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
Personal Signature	Community + Pro	The user (in `/users/2/view_signature.cfm`)	`user_signatures` table, one row per user	Toggled via `domains.allow_user_signatures`
Organizational Signature	Pro only	The admin (in `Email Policies > Org Signatures`)	`org_signatures` table, one row per `(domain_id, department_label)`	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
`modern_card`	Logo left, accent bar, contact stack right
`two_column_pro`	Left contact, right org block + CTA button
`with_social_bar`	Vertical contact + horizontal social-icon row
`banner_with_logo`	Full-width banner with logo top, contact below
`promo_footer`	Contact + bottom promotional image with link
`compact_text`	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
`{{user.first_name}}`	`mailboxes.first_name`
`{{user.last_name}}`	`mailboxes.last_name`
`{{user.title}}`	`mailboxes.title`
`{{user.phone}}`	`mailboxes.phone`
`{{user.mobile}}`	`mailboxes.mobile`
`{{user.department}}`	`mailboxes.department`
`{{user.email}}`	`mailboxes.username`
`{{org.name}}`	`domains.org_name`
`{{org.phone}}`	`domains.org_phone`
`{{org.address}}`	`domains.org_address`
`{{org.website}}`	`domains.org_website`
`{{org.logo_url}}`	`domains.org_logo_path` (raw URL — not cid: extracted)
`{{dept.name}}`	`mailboxes.department`

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 (`allow_user_signatures` or `org_*` columns)	Per-domain toggle flips the Personal-vs-Org winner; org_* values feed `{{org.*}}` substitution
Admin edits a mailbox (Pro fields: first_name, title, dept, etc.)	`{{user.*}}` and `{{dept.name}}` 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 `...` 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 `<img src="cid:signature_org_<id>_img_<N>">`.
5. At message-send time the milter 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 milter 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 milter 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.

`allow_user_signatures`	Personal Sig present?	Result
`0`	yes	Personal Sig **ignored**; resolver falls to Department / Default Org Sig
`0`	no	Resolver falls to Department / Default Org Sig
`1`	yes	Personal Sig **wins** (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".

Revision #14
Created 2026-05-31 12:52:42 UTC by Dino Edwards
Updated 2026-06-13 12:30:28 UTC by Dino Edwards