

# Email Policies

- [Disclaimers](#)
- [External Banner](#)
- [Organizational Signatures](#)

# Disclaimers

# Disclaimers

*Pro Edition feature.* Maps to **Email Policies > Disclaimers** (`view_disclaimers.cfm`, `edit_disclaimer.cfm`, `disclaimer_delete.cfm`).

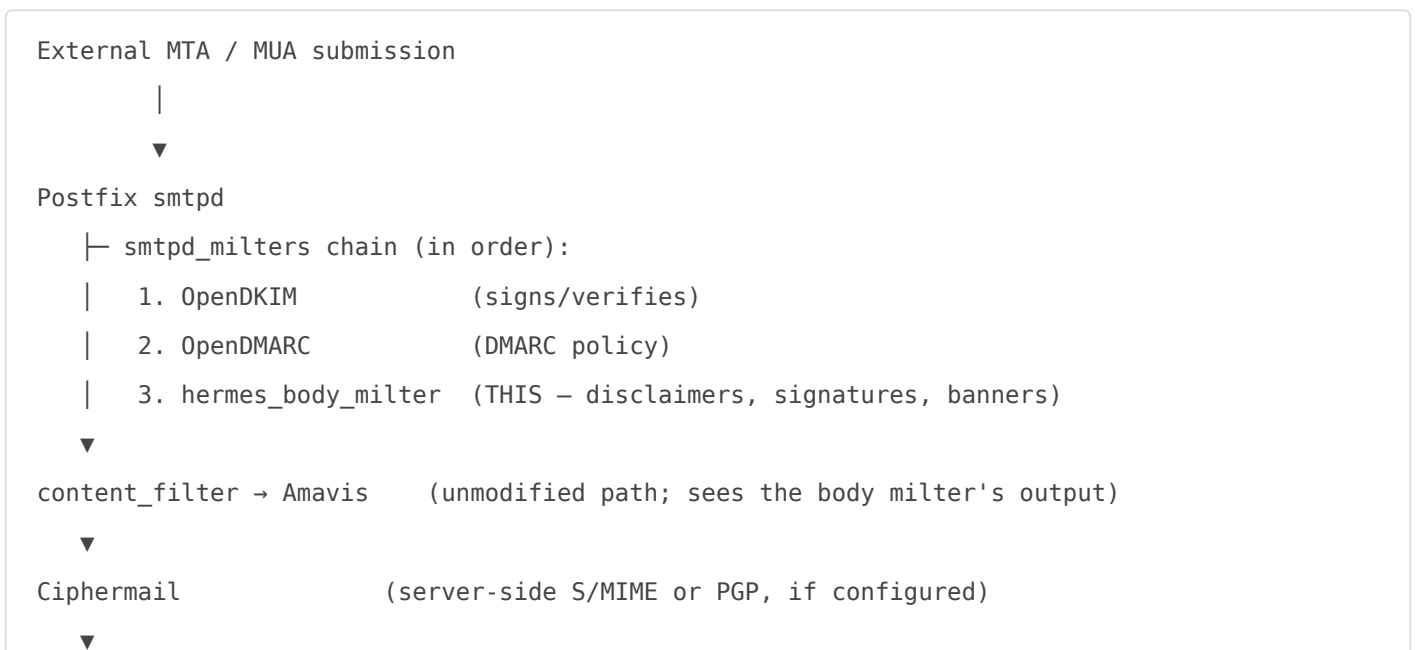
Hermes appends a configurable disclaimer to outbound mail at the gateway, with two scopes:

Scope	Sender match	Use case
<b>Domain</b>	All senders in <code>@example.com</code>	Default org-wide compliance/legal language
<b>Relay Recipient</b>	Specific full address (e.g. <code>vendor@example.com</code> )	Per-relay-user override for tenants with extra regulatory language

Most-specific match wins: a relay-recipient match is used before the domain default.

## Pipeline placement

Disclaimers are applied at SMTP receive time by the `hermes_body_milter` container, which Postfix consults as a milter alongside OpenDKIM and OpenDMARC.



Postfix :10026

(OpenDKIM signs the final composed body here)



external

Body modification happens at smtpd time, **before** content\_filter routes to Amavis. By the time Amavis sees the message, the disclaimer is already baked in. Amavis processes a normal-looking message; no internal-state coupling, no temp-file races.

OpenDKIM's outbound signing fires at the `:10026` re-injection — after both the body milter and Ciphermail. Hermes' own DKIM therefore always covers whatever the recipient ultimately receives. Ciphermail's server-side crypto also covers the disclaimer because Ciphermail runs after the milter.

## Behavior with S/MIME, PGP, and DKIM-signed mail

The behavior depends on **who** signed/encrypted the message and **when** in the pipeline.

### Server-side: signed/encrypted by Ciphermail — disclaimer is applied

Ciphermail runs **after** the body milter. Mail arrives at the milter as plaintext, the disclaimer is appended, then Ciphermail signs or encrypts the modified body. The recipient sees a valid signature **and** the disclaimer. No conflict.

### Client-side: signed/encrypted by the user's MUA — disclaimer is skipped

Mail signed in Outlook (S/MIME) or Thunderbird+Enigmail (PGP) arrives at the gateway with the cryptographic envelope already sealed. Modifying the body would either invalidate the signature or mangle the ciphertext.

The body milter detects the following patterns in the headers (or first 32 KB of the body) and exits unchanged when any matches:

Pattern matched	Meaning
-----------------	---------

Content-Type: multipart/signed; protocol="application/pkcs7-signature"	S/MIME detached signature
Content-Type: application/pkcs7-mime	S/MIME opaque-signed or enveloped
Content-Type: multipart/signed; protocol="application/pgp-signature"	PGP/MIME detached signature
Content-Type: multipart/encrypted; protocol="application/pgp-encrypted"	PGP/MIME encrypted
-----BEGIN PGP SIGNED MESSAGE----- in body	PGP inline-signed
-----BEGIN PGP MESSAGE----- in body	PGP inline-encrypted

When any of those match, the body is left untouched, the signature stays valid, the user's legal-text expectations are preserved (their MUA template is already in the body), and the gateway gets out of the way.

“ **Operational consequence.** A site whose users sign client-side will not get gateway disclaimers on those specific signed messages — by design. If org-wide legal text on **all** outbound is mandatory, the only safe pattern is server-side signing in Ciphemail with the disclaimer applied first.

## DKIM: Hermes-signed mail is fine; upstream-signed mail is skipped

OpenDKIM signs at the Postfix `:10026` re-injection step — **after** the body milter. So Hermes' own DKIM signature always covers the recipient's view of the message (with disclaimer baked in). No conflict.

The risk is mail that arrives at Hermes **already DKIM-signed by an upstream MTA** — typically a relay user whose own mail server signs before forwarding through us. Modifying that body would invalidate the upstream signature at the recipient.

The body milter treats a pre-existing `DKIM-Signature:` header the same way as a sealed S/MIME or PGP envelope and skips the disclaimer. Since Hermes' own DKIM signs at `:10026` (downstream of this milter), any DKIM-Signature header present at the milter's point in the pipeline came from somewhere upstream of Hermes.

# Reply-chain handling — no dedup, by design

The milter does **not** detect or skip messages that already carry a previous disclaimer in their quoted history. Every outbound message gets a fresh disclaimer applied — including replies inside a long thread.

This matches industry norm: commercial server-side disclaimer / signature platforms (Exclaimer, Crossware, CodeTwo, Microsoft 365 transport rules) all stamp every outbound without dedup. The reasoning:

- **Compliance.** Many regulatory regimes (HIPAA email confidentiality, GDPR data-controller notices, financial-services disclosure) treat each transmission as requiring its own disclaimer. Stamping only the first message in a thread arguably leaves later replies non-compliant.
- **Self-contained messages.** If a recipient forwards a reply (with quoted history) to a third party, the disclaimer is preserved per-message in the forwarded text.
- **Predictable behavior.** Operators don't have to explain "sometimes the disclaimer shows, sometimes it doesn't."
- **Cosmetic concern is weak.** Modern MUAs (Gmail, Outlook, Apple Mail) collapse quoted history by default, so stacked disclaimers in long threads are rarely visible to readers.

Earlier iterations of #214 included a sentinel-marker dedup mechanism (`[HD]` / `<!-- HERMES_DISCLAIMER_V1 -->`). That was removed during DEV testing in favor of the industry-norm pattern.

## Position: append vs prepend

The schema and UI both expose `position = append | prepend`, but **v1 honors append only**. Prepend is tracked as a v2 enhancement.

## Failure semantics

The body milter is **graceful-degradation** by design. Postfix's `milter_default_action = accept` means:

- Milter container down or unreachable → mail flows **unmodified** (missed disclaimer, but no delivery outage)

- Map file unreadable → no entries match → all mail flows unmodified
- Modifier raises an exception → caught and logged → mail flows unmodified
- altermime / parse errors → caught and logged → mail flows unmodified

In every failure case, mail keeps flowing. Worst case is a missed disclaimer, never lost mail. Compare the legacy "modify in amavis hook" approach (#214 Phase 3 v1, retired) which silently dropped messages when the in-place body modification desynced amavis's internal state.

## Files generated on save/delete

The CFML include `inc/disclaimer_write_and_reload.cfm` runs after every save or delete and rewrites the entire on-disk state from the `disclaimers` table:

```

/etc/hermes/body_milter/disclaimers/disclaimer_by_sender  sender → option map
/etc/hermes/body_milter/disclaimers/files/<option>/
  body.txt          plain-text disclaimer
  body.html         html disclaimer (may have  refs)
  images/
    1.png           per-disclaimer inline images (#230)
    2.jpg
    ...

```

Where `<option>` is `domain_<safe>` or `relay_<safe>` (non-alphanumeric chars in the source key are replaced with `_`).

Each disclaimer gets its own subdirectory. The files directory is wiped (per-option subdirectories deleted recursively, but the parent `files/` directory and its `.gitkeep` are preserved) and rewritten on every save. There is no incremental update — this guarantees deleted rows and renamed scope keys never leave stale files (or stale image binaries) behind.

**No reload step needed.** The body milter mtime-watches each map file on every message and reloads when it changes. The CFML `cffile` write to the map file is enough to make the change take effect on the next message processed by the milter.

## Inline images (#230)

Admins can paste or upload images directly into the Quill editor when authoring a disclaimer. Supported formats: **PNG, JPEG, GIF**. SVG and WebP are explicitly rejected (security and recipient-compatibility reasons). Limits enforced at save time:

- **5 images max** per disclaimer
- **200 KB per image** (after base64 decode)
- **1 MB total** across all images in a single disclaimer

If any limit is exceeded, the save is rejected with a specific error explaining what failed. Admins can reduce image count or size and re-save.

### How it works:

1. Quill embeds pasted/uploaded images as base64 inline `` in the HTML body. The base64 representation is what's stored in the `disclaimers.body_html` column.
2. At save time, the regenerator parses `body_html` for `data:` URLs, decodes each base64 blob, writes the binary as `<option>/images/<N>.<ext>`, and rewrites the HTML in `<option>/body.html` to use `` references.
3. At message-send time, the body milter reads `body.html`, walks `` references, and attaches each referenced image as an `image/<format>` MIME part with `Content-ID: <disclaimer_<option>_img_<N>>` and `Content-Disposition: inline`.
4. The milter wraps the message as `multipart/related` so the recipient MUA resolves cid references against the inline parts.

### MIME structure transformation (representative example):

```
Original outbound:
  multipart/alternative
    text/plain
    text/html (no images)

After milter (with disclaimer including 1 image):
  multipart/related
    multipart/alternative
      text/plain (with text disclaimer appended; images omitted from text)
      text/html (with html disclaimer + )
    image/png
      Content-ID: <disclaimer_..._img_1>
      Content-Disposition: inline
```

This structure renders inline in all major MUAs (Gmail, Outlook, Apple Mail, Thunderbird, mobile clients).

**The plain-text version of the disclaimer omits images entirely** — base64 inline images don't translate to text, and recipients viewing the message in plain-text mode see the disclaimer text without any image markers.

**Hermes' own DKIM signature covers the modified body** (including the multipart/related wrap and image parts), because OpenDKIM signs at the postfix `:10026` re-injection step — downstream of the body milter. The signature validates against what the recipient receives.

## Auto-derive of plain-text part

The Quill editor on `edit_disclaimer.cfm` drives `body_html`. By default the plain-text part shipped to recipients with a non-HTML MUA is auto-derived from the HTML on save: `<br>`, `</p>`, `</li>` become newlines, all other tags are stripped, runs of 3+ newlines collapse to 2.

Admins who need character-perfect plain text different from the auto-strip (e.g. for regulated industries) can toggle **Edit plain-text version separately** to expose a second editor. When set, `body_text` is shipped verbatim instead of derived.

## Disabled rows

Rows with `enabled = 0` are skipped entirely on regen — no files written, no map entry. The milter never matches that scope until the row is re-enabled.

## Internal-only mail

v1 does **not** suppress disclaimers for internal-only mail (sender + all recipients in `@local_domains`). Domain disclaimers will be applied to internal mail in the same domain. If this is a problem for your install, file a feature request to add an internal-only bypass.

## Why a separate milter and not an amavis hook

Earlier #214 iterations attempted to dispatch the disclaimer from inside an amavisd-new `Custom.pm` `before_send` hook, calling `altermime` via `system()` on the temp file amavis was managing. amavisd-new 2.13 caused two problems: the legacy `@disclaimer_options_bysender_maps` dispatch path was removed (variables still parse but no code reads them), and the `before_send` hook documentation says "may modify mail" but in practice in-place body modification desynchronizes amavis's internal MIME state and silently loses mail.

The body milter approach moves the body-modification step out of amavis entirely. amavis's role is unchanged from before #214 ever existed; the milter sits in postfix's smtpd\_milters chain alongside OpenDKIM and OpenDMARC, the same architectural pattern Hermes already uses for body-touching policy enforcement. amavis is fully decoupled from the disclaimer feature, which means amavis upgrades and the disclaimer feature evolve independently.

This same milter container is intended to host:

- **#226 User Signatures** (per-mailbox personal text from LDAP attributes or user-portal editor)
- **#228 External Sender Banner** (warning banner on inbound external mail)
- **Future Link Guard** (URL rewriting through a click-through endpoint)

Each is a `Modifier` subclass in `/usr/local/bin/hermes-body-milter` registered in the `MODIFIERS` list. The dispatcher is unchanged.

# External Banner

# External Banner

Maps to **Email Policies > External Banner** (`view_external_banners.cfm`, `edit_external_banner.cfm`, `external_banner_delete.cfm`). Available on both **Community** and **Pro** editions — phishing protection is a baseline security feature, not a Pro upsell.

Hermes prepends (or optionally appends) a warning banner to **inbound** mail from external senders destined for a local recipient. The banner is injected into the message body itself, so every MUA — webmail, Outlook, Apple Mail, mobile clients — renders it without relying on transport rules or recipient-side configuration. Tracked as #228.

## Scope

Scope	Recipient match	Use case
<b>System default</b>	All recipient domains (no override)	Single banner used everywhere; recommended starting point
<b>Per-recipient-domain</b>	Specific local mailbox domain (e.g. <code>legal.example.com</code> )	Different copy or compliance language for one domain

Resolution at message time, in the body milter's `ExternalBannerModifier`:

1. Look up the first **local recipient's** domain in `/etc/hermes/body_milter/banners/banner_by_recipient_domain`.
2. If a matching row exists, use it.
3. Otherwise fall back to the `_default` system-wide entry.
4. Otherwise no banner is applied.

Only the first local recipient is consulted — mixed-domain envelopes get the banner of the first local recipient encountered. This keeps the modification deterministic regardless of envelope ordering.

The `recipient_domain` field is **locked after creation**. Delete and re-create the row to change scope.

# What counts as "external"

The body milter uses Postfix's `/etc/postfix/relay_domains` file as the source of truth for "local". A message is considered **inbound from an external sender** when:

- The `MAIL FROM` sender domain is **not** in `relay_domains`, AND
- At least one `RCPT TO` recipient domain **is** in `relay_domains`.

Internal-to-internal mail (sender + all recipients local) is classified as `direction = internal` and the banner is **not** applied. There is no separate allowlist of "trusted partner" external senders today — every external sender to a local recipient triggers the banner if one is configured for that recipient's domain.

## Pipeline placement

The banner is injected at SMTP receive time by the `hermes_body_milter` container, the same container that emits outbound disclaimers ([disclaimers.md](#)) and organizational signatures ([organizational-signatures.md](#)). The milter listens on `inet:hermes_body_milter:8893` and Postfix consults it as part of `smtpd_milters`.

```
Inbound external MTA
  |
  v
Postfix smtpd
+- smtpd_milters chain (in order):
|   1. OpenDKIM           (verifies upstream DKIM signature)
|   2. OpenDMARC          (DMARC policy + ARC verification)
|   3. hermes_body_milter (THIS -- banner prepended here)
|       --> Authentication-Results header has already been written
|           by OpenDKIM/OpenDMARC BEFORE the banner touches the body
  v
content_filter --> Amavis    (sees the banner-prepended body)
  v
Ciphemail           (server-side S/MIME or PGP, if configured)
  v
Postfix :10026      (multi-instance OpenDKIM re-signs the final body)
  v
Local delivery (Dovecot LMTP)
```

Key ordering points:

- **OpenDKIM verifies first.** The upstream sender's DKIM verdict is captured in `Authentication-Results:` headers **before** the banner is injected. The header is preserved on the message; the banner does not retroactively change what OpenDKIM saw at smtpd time.
- **Amavis sees the modified body.** Spam scoring runs against the banner-prepended message. This is intentional — the banner content is short and stable and does not skew SpamAssassin scores in practice.
- **Hermes' downstream re-sign covers the modified body.** The multi-instance OpenDKIM at `:10026` (#232) signs after Ciphemail rebuild, so the final outgoing-to-Dovecot body is covered by Hermes' own signature.

# Behavior with signed and encrypted mail

The modifier inherits the same skip rules as [Disclaimers](#) for sealed envelopes:

Pattern matched	Meaning	Banner action
<code>Content-Type: multipart/signed; protocol="application/pkcs7-signature"</code>	S/MIME detached	<b>Skip</b>
<code>Content-Type: application/pkcs7-mime</code>	S/MIME opaque/enveloped	<b>Skip</b>
<code>Content-Type: multipart/signed; protocol="application/pgp-signature"</code>	PGP/MIME detached	<b>Skip</b>
<code>Content-Type: multipart/encrypted; protocol="application/pgp-encrypted"</code>	PGP/MIME encrypted	<b>Skip</b>
<code>-----BEGIN PGP SIGNED MESSAGE-----</code> in body	PGP inline-signed	<b>Skip</b>
<code>-----BEGIN PGP MESSAGE-----</code> in body	PGP inline-encrypted	<b>Skip</b>
Pre-existing <code>DKIM-Signature:</code> header on inbound mail	Upstream DKIM signed	<b>Modify anyway</b> (see below)

The corresponding flags on `ExternalBannerModifier` are `skip_on_signed = True`, `skip_on_pgp_inline = True`, `skip_on_dkim = False`.

## Why the banner does NOT skip on upstream DKIM

About 95% of inbound mail today carries a `DKIM-Signature:` header. If the banner skipped on DKIM, the feature would be effectively inert — the warning would only land on the unsigned minority that needs it least.

Hermes already records the upstream DKIM verdict in `Authentication-Results:` before modifying the body. Recipients overwhelmingly read mail through Dovecot/IMAP and the recipient MUA does not re-verify upstream DKIM. The banner is therefore safe in the common case.

The narrower edge case — a recipient who forwards Hermes-banner'd mail to a downstream MX that **does** re-verify upstream DKIM — is addressed by [ARC sealing](#) (#229). Hermes' ARC seal at `:10026` records `cv=fail` for the upstream chain (because we modified the body), but the seal itself is mathematically valid and the downstream MX can trust Hermes' ARC verdict if Hermes is on its allowlist. See [ARC Settings](#) for the full discussion of the `cv=fail`-by-design pattern.

“ **Operational consequence.** Banner injection breaks the original sender's DKIM body hash and any upstream ARC body hash. **This is by design.** Hermes is the authoritative auth boundary for the domains it relays; customer downstream MX servers must allowlist Hermes and accept its delivered mail without re-running DKIM/SPF/DMARC/ARC. A downstream MX that re-verifies upstream auth on mail Hermes forwards is misconfigured — cross-ref [ARC Settings](#), [DKIM Settings](#), and [DMARC Settings](#).

## Position: prepend vs append

Position	Behavior	Recommendation
<b>Top</b> ( <code>prepend</code> )	Banner becomes the first child of the message body (above any quoted history)	<b>Industry standard</b> — users see the warning before reading any content
<b>Bottom</b> ( <code>append</code> )	Banner is appended after the user-visible body	Available for sites that prefer it; rarely used

Both positions are implemented end to end (unlike Disclaimers, where only `append` is honored in v1). HTML prepend is done with BeautifulSoup: the banner fragment is inserted as the first child of `<body>` when present, otherwise prepended to the root.

## Templates

Banners use a **server-side template gallery**, not a free-form WYSIWYG editor. Quill 2.x's HTML normalization strips inline styles that Gmail and Outlook need (the same problem hit on Organizational Signatures #226 Phase 2 and on this feature), so admins pick a template and fill in form fields; the server renders pixel-perfect HTML at save time.

Bundled templates (each `inc/external_banner_templates/<key>.cfm`):

Template key	Display name	When to pick it
<code>warning_yellow</code>	<b>Warning Yellow</b>	Default. Yellow background with orange accent. Matches Microsoft 365 / Mimecast banner style most users recognize
<code>critical_red</code>	<b>Critical Red</b>	Red background, white text. Phishing-prone industries or post-incident periods where alert level needs to be raised
<code>subtle_info</code>	<b>Subtle Info</b>	Light gray with blue accent. Less alarming for high-volume inbound (support/sales) where alert fatigue is a concern
<code>plain_text</code>	<b>Plain Text</b>	Bold prefix + text, no background or border. Maximum cross-MUA compatibility, including text-only clients

All four templates expose the same field set:

Field	Type	Default	Notes
<code>prefix</code>	text	<code>[EXTERNAL]</code>	Short tag rendered bold at the start. Plain ASCII recommended for Outlook
<code>headline</code>	text	"This message originated from outside your organization."	First line, regular weight
<code>body</code>	text	"Do not click links or open attachments unless you recognize the sender..."	Second line, smaller text
<code>show_learn_more</code>	checkbox	<code>false</code>	Reveals the next two fields
<code>learn_more_url</code>	url	<i>empty</i>	Optional link to internal phishing-awareness training or wiki
<code>learn_more_label</code>	text	"Learn more about phishing"	Visible label for the learn-more link

All templates emit **table-based HTML with `bgcolor=` attributes** so Outlook (which strips inline CSS but honors deprecated HTML attributes) renders the banner correctly. Inline styles are belt-

and-suspenders for Gmail, Apple Mail, and mobile clients.

The edit page renders a live preview in an iframe via `inc/render_external_banner_preview.cfm` so the admin sees exactly what `save_external_banner_action.cfm` will store.

## Files generated on save/delete

`inc/external_banner_write_and_reload.cfm` runs after every save or delete and rewrites the entire on-disk state from the `external_banners` table:

```
/etc/hermes/body_milter/banners/banner_by_recipient_domain
  <recipient_domain>\t<option>
  _default\t<option>          special key, system-wide fallback

/etc/hermes/body_milter/banners/files/<option>/
  body.txt          plain-text banner (auto-derived at save)
  body.html         pre-rendered html banner
  position          "prepend" or "append" sidecar file
  images/          per-banner inline images (#230 cid pattern)
    1.png
    2.jpg
    ...
```

Where `<option>` is:

- `banner_default` for the system-wide row (NULL `recipient_domain`)
- `banner_<sanitized_recipient_domain>` for per-domain overrides (non-alphanumeric characters replaced with `_`)

The `files/` subdirectory is wiped on every regen (per-banner subdirs deleted recursively; the `.gitkeep` is preserved). This guarantees deleted rows and renamed scopes never leave stale files behind.

**No reload step needed.** The body milter mtime-stats each map file on every message and reloads automatically when its mtime changes. The CFML `cffile` write to the map file is enough to make the change take effect on the next message.

## Plain-text part

The HTML body stored in `external_banners.body_html` is rendered server-side from the chosen template. The plain-text counterpart in `body_text` is auto-derived at save time:

- `<br>` becomes a newline
- `</p>`, `</li>`, `</tr>`, `</td>`, `</div>` become newlines
- All remaining tags are stripped
- Runs of 3+ newlines collapse to 2

The plain-text version is shipped to recipients viewing the message as `text/plain`. Inline images are omitted from the plain-text part — data URLs don't translate to text and recipients in text mode see the banner copy without image markers.

## Inline images (#230)

The banner modifier inherits the `#230` cid inline-image pattern from Disclaimers. If a template's HTML contains `` references, the body milter:

1. Loads matching `images/<N>.<ext>` files from the option directory.
2. Attaches each as an `image/<format>` MIME part with `Content-ID: <banner_..._img_<N>` and `Content-Disposition: inline`.
3. Wraps the message as `multipart/related` so MUAs resolve cid references against the inline parts.

The cid prefix is `banner_` so banner images cannot collide with `disclaimer_` or `signature_` cids inside the same composed message (the three modifiers can all add images to the same outbound; namespacing keeps them separate).

The bundled templates do not currently use inline images — banners are pure text. The infrastructure is present for future template additions (logo, warning icon, etc.).

## Failure semantics

The body milter is **graceful-degradation** by design. Postfix's `milter_default_action = accept` means:

- Milter container down or unreachable -> mail flows **unmodified** (missed banner, no delivery outage)
- Map file unreadable -> no entries match -> all mail flows unmodified
- Per-option files missing -> log + skip the modify -> mail flows unmodified
- MIME parse exception -> caught and logged -> mail flows unmodified
- Modifier raises any other exception -> caught and logged -> mail flows unmodified

In every failure case, mail keeps flowing. Worst case is a missed banner, never lost mail. Compare the legacy "modify in amavis hook" approach (#214 Phase 3 v1, retired) which silently dropped messages when the in-place body modification desynced amavis's internal state.

## Disabled rows

Rows with `enabled = 0` are skipped entirely during regen — no files written, no map entry. The milter never matches that scope until the row is re-enabled. Useful for staging copy changes before going live (build the new row disabled, preview it on `edit_external_banner.cfm`, flip the switch when ready).

## Schema

```
CREATE TABLE IF NOT EXISTS external_banners (  
  id                int(11)          NOT NULL AUTO_INCREMENT,  
  recipient_domain varchar(255)      DEFAULT NULL,           -- NULL = system default  
  template_key     varchar(64)      NOT NULL DEFAULT 'warning_yellow',  
  fields_json      longtext         DEFAULT NULL,           -- form values for re-edit  
  body_text       longtext         DEFAULT NULL,           -- auto-derived plain text  
  body_html       longtext         NOT NULL,                 -- pre-rendered html  
  position        enum('prepend','append') NOT NULL DEFAULT 'prepend',  
  enabled         tinyint(3)       NOT NULL DEFAULT 1,  
  updated_at     timestamp        NULL DEFAULT current_timestamp() ON UPDATE  
current_timestamp(),  
  PRIMARY KEY (id),  
  UNIQUE KEY uk_recipient_domain (recipient_domain)  
);
```

The `UNIQUE KEY` on `recipient_domain` ensures only one row per recipient domain (and at most one system-default row where `recipient_domain IS NULL`). The `fields_json` blob stores the original form values so reopening the editor restores exactly what the admin typed; `body_html` is the rendered output the milter actually ships.

## Verifying it works

The banner appears in the message body, so the easiest verification is to send an inbound message from an external account to a local mailbox and view the result in any MUA (webmail,

Outlook, Apple Mail). Beyond that:

- **Body milter logs** — the modifier logs `external_banner applied: option=<name> position=<prepend|append> plain=<n> html=<n>` per modified message. Surface with `docker logs hermes_body_milter` or via [System Logs](#).
- **Authentication-Results: header** is preserved from upstream and visible in the recipient's "view source"; this confirms OpenDKIM ran **before** the banner.
- **ARC-Seal: ... cv=fail** in the outgoing message confirms the body was modified after the upstream chain — expected behavior, cross-ref [ARC Settings](#).

## Related

- [Disclaimers](#) — the outbound counterpart; same `hermes_body_milter` container, parallel design (sender-keyed instead of recipient-keyed)
- [Organizational Signatures](#) — second outbound modifier in the same container, with per-recipient resolution
- [ARC Settings](#) — full explanation of `cv=fail` after body modification and the Hermes-as-auth-boundary model
- [DKIM Settings](#), [DMARC Settings](#) — upstream-verdict context preserved in `Authentication-Results`
- [Domains](#) — local mailbox-hosting domains drive the per-domain dropdown on `edit_external_banner.cfm`
- [System Logs](#) — surface the body-milter log stream for troubleshooting

# Organizational Signatures

# Organizational Signatures

*Pro Edition feature.* Maps to **Email Policies > Org Signatures** (`view_org_signatures.cfm`, `edit_org_signature.cfm`, `org_signature_delete.cfm`).

Hermes attaches a centrally-managed signature to outbound mail at the gateway. Admins design the signature once per domain (and optionally per department); every user on that domain gets a personalized version of it on every outbound message — no per-user setup required.

## Two signature types, one pipeline

Hermes ships two distinct signature concepts that run through the same body milter and the same resolver:

Type	Tier	Owner	Storage	Per-domain control
<b>Personal Signature</b>	Community + Pro	The user (in <code>/users/2/view_signature.cfm</code> )	<code>user_signatures</code> table, one row per user	Toggled via <code>domains.allow_user_signatures</code>
<b>Organizational Signature</b>	Pro only	The admin (in <code>Email Policies &gt; Org Signatures</code> )	<code>org_signatures</code> table, one row per <code>(domain_id, department_label)</code>	One default per domain + optional per-department variants

The milter never decides which one to apply at message time. The CFML resolver picks a winner per mailbox at admin-action time and writes a precomputed `sender → option` map; the milter just looks up the option and applies whatever it finds.

## Department names — single source of truth

Departments are defined once on the **mailbox edit form** (Email Server > Mailboxes > Edit Options > Personal Information > Department), as free-text values typed by the admin. There is no

separate "Departments" table; a department exists as soon as one mailbox is in it.

The Org Sig form's Department field is a **strict dropdown sourced from the distinct `mailboxes.department` values for the selected domain**. This means:

- You cannot create an Org Sig for a dept that has no mailboxes — the dept won't appear in the dropdown.
- The dept name on both sides is guaranteed to match exactly. No typo-class drift.
- Workflow: assign at least one mailbox to the new dept first, then come back and create the Org Sig targeting it.
- Changing the domain in the Org Sig form repopulates the dropdown with that domain's depts via JavaScript (no AJAX round-trip; the per-domain map is dumped into a JS const at page load).

The mailbox edit form's Department field is a free-text input with a `<datalist>` **typeahead** showing the same per-domain dept list. Admins can pick an existing dept (auto-completes) or type a brand-new dept name (which then appears in the dropdown next time).

If you edit an existing Org Sig whose `department_label` no longer matches any current mailbox (the dept was renamed elsewhere, or all mailboxes in it were reassigned), the orphan value is preserved in the dropdown with a `(no mailboxes)` suffix so you can still see and edit/delete the row instead of silently losing the value.

The resolver at send time does a **case-insensitive trimmed match** against `mailboxes.department`, so casing or whitespace drift across edits is forgiving even in the rare cases the dropdown is bypassed (e.g. direct SQL changes).

## Resolution order

For every enabled mailbox, the resolver walks this priority chain top-down and stops at the first match:

```
1. Personal Signature
  └ if domains.allow_user_signatures = 1
    AND user_signatures has an enabled, non-empty row for this mailbox
  -> wins. option = user_<sanitized_email>

2. Department Organizational Signature
  └ else if mailboxes.department is non-empty
    AND org_signatures has enabled = 1 row matching
      (domain_id, department_label)
  -> wins. option = org_<row_id>
```

### 3. Domain Default Organizational Signature

```
└ else if org_signatures has enabled = 1 row matching
    (domain_id, department_label IS NULL)
-> wins. option = org_<row_id>
```

### 4. None

```
└ no map entry; the milter applies no signature to this sender's mail.
```

The chain is **per-mailbox**, not per-message. The resolver runs at admin-action time (see [Triggers](#)), serializes the winning option for every mailbox into one map file, and the milter consults that map at send time. There is no per-message DB query and no per-message resolution logic in the milter.

## Pipeline placement

Same chain as Disclaimers (#214) — see [disclaimers.md](#) for the full Postfix / OpenDKIM / Ciphermail picture. The summarized order:

```
External MTA / MUA submission
```

```
|
```

```
▼
```

```
Postfix smtpd
```

```
└ smtpd_milters chain (in order):
```

```
| 1. OpenDKIM (signs/verifies)
```

```
| 2. OpenDMARC (DMARC policy)
```

```
| 3. hermes_body_milter (THIS – signatures, then disclaimers)
```

```
▼
```

```
Amavis → Ciphermail → Postfix :10026 (DKIM sign) → external
```

Inside the body milter, **SignatureModifier runs before DisclaimerModifier**, so the layered output on the outbound message is:

```
[user body]
```

```
[signature] ← Personal or Organizational, picked by resolver
```

```
[disclaimer] ← if a disclaimer is configured for this sender
```

OpenDKIM signs at the `:10026` re-injection — after both modifiers — so Hermes' own DKIM signature always covers the recipient's view of the message (with signature and disclaimer baked in).

# Templates

Phase 2A ships **six bundled templates** under `/admin/2/inc/org_signature_templates/`:

Template key	Layout
<code>modern_card</code>	Logo left, accent bar, contact stack right
<code>two_column_pro</code>	Left contact, right org block + CTA button
<code>with_social_bar</code>	Vertical contact + horizontal social-icon row
<code>banner_with_logo</code>	Full-width banner with logo top, contact below
<code>promo_footer</code>	Contact + bottom promotional image with link
<code>compact_text</code>	Minimal text-only, no images, no styling

Each template is a `.cfm` file that declares its **field schema** (text / email / url / color / checkbox / image fields with optional `showIf` gating) and renders pixel-perfect HTML when the renderer is invoked. Admins fill in the form on `edit_org_signature.cfm`; the gallery thumbnail + live sandboxed-iframe preview show the result before save.

All the auto-filled fields — Name, Title, Phone, Mobile, Email (`{{user.*}}` from the mailbox row) plus Website and Address (`{{org.*}}` from the domain row) — are **collapsed under an "Override auto-filled fields" toggle** by default. The admin doesn't see or edit them in the common case; the placeholders flow through to the rendered HTML unchanged and the milter fills in the recipient's data at send time. Toggling the override on exposes the fields for the rare cases that need literal text instead of substitution (shared mailboxes without personal info, seasonal URL overrides, etc.).

The genuinely admin-supplied fields stay always visible — Logo, accent color, show/hide toggles for each line, CTA text and URL, social URLs, disclaimer text, and any template-specific extras (banner height, promo image, etc.). These are the admin's actual editing surface.

Net workflow: pick a template, upload a logo, set the accent color, save. Done. Every mailbox on the domain gets a fully personalized signature on its next outbound message without any per-user form input.

Templates are version-controlled in the repo, not in the database. To add a new template, drop a new `.cfm` file in `org_signature_templates/`, add its key to the registry in `inc/org_signature_template_loader.cfm`, and produce a 240×140 thumbnail PNG. No schema migration needed.

# Placeholder substitution at send time

The signature HTML stored in `org_signatures.rendered_html` (and on disk in `body.html`) keeps `{{namespace.field}}` tokens **literal**. The body milter substitutes them per recipient at message-send time against the sender's row in `sender_data.json`.

Available placeholders (Phase 2B):

Token	Source column
<code>{{user.first_name}}</code>	<code>mailboxes.first_name</code>
<code>{{user.last_name}}</code>	<code>mailboxes.last_name</code>
<code>{{user.title}}</code>	<code>mailboxes.title</code>
<code>{{user.phone}}</code>	<code>mailboxes.phone</code>
<code>{{user.mobile}}</code>	<code>mailboxes.mobile</code>
<code>{{user.department}}</code>	<code>mailboxes.department</code>
<code>{{user.email}}</code>	<code>mailboxes.username</code>
<code>{{org.name}}</code>	<code>domains.org_name</code>
<code>{{org.phone}}</code>	<code>domains.org_phone</code>
<code>{{org.address}}</code>	<code>domains.org_address</code>
<code>{{org.website}}</code>	<code>domains.org_website</code>
<code>{{org.logo_url}}</code>	<code>domains.org_logo_path</code> (raw URL — not cid: extracted)
<code>{{dept.name}}</code>	<code>mailboxes.department</code>

Tokens whose corresponding field is empty resolve to **empty string**, not literal `{{...}}`. So if `mailboxes.title` is blank for a given user, the `{{user.title}}` token disappears cleanly from delivered mail. Unknown namespaces (anything outside `user`, `org`, `dept`) are also substituted to empty.

The substitution is a single regex pass on the body html and body text inside `SignatureModifier.modify()` — well under a millisecond per message. The map and `sender_data.json` both live in process memory, refreshed only when their file mtime changes.

**No DB connection from the milter, ever.** All resolution and substitution data is precomputed by CFML and dropped on disk; the milter consumes the file artifacts.

# Triggers — when the resolver re-runs

Both `signature_by_sender` and `sender_data.json` are rewritten in full by `inc/signature_regen_map.cfm` on every event that could affect a winner or a substitution value:

Event	Why it matters
Admin saves an Org Sig	New / edited row may win for senders that previously had no match or a lower-tier match
Admin deletes an Org Sig	Losers fall back to the next tier (or none)
Admin edits a domain ( <code>allow_user_signatures</code> or <code>org_*</code> columns)	Per-domain toggle flips the Personal-vs-Org winner; <code>org_*</code> values feed <code>{{org.*}}</code> substitution
Admin edits a mailbox (Pro fields: <code>first_name</code> , <code>title</code> , <code>dept</code> , etc.)	<code>{{user.*}}</code> and <code>{{dept.name}}</code> substitution data changes; a department change can flip Default → Department winner
Admin adds a mailbox	New sender enters resolution and may pick up a domain-default Org Sig
Admin deletes a mailbox	Sender must drop from the map
User saves their Personal Signature	May now win over the previously-resolved Org Sig (or vice versa if disabling)

Each trigger runs the same shared resolver. **Full rebuild every time, not incremental.** With low-thousands of mailboxes the rebuild is well under a second, and the simplicity rules out drift bugs ("did we forget to update X for sender Y" can't happen).

The body milter mtime-watches both files and reloads in process memory on the next message after the file changes. **No SIGHUP, no IPC, no container restart.**

## Files generated on save

The CFML resolver writes:

```
/etc/hermes/body_milter/signatures/signature_by_sender  sender → option map
/etc/hermes/body_milter/signatures/sender_data.json    sender → {{token}} dict
/etc/hermes/body_milter/signatures/files/<option>/
  body.txt      plain-text signature (auto-derived from html)
  body.html    html signature with cid: refs (placeholders intact)
```

```
images/
  1.png      per-option inline images (#230 pattern)
  2.jpg
  ...
```

Where `<option>` is `user_<sanitized_email>` (Personal Sig) or `org_<row_id>` (Org Sig).  
`<sanitized_email>` is `bob.smith@example.com` → `bob_smith_at_example_com` (@ → `_at_`, non-alphanumerics → `_`).

`signature_by_sender` example:

```
alice@example.com  org_42
bob@example.com    user_bob_at_example_com
carol@example.com  org_43
```

`sender_data.json` example (post-Lucee uppercasing — the milter normalizes to lowercase on load):

```
{
  "alice@example.com": {
    "USER.FIRST_NAME": "Alice",
    "USER.TITLE": "Sales Manager",
    "ORG.NAME": "Acme",
    "ORG.PHONE": "555-0100",
    "DEPT.NAME": "Sales"
  }
}
```

The `files/<option>/` dir is wiped before re-render on every save of that row, so deleted images and renamed scope keys never leave stale binaries behind.

## Inline images (cid: extraction)

Same pattern as Disclaimers (#230) — see [disclaimers.md](#) for the MIME / multipart-related details.

For Org Signatures, the cid: namespace is `signature_org_<row_id>_img_<N>` (Personal Signatures use `signature_user_<sanitized_email>_img_<N>`). Both share the milter regex `cid:(signature_[\w.-]+_img_\d+)`, so cid: refs from either signature type are queued for inline attachment alongside any cid: refs from a domain disclaimer on the same message — no namespace collisions.

Per-template image fields use the **same data: → cid: pipeline** as user-pasted Personal Sig images. At admin-save time:

1. Admin uploads the file in the form (or pastes a URL — both are handled).
2. Browser converts the file to a `data:image/...;base64,...` URI via `FileReader`, capped at 1 MB per image.
3. CFML renders the template; the resulting HTML carries the data: URI inline.
4. `inc/org_signature_write_files.cfm` extracts each `data:` URI, decodes the base64 into a binary file under `images/`, and rewrites the html to reference ``.
5. At message-send time the militer walks the cid: refs, attaches each image as an `image/<format>` MIME part with `Content-ID` and `Content-Disposition: inline`, and wraps the message as `multipart/related`.

`{{org.logo_url}}` is **not** cid: extracted — it's a raw URL substituted into the html as-is. Use it for hosted-elsewhere logos (your CDN, your website). Use the per-template **image** field for cid:-attached inline logos when you want them to render even in MUAs that block external images.

## Behavior with S/MIME, PGP, DKIM-signed mail

Identical to Disclaimers — same skip rules in the same Modifier base class. Pre-signed envelopes, PGP inline, and pre-existing DKIM-Signature headers all cause the body militer to leave the message untouched.

See [disclaimers.md "Behavior with S/MIME, PGP, and DKIM-signed mail"](#) for the table of patterns and the operational consequences.

## Reply-chain handling

No dedup — every outbound gets a fresh signature, including replies inside a long thread. Same industry-norm pattern Disclaimers uses; same rationale (compliance, self-contained messages, predictability). See [disclaimers.md "Reply-chain handling"](#).

## Failure semantics

Same graceful-degradation contract as Disclaimers (`milter_default_action = accept`). If the militer container is down, if the map file is unreadable, if the modifier raises an exception, if substitution blows up — **mail flows unmodified**. Worst case is a missed signature; mail never gets dropped.

See [disclaimers.md](#) "Failure semantics".

# Disabled rows

`org_signatures.enabled = 0` causes the resolver to skip the row entirely:

- The on-disk `org_<id>/` dir is wiped clean
- No mailboxes resolve to that option
- Mailboxes that previously resolved to that option fall back to the next tier (department → default → none)

Re-enabling rebuilds the on-disk files and re-points the affected mailboxes' map entries on the next regen.

# Interaction with

## `domains.allow_user_signatures`

This per-domain toggle is the single switch that controls whether Personal Signatures can win over Organizational Signatures.

<code>allow_user_signatures</code>	Personal Sig present?	Result
0	yes	Personal Sig <b>ignored</b> ; resolver falls to Department / Default Org Sig
0	no	Resolver falls to Department / Default Org Sig
1	yes	Personal Sig <b>wins</b> (top of resolution chain)
1	no	Resolver falls to Department / Default Org Sig

Toggle this off when you need to **lock everyone into Organizational Signatures** for branding/compliance — useful when a marketing or legal team wants centrally-controlled output and doesn't want individual users overriding it.

A previously-saved Personal Signature is **not deleted** when the toggle goes off — it just stops being resolved. Toggling back on re-activates it on the next regen.

# Pro license behavior

The Org Signatures admin page is gated by `session.edition EQ "Pro"`:

- **Pro license valid:** Full UI access; admins can create / edit / delete / enable / disable Org Signatures and toggle `allow_user_signatures`.
- **Pro license missing or expired:** Sidebar entry shows a `PRO` badge; the list and edit pages reject the load with an upsell flash. The save action handler ALSO rejects on the server side (defense in depth — UI-level gating alone isn't a security control).
- **Existing Org Signatures on a downgrade:** Rows persist in the database. The resolver still runs and the milter continues applying them at send time. Personal Signatures continue working as well. The downgrade is a **UI restriction**, not a runtime feature kill.

This is the same "feature stays on, admin UI locks" pattern as Disclaimers and other Pro features. If a customer wants the Pro feature actually disabled at runtime on downgrade, the path is to delete the rows or set them all to `enabled = 0`.

## Why a separate milter and not an amavis hook

Same reasoning as Disclaimers (#214 Phase 3). amavisd-new 2.13's `before_send` hook silently desynchronizes amavis's internal MIME state during in-place body modification, which can drop mail. The body milter approach moves body modification out of amavis entirely; amavis is fully decoupled.

See [disclaimers.md "Why a separate milter and not an amavis hook"](#).