File Expressions

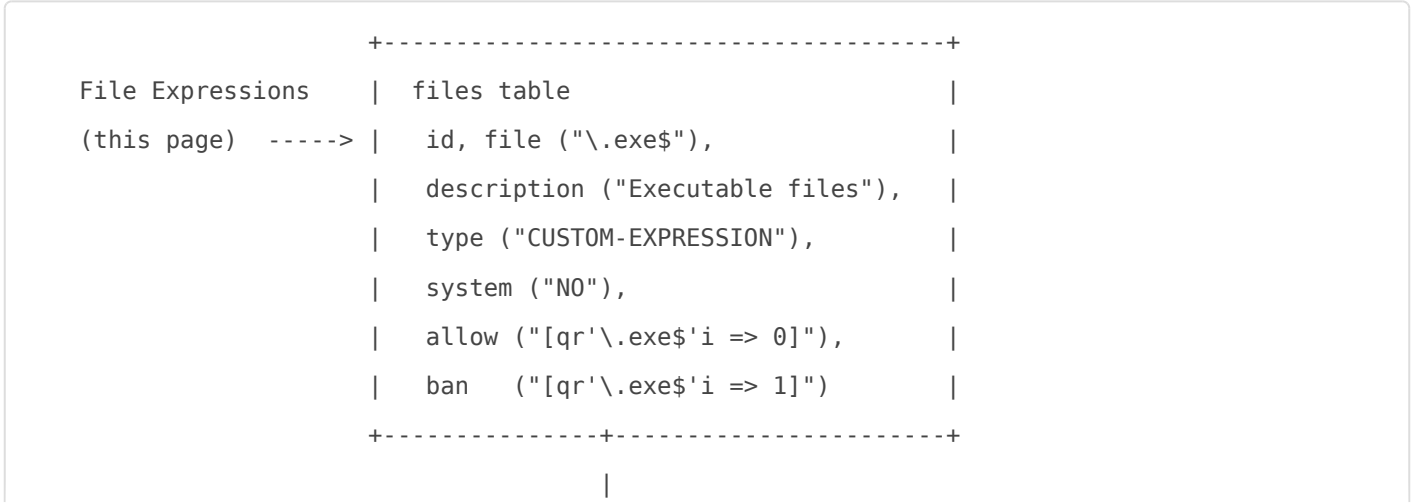
File Expressions

Admin path: **Content Checks > File Expressions** (`view_file_expressions.cfm`, `inc/get_file_expressions.cfm`, `inc/update_amavis_config_files.cfm`).

This page maintains the catalogue of **regex patterns** that Amavis can match against attachment filenames. Where File Extensions is a one-extension-per-row list (`.exe`, `.docm`, `.iso`), File Expressions is the free-form regex sibling — any Perl-compatible pattern that should fire on the attachment name: double-extension traps (`^.+\.(exe|scr)\.[a-z0-9]+$`), disguised-archive patterns (`^invoice.*\.pdf\.zip$`), or any project-specific filename signature an extension list can't express. The page itself does not block anything — it only registers patterns. The block / allow decision is taken by a File Rule that bundles expressions (and extensions, file types, MIME types) into a named ruleset, which is then bound to recipient traffic via an SVF policy on Anti-Spam Settings.

The expression catalogue is **entirely operator-driven** — Hermes ships no system-managed expressions. The shipped High-Risk catch-all ("Double Extensions in File Name") and the Windows Class ID block live on the File Extensions page as `type = 'FILE-HIGH'` rows. Everything on the File Expressions page is something the operator added.

Where File Expressions sits



The rendered `@banned_filename_re` block is enforced at content-filter time inside `hermes_mail_filter`. A matched expression triggers Amavis's `final_banned_destiny` action (`D_BOUNCE`, `D_DISCARD`, or `D_PASS` — set globally on Anti-Spam Settings).

How the pattern is wrapped

The textarea takes a raw Perl regex. On save the handler wraps it into Amavis's `qr//` syntax with the `i` (case-insensitive) modifier and stores both the allow and ban form on the row:

```
[qr'\.exe$'i => 0]      (allow form, stored in files.allow)
[qr'\.exe$'i => 1]      (ban form,  stored in files.ban)
```

Whether the allow or ban form gets rendered into Amavis's `@banned_filename_re` is decided at File Rule time, not here. The File Expressions page does not have an allow/ban toggle — both forms are stored so the same expression can serve allow-rules and ban-rules without re-typing.

There is no case-sensitive variant on this page. Every File Expression is stored with the `i` modifier. Operators who need strict case have to drop down to the File Rule's per-component selection or use a regex character class on the pattern itself (`\.[Ee][Xx][Ee]$`).

The page

A page guide callout, an Expression Helper card (build / pick / test), an Add Expressions card with a bulk textarea, and a single DataTable listing every custom expression. The DataTable is flat — system vs. custom does not apply because the catalogue is all-custom by design.

Expression Helper card

A three-section utility, collapsed by default, that exists so operators don't need to know regex to add common patterns.

Section	Purpose
Build an Expression	Pick a match mode (Ends with / Starts with / Contains / Exact), enter plain text, click **Build**. The helper regex-escapes the input, wraps it with the appropriate anchors (`^…` , `…$` , `^…$`), and shows the generated pattern with a plain-English explanation
Quick Select Common Patterns	A dropdown of pre-built patterns (`\.exe$` , `\.bat$` , `^invoice` , `\.(exe|bat|cmd|scr|pif)$` , etc.) — click **Use** to drop the pattern into the Add form
Test a Pattern	A pattern + filename pair with a **Test** button — runs `new RegExp(pattern, 'i').test(filename)` in the browser and reports Match / No match / Invalid regex. Lets the operator sanity-check before saving

The Build helper escapes `. * + ? ^ $ { } ( ) \| [ ] \` in the user input before wrapping, so a builder entry of `invoice.pdf` becomes `invoice\.pdf$` , not `invoice.pdf$` .

Add File Expressions card

Field	Stored as	Notes
File Expressions	`files.file` (the regex) + `files.description`	One per line; format is `regex_pattern description` where the first space separates pattern from label. A pattern with no space becomes its own description (useful for self-documenting patterns like `\.docm$`)

The handler line-splits the textarea on LF or CRLF, strips whitespace, and inserts each non-blank entry. Per entry it checks one thing: that no row already exists in `files` with the same `file` value under `type = 'CUSTOM-EXPRESSION'`. Duplicates are skipped and surfaced in the partial-success alert ("Duplicate: \.exe$"); the rest still insert.

There is **no regex-validity check on save** — the regex is stored as-typed and any syntax error is exposed at Amavis reload time, not in the alert. Use the Test a Pattern section of the helper before saving to catch malformed patterns first.

File Expressions DataTable

Column	Source
(checkbox)	Selection for bulk Delete Selected
Regex Pattern	`files.file` (rendered inside a `<code>` block)
Description	`files.description`
Actions	Per-row Delete button (single-row confirm)

The DataTable shows only `type = 'CUSTOM-EXPRESSION'` rows. No edit-in-place — to change a pattern the operator deletes it and re-adds.

Foreign-key guard on delete

A custom expression cannot be deleted while it is referenced by any File Rule. The single-row Delete handler runs:

```
SELECT COUNT(*) AS cnt FROM file_rule_components
WHERE file_id = :id
```

If `cnt > 0`, the delete is refused with alert `m = 40` and the DataTable shows the offending rule name(s) ("This expression is referenced by the following File Rule(s): **Block-Disguised-Exe**"). The operator's path is to open File Rules, remove the expression from the rule, then come back here and delete it.

Bulk Delete applies the same guard per-id and accumulates partial results — alert `m = 41` reports "N deleted, M blocked" with the blocked rows' pattern and rule names attached, so the operator knows exactly what to unwire first.

Save and apply flow

```
1. View page submits action="add_entries" | "delete" | "bulk_delete"
2. For each valid entry:
     a. Generate ban  string: "[qr'<pattern>'i => 1]"
     b. Generate allow string: "[qr'<pattern>'i => 0]"
     c. INSERT INTO files (file, description, type, system, allow, ban)
        with type='CUSTOM-EXPRESSION' and system='NO'
3. If at least one row was added or deleted:
     a. update_amavis_config_files.cfm:
          - Read /opt/hermes/conf_files/50-user.HERMES (template)
          - Substitute the SERVER/destiny/DKIM/MySQL-credential
            placeholders from spam_settings and creds files
          - Render every File Rule's components into an
            @banned_filename_re block (per-rule, in priority order,
            using the allow/ban regex stored on each files row -
            including the CUSTOM-EXPRESSION rows this page creates)
          - Back up /etc/amavis/conf.d/50-user -> 50-user.HERMES,
            move rendered file into place
     b. docker exec hermes_mail_filter /etc/init.d/amavis force-reload
        (30-second timeout)
4. session.m = 1 (add) | 2 (single/bulk delete) | 30 (empty submit)
   | 40 (FK refused) | 41 (bulk partial)
```

Amavis is reloaded with `force-reload` rather than restarted — the daemon re-reads `50-user` without dropping connections, and mail in flight is not interrupted. The reload step is wrapped in `cftry`/`cfcatch` and the catch block is intentionally silent: if the reload itself fails the DB rows are already in place, and the next save (or a manual `force-reload`) will re-render. The page does not roll back on reload failure.

Failure semantics

Alert	Trigger
`m = 1`	Add Expressions completed (with `entries_added` / `entries_skipped` / `entry_errors` set on session for the per-row breakdown)
`m = 2`	Single Delete succeeded; Amavis reloaded
`m = 30`	Add submitted with an empty textarea
`m = 31`	Pattern field empty (legacy edit path, no longer reachable from the current UI)
`m = 32`	Duplicate pattern (legacy edit path)

Alert	Trigger
`m = 40`	Single Delete refused — the expression is wired into at least one File Rule (rule names surfaced in the alert)
`m = 41`	Bulk Delete partial — `deleted_count` rows removed, `blocked_count` rows refused (the per-row pattern + rule-name list is HTML-rendered into the alert body)

The per-row error list is HTML-rendered into alert `m = 1` so the operator sees every duplicate at once. No row is silently dropped without an explanation.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_file_expressions.cfm`	`hermes_commandbox`	The page (add + delete + bulk delete + Expression Helper + Amavis reload)
`config/hermes/var/www/html/admin/2/inc/get_file_expressions.cfm`	`hermes_commandbox`	Loads `type = 'CUSTOM-EXPRESSION'` rows into the DataTable
`config/hermes/var/www/html/admin/2/inc/update_amavis_config_files.cfm`	`hermes_commandbox`	Renders `50-user` from template + File Rules (called on every change here too — expression edits affect rendered `@banned_filename_re` blocks)
`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 rendered `@banned_filename_re` blocks
`/etc/amavis/conf.d/50-user`	`hermes_mail_filter`	Live Amavis config; reloaded with `force-reload` on every save
`files` table, `type = 'CUSTOM-EXPRESSION'`	`hermes_db_server` (`hermes` DB)	Source of truth for the expression catalogue
`file_rule_components` table	`hermes_db_server` (`hermes` DB)	Cross-reference checked by the delete guard
`hermes_mail_filter` container	—	Hosts Amavis; receives `force-reload` (not restart) on every change

Operational consequences

- **No regex validation at save.** A malformed regex inserts cleanly and only surfaces at Amavis reload time. The reload itself does not roll back the DB. If reload starts failing immediately after an Add, the most recent expression is the suspect — open it, paste it into the Test a Pattern helper, and look for unescaped metacharacters or unbalanced

groups. The pattern with `\.exe$` works; a typo of `\.exe$.` (trailing dot) parses but matches nothing.

- **Case is always insensitive.** Every expression renders with the `i` modifier. There is no per-expression case toggle. Operators who need strict case have to encode it in the pattern itself.
- **Order does not matter on this page.** Expressions are stored flat. The evaluation order that Amavis sees is decided by the File Rule that bundles them — each component's `priority` column on `file_rule_components`. Changing the description here will not reorder anything.
- **Custom-Expression rows are visible to File Rules under "Custom Expressions".** When the operator opens the Add/Edit modal on File Rules, every row this page creates shows up in the **Custom Expressions** card alongside the system catalogue. That is the only place the bundling happens.

Related

- File Extensions — sibling page for plain extension entries (`.exe`, `.docm`); the simpler half of the same `files` table, distinguished by `type IN ('EXT', 'EXT-HIGH')`
- File Rules — bundles extensions and expressions into named, prioritised rulesets; the consumer of every row this page creates
- Message Rules — content-level SpamAssassin rules (header / body / regex) — the body / header equivalent of what File Expressions does for attachment names
- Anti-Spam Settings — defines `final_banned_destiny` (what Amavis does with a banned-expression match) and binds File Rules to recipients via SVF Policies
- Antivirus Settings — ClamAV runs in the same Amavis pass; a virus verdict on the same attachment overrides the banned-expression result
- Score Overrides — sibling Amavis tuning page; both write into Amavis configuration but expression matches are categorical (matched -> banned) where SA rules are weighted
- ARC Settings — note that banned-expression rejections are a body-side filter result, not an authentication result — they fire after ARC chain evaluation
- Message History — a banned-expression rejection appears with Type `Banned` and the matched expression surfaced in the detail view
- System Logs — Amavis logs the matched regex as `Blocked BANNED (\.exe$,…)` on the `amavis[...]:` line

---

Revision #14
Created 2026-05-31 12:52:24 UTC by Dino Edwards
Updated 2026-06-13 12:30:18 UTC by Dino Edwards