into the same `50-user` regeneration chain but rule matches are categorical where SA score overrides are weighted
- ARC Settings — note that banned-rule rejections are a body-side filter result, not an authentication result — they fire after ARC chain evaluation
- Message History — a banned-rule rejection appears with Type `Banned` and the matched rule + component surfaced in the detail view
- System Logs — Amavis logs the rule name and matched component as `Blocked BANNED ('<rule_name>' matched)` on the `amavis[...]:` line

---

Revision #8
Created 2026-05-31 12:52:25 UTC by Dino Edwards
Updated 2026-05-31 14:01:20 UTC by Dino Edwards