Email Server

- Aliases
- Domains
- Mailbox Rules
- Mailboxes
- SAN Management
- Settings
- Shared Mailboxes

Aliases

Aliases

Admin path: **Email Server > Aliases** (`view_mailbox_aliases.cfm`, `inc/add_mailbox_alias_action.cfm`, `inc/edit_mailbox_alias_action.cfm`, `inc/delete_mailbox_alias_action.cfm`, `inc/get_mailbox_alias_json.cfm`).

This page manages **alternate email addresses for local mailboxes** on the Email Server topology. Each row in the `mailbox_aliases` table maps one inbound address (e.g., `sales@company.com`) to either an existing local mailbox or to Postfix's discard transport for silent disposal. The destination must be local — to an existing Dovecot mailbox on this server. For forwarding to external addresses or for relay-topology domains, use Email Relay > Virtual Recipients instead.

Aliases have **no SMTP authentication, no IMAP/POP3 access, and no password of their own**. They are rewrite rules consumed by Postfix before content filtering. The optional **Send-As** flag adds a row to `sender_login_maps` so the destination mailbox owner can send mail under the alias address from their existing IMAP/Submission session.

Not the same as Virtual Recipients

Email Server aliases and Email Relay virtual recipients share the same underlying Postfix lookup but enforce different topology rules. See Virtual Recipients for the full distinction; the short version:

	Mailbox Aliases (this page)	Virtual Recipients
Table	`mailbox_aliases`	`virtual_recipients`
Domain type	Mailbox domains (`domains.type = 'mailbox'`)	Relay domains (`domains.type = 'relay'` or NULL)
Delivery target	A local Dovecot mailbox, or `discard:silently`	Anywhere — internal or external
UNIQUE on address	Yes (one delivery per alias)	No (fan-out via multiple rows)
Send-As	Optional, surfaced as a toggle	Schema flag, not yet wired through
Catch-all (`@domain`)	Not supported	Supported

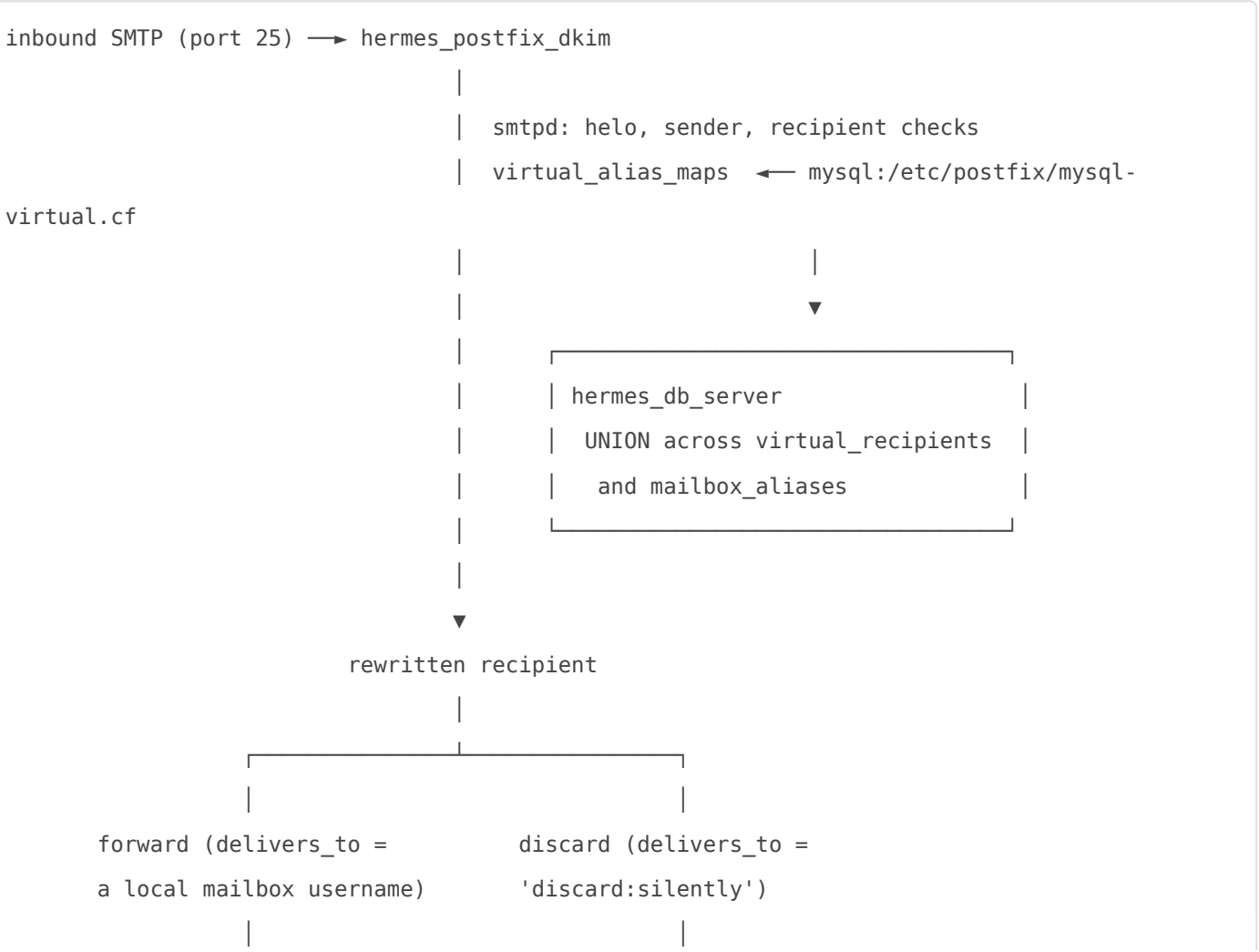
	Mailbox Aliases (this page)	Virtual Recipients
Discard transport	Supported (silent drop)	Not supported
Typical use	`support@company.com → tina@company.com` (both local)	`info@company.com → admin@externalpartner.example`

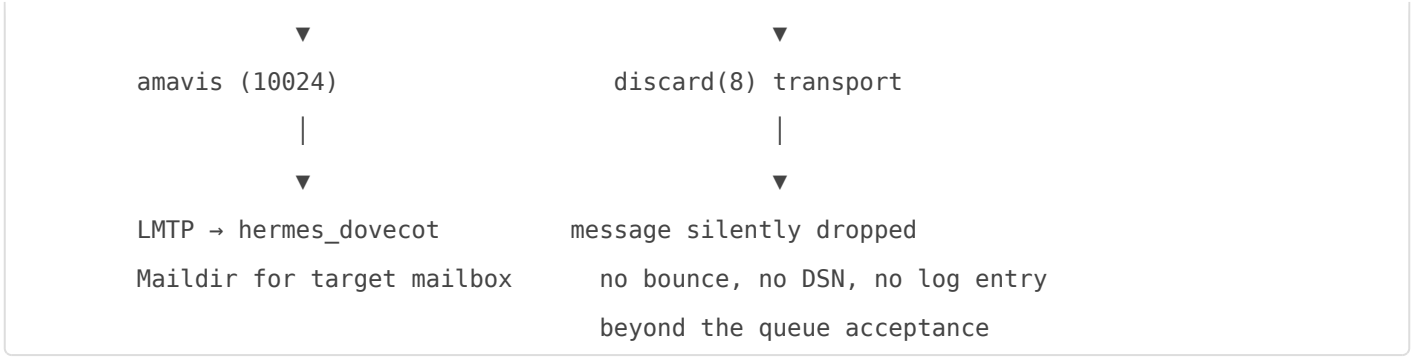
Both tables feed the same `virtual_alias_maps` lookup via a single UNION query in `mysql-virtual.cf`:

```
SELECT maps        FROM virtual_recipients WHERE virtual_address = '%s'
UNION
SELECT delivers_to FROM mailbox_aliases    WHERE alias_address   = '%s'
```

The add handlers in each page enforce the topology gate: trying to create a mailbox alias for a relay domain is rejected with error 12, and the Virtual Recipients add handler rejects mailbox-domain rows with a pointer back to this page.

Storage and lookup path





The MySQL lookup is live — adding a row in this page takes effect on the next inbound message, with no Postfix reload, no `postmap`, and no template regeneration.

The `mailbox_aliases` table

Column	Type	Role
`id`	INT PK	Surrogate key
`alias_address`	VARCHAR(255), **UNIQUE**	The address being rewritten. Full email only — no catch-all syntax. The UNIQUE constraint enforces one delivery target per alias address.
`delivers_to`	VARCHAR(255)	Destination. For `alias_type = 'forward'` this is the local mailbox username; for `alias_type = 'discard'` this is hardcoded to the literal string `discard:silently`, which Postfix routes through the discard(8) transport.
`alias_type`	VARCHAR(20)	`forward` (default) or `discard`
`send_as`	TINYINT(3)	`1` if the destination mailbox is allowed to send mail as the alias address. Wired into `sender_login_maps` on insert/update.
`domain_id`	INT	FK to `domains.id`; set on insert from the parsed domain part of `alias_address`. Used to filter the page by domain and to enforce the mailbox-topology gate.
`created_at`	DATETIME	Audit timestamp

The UNIQUE key on `alias_address` is the reason fan-out isn't supported here — one inbound address resolves to exactly one destination. To deliver one inbound address to several mailboxes, use a shared mailbox (which gives multiple users access to a single inbox) or, for true fan-out, use the relay topology with virtual recipients.

The two alias types

Forward

Delivers mail to an existing local mailbox. The mailbox must exist in the `mailboxes` table — the add handler verifies this with `error 16` on failure. The `Delivers To` dropdown is sourced from the live mailbox list (`mailbox_type = 'user'`), so you can only pick a real target.

```
sales@company.com    →   tina@company.com
support@company.com  →   helpdesk@company.com
```

Both addresses must be on a mailbox domain that this server hosts. Cross-domain forwards are allowed as long as both sides are local mailbox domains.

Discard

Silently drops all mail with no bounce, no DSN, and no error returned to the sender. The handler hardcodes `delivers_to = 'discard:silently'`, which Postfix interprets as the discard(8) transport with the literal nexthop `silently`. Useful for addresses like `noreply@` or `donotreply@` where bounces would invite spam-mining attempts.

```
noreply@company.com      →   discarded
donotreply@company.com   →   discarded
unsubscribe@company.com  →   discarded
```

> **Operational consequence.** Discard is irrecoverable — there is no queue entry, no quarantine, no recovery. The message is accepted by Postfix and immediately dropped. Use discard for addresses that should never receive replies; do not use it as a quiet alternative to bouncing mail you actually want to reject (use Postfix recipient restrictions for that).

Fields on the page

Add Alias modal

Field	Notes
Alias Address	Full email. Must validate as an email, must be on a mailbox domain (`domains.type = 'mailbox'`), and must not already exist as a mailbox, an alias, or a virtual recipient. Conflicts produce errors 12 / 13 / 14 / 17 respectively.
Type	`Forward (deliver to mailbox)` (default) or `Discard (silently drop all mail)`. JS toggles the Delivers To and Send-As fields based on selection.
Delivers To	Tom Select typeahead populated from `mailboxes WHERE mailbox_type = 'user'`. Required for forward type, ignored for discard. The handler verifies the target mailbox exists at submit time.
Allow Send-As	`No` (default) or `Yes`. Only applies to forward type. When `Yes`, an `INSERT IGNORE` into `sender_login_maps` allows the destination mailbox owner to send under the alias address from their existing Submission session.

Aliases table

DataTables surface — searchable, sortable, paginated, `stateSave: true`. Columns:

Column	Source
Actions	Edit (opens modal) / Delete (opens confirmation modal)
Alias	`mailbox_aliases.alias_address`
Domain	`domains.domain` (joined via `domain_id`)
Type	Badge — `Forward` (blue) or `Discard` (dark)
Delivers To	`mailbox_aliases.delivers_to` for forwards; `Silently dropped` for discards
Send-As	Badge — `YES` / `NO` for forwards; em-dash for discards

A Domain filter dropdown above the table narrows the visible rows to a single mailbox domain. The dropdown only lists domains that currently have at least one alias.

Edit modal

Address is read-only after creation — changing the local-part would break any send-as mappings that already reference it. Type, Delivers To, and Send-As are all editable, with the same forward/discard toggle behavior as the Add modal. The handler diffs the old send-as state against the new one and adds or removes the `sender_login_maps` row accordingly so the change to send-as is reflected without rewriting unrelated maps.

Delete

Per-row delete with a confirmation modal. The handler removes the alias row and any `sender_login_maps` entries for the alias address. Because aliases don't own a Maildir or any on-disk state, deletion is instant and reversible only by re-creating the alias.

Send-As — what it actually does

When Send-As is enabled on a forward alias, the handler inserts:

```
INSERT IGNORE INTO sender_login_maps (sender, login_user)
VALUES ('sales@company.com', 'tina@company.com');
```

That row participates in Postfix's `smtpd_sender_login_maps` lookup on the submission port. The effect: when `tina@company.com` authenticates to Submission (587) and tries to send a message with `From: sales@company.com`, Postfix accepts the From: because the `(sender, login_user)` pair exists in the map. Without Send-As, Postfix's `reject_sender_login_mismatch` would reject the submission because `tina@` is not the canonical owner of `sales@`.

This makes Send-As a true alternate-identity grant, not just a "vanity From:". The user typically configures the alias as a secondary identity in their mail client (Outlook → Account Settings → multiple email addresses; Apple Mail → Edit Email Addresses; Thunderbird → Manage Identities) and picks it from the From: dropdown when composing.

The deletion handler removes the matching `sender_login_maps` row when the alias is deleted; the edit handler removes the old row and inserts the new one when Send-As is toggled or Delivers To changes.

Conflict checks at insert time

The add handler runs four duplicate checks before the INSERT:

Check	Error	What it prevents
`mailboxes WHERE username = alias_address`	13	Alias collides with an actual mailbox. The mailbox itself would always win the lookup, so the alias would be dead weight.
`mailbox_aliases WHERE alias_address = alias_address`	14	Duplicate alias row (also enforced by the UNIQUE key, but caught earlier with a friendlier message).

Check	Error	What it prevents
`virtual_recipients WHERE virtual_address = alias_address`	17	Alias collides with a relay-topology virtual recipient. The UNION lookup would return both rows and the resulting fan-out is almost never the intent — the error tells the admin to remove the relay-side row first.
`domains WHERE domain = X AND type = 'mailbox'`	12	Alias's domain isn't on the mailbox-topology side. Use Virtual Recipients for relay domains.

All four checks are advisory in the UI sense but enforced server-side so a forged form post can't bypass them.

Domain-delete dependency

There is no explicit dependency check on mailbox-domain deletion for aliases — but mailbox domains are typically not removed unless every mailbox under them is also being removed, and the alias rows become orphaned (`domain_id` no longer resolves) rather than actively harmful. Stale `mailbox_aliases` rows whose `domain_id` no longer exists are skipped by the page query because of the `INNER JOIN domains ... AND d.type = 'mailbox'`. Operational best practice: delete aliases first, then mailboxes, then the domain.

Failure semantics

What breaks	What happens
Blank alias address in Add	error 10 banner, no DB write
Invalid email format	error 11
Domain not in `domains` or not mailbox-type	error 12
Address already exists as a mailbox	error 13
Address already exists as an alias	error 14
Address already exists as a virtual recipient	error 17
Forward type with blank Delivers To	error 15
Delivers To target mailbox doesn't exist	error 16
Edit with missing alias_id	error 20
Edit / delete with stale alias_id	error 21

What breaks	What happens
MySQL `hermes_db_server` down	Postfix `virtual_alias_maps` lookups fail. Default behavior is to defer affected mail with a temporary error and retry — legitimate mail is held, not bounced.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_mailbox_aliases.cfm`	`hermes_commandbox`	Page + table + Add / Edit / Delete modals
`config/hermes/var/www/html/admin/2/inc/add_mailbox_alias_action.cfm`	`hermes_commandbox`	Add handler with the four-way conflict check
`config/hermes/var/www/html/admin/2/inc/edit_mailbox_alias_action.cfm`	`hermes_commandbox`	Edit handler — toggles `sender_login_maps` on send-as changes
`config/hermes/var/www/html/admin/2/inc/delete_mailbox_alias_action.cfm`	`hermes_commandbox`	Delete handler — removes alias row + any send-as map entry
`config/hermes/var/www/html/admin/2/inc/get_mailbox_alias_json.cfm`	`hermes_commandbox`	AJAX endpoint that hydrates the Edit modal
`/etc/postfix/mysql-virtual.cf`	`hermes_postfix_dkim` (volume-mounted)	The UNION lookup definition shared with `virtual_recipients`
`mailbox_aliases`, `sender_login_maps`, `mailboxes`, `domains`, `virtual_recipients`	`hermes_db_server`	Storage and conflict-detection tables

Nothing on this page shells out to Postfix — no `postmap`, no `postfix reload`, no template regeneration. The MySQL lookup picks up new rows on the next inbound message.

Related

- Email Relay > Virtual Recipients — the relay-topology equivalent. Use that page when the destination is external (Gmail, partner domain) or when fan-out to multiple destinations from one address is needed.
- Domains — the mailbox-domain list this page filters against. An alias's domain must exist there with `type = 'mailbox'`.
- Mailboxes — the destination mailbox list. The Delivers To dropdown is populated from active user mailboxes.

- Shared Mailboxes — when several users need to read the same incoming mail (rather than one user receiving forwards), use a shared mailbox instead of a forward alias.
- Mailbox Rules — Sieve-based filtering that runs on the destination mailbox after alias rewrite. Aliases route mail to a mailbox; Sieve rules then sort it within that mailbox.
- Settings — the global Email Server toggles. Aliases work regardless of the Mailbox Sharing master switch — they have no Dovecot-side configuration to be gated on.
- Authentication Settings — Submission-port authentication that the Send-As flag piggybacks on. A user must be able to authenticate to Submission as their primary address before Send-As lets them switch identities.

Domains

Domains

Admin path: **Email Server > Domains** (`view_mailbox_domains.cfm`, `inc/mailbox_domain_add_action.cfm`, `inc/mailbox_domain_edit_action.cfm`, `inc/mailbox_domain_delete_action.cfm`, `inc/get_mailbox_domain_json.cfm`, `inc/sync_mailbox_sans.cfm`, `inc/generate_nginx_configuration.cfm`, `inc/generate_transports.cfm`, `inc/generate_relay_domains.cfm`, `inc/generate_postfix_configuration.cfm`, `inc/add_domain_djigzo.cfm`, `inc/delete_domain_djigzo.cfm`).

This page manages the list of **mail-server domains** — the SMTP domains for which Hermes is itself the destination MTA, accepting inbound mail via Postfix and delivering it locally over LMTP to Dovecot mailboxes on `/mnt/vmail`. Each row pairs a `domains` row (`type='mailbox'`) with a `mailbox_domains` row (the per-domain SAN certificate binding) plus a `transport` row hardwired to `lmtp:[hermes_dovecot]:24`, a `senders` row, and a domain-wide `recipients` row carrying the default Amavis SVF policy.

This is the **mailbox-topology** counterpart to Email Relay > Domains. Both pages edit the same `domains` table but use the `type` column to partition rows: `type='relay'` belongs to the Relay page and forwards mail downstream; `type='mailbox'` belongs to this page and delivers mail locally. A single installation can run any mix of the two topologies — see Email Relay > Domains § Hermes topology overview for the high-level diagram.

> **Not to be confused with Email Relay > Domains.** The Relay page handles domains where Hermes forwards mail to a downstream MX (M365, Exchange, Google Workspace, an internal hub). This page handles domains where Hermes IS the final destination — mailboxes, IMAP/POP3, Submission, ManageSieve, Nextcloud Mail, autodiscover/autoconfig, DAV — backed by Dovecot.

Configuration storage

A single Add Mailbox Domain submission writes (or upserts) **five** rows across four tables and regenerates Postfix + Nginx + Ciphermail:

Table	Role
`domains`	One row per mailbox domain. `type='mailbox'` partitions it from the Relay page. Mailbox-specific metadata lives here: `default_quota_mb` (default per-mailbox quota in MB), `catchall_mailbox` (optional `postmaster@domain` style address), `nextcloud_enabled` (per-domain default — controls whether new mailboxes get a Nextcloud account), `enforce_mfa` (per-domain default for 2FA), `org_name` / `org_phone` / `org_address` / `org_website` / `org_logo_path` (Pro Organization Information for signature placeholder substitution), `allow_user_signatures` (gates the user-portal personal-signature editor for this domain).
`mailbox_domains`	One row per mailbox domain. `mailbox_certificate` foreign-keys into `system_certificates` — the per-domain TLS cert used by Dovecot IMAP/POP3/Submission, the autodiscover/autoconfig vhosts, and the DAV per-domain vhost.
`mailbox_sans`	One row per SAN prefix × domain (built from `additional_sans`). Drives per-SAN DNS/IP probe state for the certificate validator.
`transport`	Always `lmtp:[hermes_dovecot]:24` — mail-server domains never use SMTP forwarding.
`senders` + `recipients`	`senders.sender = domain`, `recipients.recipient = @domain` with `domain='1'` + the default `spam_policies` policy attached so Amavis runs on every inbound message.

The mailbox-domain row in `domains` deliberately reuses many columns from the relay path so the Postfix generators (`generate_transports`, `generate_relay_domains`, `generate_postfix_configuration`) treat both topologies uniformly — the only thing that differs is the transport string and the per-mailbox personal info / org info columns.

How a mailbox domain becomes live config

```
form submit  ──► mailbox_domain_add_action.cfm
                    |
                    |  validate domain + cert mode (Pro gate on 'auto')
                    |  duplicate-check against domains.domain
                    |
                    |  --- write DB ---
                    |  INSERT transport (lmtp:[hermes_dovecot]:24)
                    |  INSERT senders   (sender = domain, action = OK)
```

```
              |  INSERT recipients(recipient = @domain,
              |                    domain='1', policy_id=default,
              |                    status='OK')
              |  INSERT domains   (..., type='mailbox', default_quota_mb,
              |                    catchall_mailbox, nextcloud_enabled,
              |                    enforce_mfa, created_at, updated_at)
              |  UPSERT mailbox_domains (domain, mailbox_certificate)
              |
              |  --- regenerate ---
              v
sync_mailbox_sans.cfm           -> mailbox_sans (one per prefix)
generate_transports.cfm         -> /etc/postfix/transport + postmap
generate_relay_domains.cfm      -> /etc/postfix/relay_domains
generate_postfix_configuration.cfm
                                -> /etc/postfix/main.cf
                                   + postfix reload (docker exec)
generate_nginx_configuration.cfm
                                -> per-domain Nginx vhosts
                                   (autodiscover, autoconfig, DAV)
add_domain_djigzo.cfm           -> registers domain in Ciphermail
occ group:add <domain>          -> Nextcloud group (if NC enabled)
                                   (docker exec hermes_nextcloud)
              |
              v
preload_restart_nginx.cfm?returnUrl=... (Nginx restart, then redirect)
```

Edit follows the same shape minus the inserts (UPDATE on `domains`, UPSERT on `mailbox_domains`, re-sync SANs, regen Nginx). Delete reverses the writes after running dependency checks (see Delete below).

Fields on the page

Add Mailbox Domain card

Field	Default	Notes

Domain Name	(empty)	Trimmed, lower-cased, validated by the email-trick. Rejected if the domain already exists in `domains` (as relay or mailbox). The `mailbox_domains` table is allowed to have a pre-existing row (left over from prior ACME work) — it gets UPSERTed in place.
Default Quota (GB)	`5`	Per-domain default for new mailboxes. Stored in DB as MB (`default_quota_mb`). 0.5 GB minimum, 1024 GB max, 0.5 GB step. The per-mailbox quota is set on Mailboxes; this is the value pre-filled when adding a new mailbox under the domain.
Catch-All Mailbox	(empty)	Optional. An existing mailbox address that receives mail for any unknown recipient at the domain. Free-text — admin's responsibility to point at a real mailbox.
SAN Certificate — Auto-managed (Let's Encrypt)	Pro: checked / Community: disabled	*Pro Edition only.* Creates a placeholder Acme row in `system_certificates`; the certificate validator then validates SAN DNS + IP, requests the cert, and auto-renews. Zero maintenance once DNS is in place.
SAN Certificate — Use existing certificate	Community: checked	Pulls from `system_certificates` where `san='1'` OR the row is a system-flagged placeholder. The dropdown labels system placeholders as `TEMPORARY PLACEHOLDER (replace before production)` and sorts them last so the default is a real SAN cert.
Enable Nextcloud webmail for this domain	unchecked	Per-domain default for new mailboxes. When checked, creates a Nextcloud group named after the domain (via `occ group:add`) and pre-fills the Nextcloud toggle on the Add Mailbox form. Does **not** retroactively enable NC for existing mailboxes.
Require Two-Factor Authentication for this domain	unchecked	Per-domain default for new mailboxes. Same convention as Nextcloud — defaults only, no cascade to existing rows.

Mailbox domains table

Sortable, searchable, exportable. Columns:

Column	Source	Badge logic
Domain	`domains.domain`	Plain text
Certificate	`system_certificates.friendly_name` via `mailbox_domains.mailbox_certificate`	Link to `view_system_certificates.cfm`; badge `Auto (LE)` for `type='Acme'`, `Imported` otherwise; `Missing` if no binding
Cert Status	derived from `mailbox_sans` rows for the domain	`Verified` (all SANs DNS-confirmed) / `Partial` / `Awaiting Cert` / `Pending` / `DNS Failed` / `No SANs` / `No Cert`. Imported certs always show `Imported` .
Default Quota	`default_quota_mb`	Rendered in GB
Catch-All	`catchall_mailbox`	Em-dash if NULL
Nextcloud	`nextcloud_enabled`	`Enabled` (success) / `Disabled` (secondary)
2FA	`enforce_mfa`	`Required` (success) / `Optional` (secondary)
DKIM	aggregated from `dkim_sign`	`Active` / `Disabled` / `None` — same logic as the Relay page
Actions	—	Edit (opens modal), DNS Records (opens helper modal), DKIM Keys (→ `edit_domain_dkim.cfm`), Delete

Edit Mailbox Domain modal

Opens via `openEditModal(id)`, fetches `./inc/get_mailbox_domain_json.cfm` over AJAX, hydrates every form field. **Domain Name is read-only on edit** — same convention as the Relay page (renaming a domain across all the joined tables is risky enough that the page enforces add-and-delete instead).

The Edit modal carries everything from Add plus three extra sections that exist only after creation:

Section	Notes
Organization Information *(Pro only)*	`org_name`, `org_phone`, `org_address`, `org_website`. Used by the body milter's signature substitution to fill `{{org.name}}` , `{{org.phone}}`, `{{org.address}}`, `{{org.website}}` placeholders in organizational signatures. See Organizational Signatures. All fields optional. Community installs see a Pro upsell badge and the inputs are HTML-disabled — the action handler also skips the UPDATE on Community so a tampered form post can't write data and existing values survive a Pro→Community downgrade.

Section	Notes
`org_logo_path`	Column exists but no UI yet — placeholder for follow-up integration with the inline image pipeline that ships organizational signature logos.
Allow users in this domain to manage their own signatures	Per-domain toggle (`allow_user_signatures`, both tiers). When on, mailbox users see a Signature page in `/users/2/`. When off, the page is hidden and any user-edited signature rows for the domain are ignored at send time. The body milter respects this on the next signature-map regen.

The modal explicitly tags `Nextcloud webmail` and `Two-Factor Authentication` as **defaults for new mailboxes** — toggling them does **not** flip the corresponding per-mailbox flags on existing rows. To change an existing mailbox use the per-mailbox Edit Options dialog on Mailboxes.

DNS Records modal

Per-domain reference card surfacing every DNS record an operator needs to publish for the domain to actually receive mail and support client auto-discovery: MX, autoconfig/autodiscover CNAMEs, the SRV chain (`_imap`, `_imaps`, `_pop3`, `_pop3s`, `_submission`, `_submissions`, `_sieve`, `_autodiscover`), CalDAV/CardDAV SRV+TXT (`_caldavs`, `_carddavs` with `path=/nc/remote.php/dav/`), plus example SPF and DMARC TXT records. DKIM TXT records are listed separately under DKIM Keys.

Console host (`parameters2 console.host`) is interpolated into every record so the values are copy-paste ready.

Delete Mailbox Domain modal

Confirms the destructive action. The handler runs two dependency checks before allowing the delete:

Check	If it returns rows →
Mailboxes under this domain (`mailboxes.domain_id = <id>`)	Error 16, abort, link admin to Mailboxes to clear them first
Recipients still attached to the domain (excluding the domain-wide `@domain` row)	Error 17, abort

If both pass, the handler:

1. Captures the bound `mailbox_certificate` id (for orphan-cert detection).
2. Deletes `mailbox_domains`, `domains`, `transport`, `senders`, `recipients` (the five rows linked at creation).
3. Deletes the domain's `mailbox_sans` rows **directly** (does not call `sync_mailbox_sans.cfm` — sync would nuke validated IP/DNS state on other domains if it ran during a delete→re-add

cycle).

4. Regenerates Postfix + Nginx, deregisters from Ciphermail, runs `occ group:delete <domain>` against Nextcloud (non-fatal).
5. If the bound certificate now belongs to no other mailbox domain, surfaces an **Orphaned Certificate** flash on the next page render pointing the admin to System Certificates. The cert is **not** auto-deleted because Let's Encrypt limits duplicate certificate issuance to 5 per week and accidentally throwing away a cert you might re-need is a non-recoverable mistake.

> **Operational consequence — mailbox data on disk is NOT deleted.** The delete handler removes the Dovecot domain wiring (transport, recipient acceptance, cert binding) but does **not** touch `/mnt/vmail/<domain>/`. If you intend to permanently retire a domain, remove the mailbox directories from the host after the delete completes.

Per-domain Nginx vhosts

Each mailbox domain generates per-domain Nginx vhosts for:

- `autodiscover.<domain>` — Outlook / iOS Mail auto-configuration
- `autoconfig.<domain>` — Thunderbird / K-9 Mail auto-configuration
- The DAV chain via the SRV records published by the DNS Records modal

Add and Edit both call `generate_nginx_configuration.cfm` then redirect through `preload_restart_nginx.cfm` (the canonical restart pattern that avoids the brief `ERR_CONNECTION_REFUSED` blip in user-driven flows).

> **Known gotcha — editing the vhost template does NOT update already-generated vhosts.** The generator writes per-domain files at install time and on subsequent saves. If the underlying template (in `/opt/hermes/templates/`) is hand-edited, existing vhost files stay stale until each domain is re-saved (or until a separate re-render pass is run). Operators changing the template should plan for a bulk re-save afterwards.

Cert SAN binding and the validator

`sync_mailbox_sans.cfm` reads `additional_sans` (the global list of prefixes — `mail.`, `autodiscover.`, `autoconfig.`, plus any custom ones) and writes one `mailbox_sans` row per prefix × this domain, pointing at the selected certificate. Each row carries IP and DNS probe state.

A separate scheduled task (System > SAN Management) walks `mailbox_sans` every 30 minutes, probes each subdomain for the expected IP and DNS A/CNAME record, and updates `ip_result_msg` / `dns_result_msg`. The Cert Status column on the main table summarizes these results.

For Pro Edition's auto-managed certs the validator then triggers a Let's Encrypt issuance once every SAN passes both probes. For imported certs the probes are informational only — the cert is trusted as-is.

See SAN Management for the full SAN editor.

Failure semantics

What breaks	What happens
Domain name empty	`session.m = 10`, redirect, no DB write
Domain name fails email-trick validation	`session.m = 11`, redirect, no DB write
Domain already exists in `domains` (relay or mailbox)	`session.m = 12`, redirect, no DB write
Auto-managed selected on Community edition	`session.m = 14`, redirect, no DB write
`cert_id` invalid for `Use existing`	`session.m = 13`, redirect, no DB write
`default_quota_gb` not a positive number	`session.m = 15`, redirect, no DB write
Delete blocked: mailboxes still exist	`session.m = 16`, redirect, abort. Detail count shown in the alert.
Delete blocked: recipients still exist	`session.m = 17`, redirect, abort
`add_domain_djigzo.cfm` errors during Ciphermail registration	Domain is already in the DB; encryption gateway will not know about the domain until the next re-save. Non-fatal.
`occ group:add` fails (NC down, group exists)	Non-fatal `cftry` — mailbox-domain creation still succeeds; admin can re-toggle in Edit to retry
Nginx vhost regen fails	Domain is in the DB; per-domain auto-discovery URLs will return errors until the next successful Edit/regen
Postfix reload fails	Live config keeps the previous values; reload error is in container logs

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_mailbox_domains.cfm`	`hermes_commandbox`	Page + Add card + Edit/Delete/DNS modals
`config/hermes/var/www/html/admin/2/inc/mailbox_domain_add_action.cfm`	`hermes_commandbox`	Add handler
`config/hermes/var/www/html/admin/2/inc/mailbox_domain_edit_action.cfm`	`hermes_commandbox`	Edit handler
`config/hermes/var/www/html/admin/2/inc/mailbox_domain_delete_action.cfm`	`hermes_commandbox`	Delete handler
`config/hermes/var/www/html/admin/2/inc/get_mailbox_domain_json.cfm`	`hermes_commandbox`	AJAX hydrator for the Edit modal
`config/hermes/var/www/html/admin/2/inc/sync_mailbox_sans.cfm`	`hermes_commandbox`	Builds `mailbox_sans` rows from `additional_sans` × domain
`config/hermes/var/www/html/admin/2/inc/generate_nginx_configuration.cfm`	`hermes_commandbox`	Per-domain vhost generator
`config/hermes/var/www/html/admin/2/inc/generate_transports.cfm` / `generate_relay_domains.cfm` / `generate_postfix_configuration.cfm`	`hermes_commandbox`	Shared Postfix regenerators (also used by Email Relay > Domains)
`config/hermes/var/www/html/admin/2/inc/add_domain_djigzo.cfm` / `delete_domain_djigzo.cfm`	`hermes_commandbox`	Ciphermail registration
`config/hermes/var/www/html/admin/2/inc/signature_regen_map.cfm`	`hermes_commandbox`	Rebuilds the body milter's `signature_by_sender` map + `sender_data.json` after org info / `allow_user_signatures` edits
`config/hermes/var/www/html/admin/2/preload_restart_nginx.cfm`	`hermes_commandbox`	Nginx restart shim used on Add and Edit redirect
`/etc/postfix/transport` + `.db`, `/etc/postfix/relay_domains`, `/etc/postfix/main.cf`	`hermes_postfix_dkim`	Postfix maps regenerated on every save
Per-domain Nginx vhost files	`hermes_nginx` (mounted)	Generated by `generate_nginx_configuration.cfm`
`domains`, `mailbox_domains`, `mailbox_sans`, `transport`, `senders`, `recipients`	`hermes_db_server`	The mailbox-domain row group
`system_certificates`, `additional_sans`	`hermes_db_server`	Cert inventory + SAN prefix list
`hermes_nextcloud` container	—	`occ group:add` / `group:delete <domain>` for the per-domain NC group
`hermes_ciphermail` container	—	Domain registration via CLITool

Every shell-out uses `docker exec ...` per the standard Hermes pattern.

Related

- Email Relay > Domains — the relay topology twin. Mailbox and relay domains share the same `domains` table but partition on `type`. **Do not confuse with this page.**
- Email Server > Mailboxes — per-mailbox CRUD. A mailbox domain is meaningless without mailboxes; add the domain here first, then add mailboxes there.
- Email Server > Settings — global Dovecot configuration (TLS profile, compression, encryption at rest, quota warning thresholds). The per-domain default quota set here is what Email Server > Settings's warning thresholds measure against on a per-mailbox basis.
- Email Server > Aliases — alias addresses that resolve to local mailboxes within a mailbox domain.
- Email Server > Shared Mailboxes — shared mailboxes are per-domain just like regular mailboxes.
- Email Server > Mailbox Rules — per-mailbox Sieve rules.
- Email Server > SAN Management — the global SAN prefix list (`additional_sans`) that `sync_mailbox_sans.cfm` multiplies against every mailbox domain.
- System Certificates — certificate inventory that the SAN Certificate dropdown draws from, including the bootstrap placeholder cert.
- LDAP RemoteAuth — mailbox users can authenticate against an upstream LDAP/AD using the same `auth_type='remote'` pattern documented for relay recipients.
- Organizational Signatures *(Pro)* — consumer of the Organization Information fields on the Edit modal.

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.

Mailboxes

Mailboxes

Admin path: **Email Server > Mailboxes** (`view_mailboxes.cfm`, `add_mailbox.cfm`, `inc/add_mailbox_action.cfm`, `inc/edit_mailbox_action.cfm`, `inc/edit_mailbox_encryption_action.cfm`, `inc/edit_mailbox_access_control_action.cfm`, `inc/delete_mailbox_action.cfm`, `inc/get_mailbox_json.cfm`, `inc/ldap_add_user_mailbox.cfm`, `inc/ldap_add_user_mailbox_remoteauth.cfm`, `inc/ldap_add_user_groups_mailbox.cfm`, `inc/ldap_delete_user_mailbox.cfm`, `inc/nextcloud_provision_user.cfm`, `inc/signature_regen_map.cfm`, `inc/send_mailbox_welcome_email.cfm`, `inc/send_mailbox_welcome_email_remoteauth.cfm`, `inc/admin_resend_mobile_setup_action.cfm`, `inc/rotate_nc_password_action.cfm`).

This page manages **individual mailboxes** inside the mail-server topology — one row per address in the `mailboxes` table, joined to a `recipients` row that carries the per-recipient policy stack (SVF policy, encryption flags, S/MIME certs, PGP keyrings, 2FA enforcement, auth type). A mailbox is the local-delivery counterpart to a Relay Recipient — same `recipients` row shape, different `recipient_type` column value (`'mailbox'` vs `'relay'`) and a sibling row in `mailboxes` that gives Dovecot a userdb entry.

This is the **per-mailbox** half of the mail-server topology. Pairs with Domains (the domains those mailboxes live under and inherit defaults from), Settings (global Dovecot config and quota warning thresholds), and the per-address feature pages: Aliases, Shared Mailboxes, Mailbox Rules, and per-mailbox app passwords.

Mailbox vs Alias vs Shared Mailbox vs Relay Recipient

Four address concepts share the namespace under a mailbox domain; keep them straight:

Concept	Stored in	Has Dovecot mailbox?	Local sign-in?
Mailbox (this page)	`mailboxes` (`mailbox_type='user'`) + `recipients` (`recipient_type='mailbox'`)	Yes — Dovecot LMTP delivery to `/mnt/vmail/<domain>/<user>/`	Yes — IMAP/POP3/Submission, web portal, Nextcloud

Concept	Stored in	Has Dovecot mailbox?	Local sign-in?
Alias	`mailbox_aliases`	No — forwards to one or more mailboxes (or silently discards)	No
Shared Mailbox	`mailboxes` (`mailbox_type='shared'`) + `shared_mailbox_permissions`	Yes — but accessed via Dovecot ACL from owner mailboxes	No direct login — owners reach it from their own session
Relay Recipient	`recipients` (`recipient_type='relay'`)	No — forwarded to a downstream MX	Yes for web portal / Submission (via app passwords)

See Aliases and Shared Mailboxes for the alias and shared variants, and Email Relay > Relay Recipients for the relay-topology equivalent.

What a Mailbox row carries

```
mailboxes table  (Dovecot userdb-driving row)
├── id, domain_id          -> joins to domains where type='mailbox'
├── username               full email (e.g. jsmith@company.com)
├── name                   display name
├── quota                  per-mailbox quota in BYTES (DB stores bytes;
│                          UI shows GB)
├── active                 1/0 — Dovecot rejects auth when 0
├── nextcloud_enabled      per-mailbox Nextcloud flag
├── mailbox_type           'user' | 'shared'
└── first_name, last_name, title, phone, mobile, department
                           (Pro Personal Information for signature
                            substitution)

recipients table  (paired row, recipient_type='mailbox')
├── recipient              same as mailboxes.username
├── policy_id              -> spam_policies (SVF policy)
├── auth_type              'local' | 'remote'
├── remoteauth_domain      NULL if local; mapping key if remote
├── enforce_mfa            0 | 1 (admin policy)
├── pdf_enabled / smime_enabled / pgp_enabled / digital_sign
└── (cert + keyring slots populated lazily by cert_generation_queue)
```

Side tables linked at create-time or lazily:

Table	Role
`user_settings`	`report_enabled` (quarantine notifications), `train_bayes`, `download_msg`, `timezone`, `ldap_username`
`maddr`	Amavis address index — required for the user portal session machinery
`sender_login_maps`	Postfix `smtpd_sender_login_maps` entry — permits the mailbox owner to send AS their own address from Submission
`app_passwords`	Per-mailbox application passwords (Argon2-hashed) for IMAP/SMTP/CalDAV/CardDAV/Nextcloud. The Add flow creates an initial `Hermes System` app password used by the Nextcloud Mail auto-profile.
`recipient_certificates`, `recipient_keystores`	S/MIME cert + PGP keyring slots (lazy — populated by the queue)
`cert_generation_queue`	Async S/MIME + PGP generation jobs
`mailbox_aliases`	If any aliases exist pointing at the mailbox
`shared_mailbox_permissions`	If the mailbox is granted access to any shared mailbox

Add Mailbox — `add_mailbox.cfm`

Single-mailbox page (not a bulk form). The admin selects a target domain, fills in the address local-part + display name + quota + auth mode + per-recipient stack (policy, notifications, encryption flags), and submits. `add_mailbox_action.cfm` then runs the full creation pipeline:

```
form submit  ──► add_mailbox_action.cfm
                    |
                    |  validate domain + email + auth mode
                    |  duplicate-check against recipients, mailboxes,
                    |     mailbox_aliases, virtual_recipients
                    |
                    |  --- write DB ---
                    |  INSERT recipients   (recipient_type='mailbox', policy,
                    |                       auth_type, remoteauth_domain,
                    |                       enforce_mfa, encryption flags)
                    |  INSERT maddr        (Amavis address index)
                    |  INSERT user_settings(notifications, train_bayes,
                    |                       download_msg, timezone)
                    |  INSERT mailboxes    (domain_id, username, name,
                    |                       quota, active=1, nextcloud_enabled)
```

```
|  INSERT sender_login_maps (permits send-as)
|
|  --- LDAP ---
|  auth_type=local  : ldap_add_user_mailbox.cfm
|                     (random userPassword, will be reset)
|  auth_type=remote : ldap_add_user_mailbox_remoteauth.cfm
|                     (no userPassword; seeAlso pointer to
|                     upstream DN, associatedDomain set to
|                     remoteauth_domain)
|  ldap_add_user_groups_mailbox.cfm
|    -> cn=mailboxes,ou=groups,dc=hermes,dc=local
|    -> cn=one_factor OR cn=two_factor (per enforce_mfa)
|  if NC enabled:
|    -> cn=nextcloud,ou=groups,dc=hermes,dc=local
|
|  --- Nextcloud (if NC enabled) ---
|  nextcloud_provision_user.cfm
|    -> occ user:add with RANDOM internal password
|       (not the user's real password — they reach NC
|        via OIDC; the internal password is defense-in-depth)
|    -> occ user:setting to pre-fill email + display name
|    -> create initial Hermes System app password
|       (used by the Mail app account profile)
|    -> create Nextcloud Mail account profile
|       (IMAP+SMTP credentials pre-wired)
|
|  --- lazy cert / keyring queue ---
|  if smime_enabled : INSERT cert_generation_queue (smime)
|  if pgp_enabled   : INSERT cert_generation_queue (pgp)
|
|  --- send welcome ---
|  local  : send_mailbox_welcome_email.cfm
|           (password-reset link, 30-min expiry)
|  remote : send_mailbox_welcome_email_remoteauth.cfm
|           (sign-in with organization password)
|
|  --- signature map ---
|  if Pro: signature_regen_map.cfm
|    -> rebuild body milter signature_by_sender map
|    -> rebuild sender_data.json
```

```
                    |
                    v
            cflocation -> view_mailboxes.cfm with session.m = 1
```

Dovecot mailbox directories on `/mnt/vmail/<domain>/<user>/` are NOT pre-created. Dovecot auto-creates the directory tree on first LMTP delivery or first IMAP login. The mailbox row alone is enough.

Password handling

Local-auth mailboxes:

- The admin enters a password on the Add form (12-char minimum, no special chars, checked against the HIBP "Have I Been Pwned" k-anon range API).
- The same password is stored in three places, each hashed by its consuming subsystem: OpenLDAP `userPassword` (Argon2id via `slappasswd -o module-load=argon2.la -h {ARGON2}`), `app_passwords` initial `Hermes System` row (Argon2id), and the Nextcloud internal user password (only on the NC side, set by `occ user:add` — but immediately replaced with a random value by `nextcloud_provision_user.cfm`, see Phase 1 of #197).
- Argon2id hashing uses the canonical `docker run --rm authelia/authelia:<version> authelia crypto hash generate argon2 --password <value>` pattern. No host-side `argon2` binary required.

RemoteAuth mailboxes (`auth_type='remote'`):

- No password is captured. The local LDAP entry has no `userPassword`; bind goes through the OpenLDAP remoteauth overlay to the upstream AD/LDAP per the `remoteauth_domain` mapping (see LDAP RemoteAuth).
- `app_passwords` still issues Hermes-side credentials for IMAP/SMTP/DAV — these remain Hermes-owned regardless of upstream password rotation.

The Mailboxes table

Single DataTable with 21 columns and an optional Domain filter dropdown above (populated only when ≥1 domain has mailboxes). Per-row columns:

Column	Source	Notes

Actions	—	Dropdown: Edit Options, Edit Encryption, Reset 2FA Devices, Manage App Passwords (→ `view_mailbox_app_passwords.cfm`), Send Mobile Setup Profile, Rotate NC Internal Password (only if NC enabled), Delete
S/MIME	link to `view_recipient_certificates.cfm?type=1&id=...`	Per-mailbox cert manager
PGP	link to `view_recipient_keyrings.cfm?type=1&id=...`	Per-mailbox keyring manager
Email	`mailboxes.username`	Full address
Display Name	`mailboxes.name`	
Domain	join on `domains.domain`	
Quota	`mailboxes.quota / 1024 / 1024 / 1024`	Rendered in GB
Auth	`recipients.auth_type`	`LOCAL` badge or `REMOTE` badge (tooltip shows `remoteauth_domain`)
2FA	LDAP `cn=two_factor` + `enforce_mfa`	Two independent pills — see Two-pill 2FA column
Policy	`spam_policies.policy_name`	
Notifications, Train Bayes, Download Msgs	`user_settings.*`	`YES` (success) / `NO` (secondary)
PDF / S/MIME / PGP Encrypt, Sign All	`recipients.*`	`YES` / `NO`
S/MIME Cert, PGP Keyring	join against `recipient_certificates`, `recipient_keystores`	`YES` (green) if a cert/keyring exists; spinner badge if a job is `pending` / `processing` in `cert_generation_queue`
Nextcloud	`mailboxes.nextcloud_enabled`	`YES` / `NO`
Status	`mailboxes.active`	`Active` (success) / `Inactive` (danger) — Dovecot rejects auth when `active=0`

The query filters `WHERE m.mailbox_type = 'user'` so shared mailboxes do not appear here — they have their own page at Shared Mailboxes.

Two-pill 2FA column

Same two-orthogonal-states model as Email Relay > Relay Recipients § Two-pill 2FA column. Admin enforcement (`recipients.enforce_mfa`) and user enrollment (`cn=two_factor` LDAP membership) are

decoupled, so the cell can show **Enrolled**, **Required**, both, or em-dash.

The page pulls all `cn=two_factor` group members in a single `ldapsearch` (via `docker exec hermes_ldap ldapsearch -Y EXTERNAL`) once per render, then each row checks for its DN substring in the result — avoids an N+1 LDAP roundtrip storm.

Edit Options modal — AJAX pre-fill

Opens via `loadEditModal(mailboxId)`, hits `inc/get_mailbox_json.cfm` over AJAX, hydrates every field with the mailbox's current values. Unlike the Relay Recipients bulk-edit foot-gun, this modal is **always single-mailbox** — there is no bulk Edit Options on this page.

Fields:

Section	Notes
Email Address	Read-only
Display Name	`mailboxes.name`
Personal Information *(collapsible, Pro only)*	`first_name`, `last_name`, `title`, `phone`, `mobile`, `department`. Used by signature placeholder substitution (`{{user.first_name}}`, `{{user.title}}`, etc.) and by department-based signature resolution. Department field uses a typeahead datalist built from the domain's existing departments via `inc/get_dept_options.cfm`. Community inputs are HTML-disabled and the action handler skips the UPDATE on Community so values survive a Pro→Community downgrade.
Mailbox Quota (GB)	Per-mailbox override of the domain default
Status	`Active` / `Inactive`
SVF Policy	Populated from `spam_policies` where `custom='1' OR default_policy='1'`
Quarantine Notifications	`user_settings.report_enabled`
Train Bayes Filter	`user_settings.train_bayes` — with prominent warning that improperly-trained Bayes affects ALL recipients
Download Messages from User Portal	`user_settings.download_msg` — with malware-risk warning
Nextcloud Webmail	`mailboxes.nextcloud_enabled`. **Enabling for an existing user requires a new password** (NC needs the password to provision the Mail app profile) — error 51 if the admin enables NC without setting a password. **Disabling** shows a `Keep Nextcloud account data` checkbox that gates whether the NC user account and data are preserved or permanently deleted.

Section	Notes
Two-Factor Authentication	`recipients.enforce_mfa`. When enabled, the user's web portal access becomes restricted to Account Settings, My App Passwords, Set Up Your Devices, and Webmail & Apps until they enroll. Email/calendar/contacts keep working throughout — only the web portal is gated. The 0→1 transition triggers an LDAP group move from `cn=one_factor` to `cn=two_factor` so Authelia challenges them on next sign-in.
Timezone	`user_settings.timezone` (Java `ZoneId` list). Used for the vacation auto-reply schedule and dashboard timestamps.
Authentication Type	Read-only — `local` or `remote`
Change Password *(local auth only)*	Optional. Minimum 12 chars, no special chars, HIBP-checked. Blank keeps the current password.

Edit Encryption modal

Per-mailbox encryption flags (`pdf_enabled`, `smime_enabled`, `digital_sign`, `pgp_enabled`) plus the cert/keyring generation parameters (CA, validity, key size, algorithm, PGP key length). Submit queues async cert + keyring generation into `cert_generation_queue` if a flag flips on and no existing cert/keyring is present — same lazy-queue pattern as Relay Recipients.

Reset 2FA Devices modal

Single-purpose modal that clears Authelia TOTP and WebAuthn device registrations via `docker exec hermes_authelia authelia storage user totp delete` and `... webauthn delete --all`. Two modes:

Mode	What it does
Default	Deletes TOTP + WebAuthn devices. User stays under 2FA enforcement and re-registers on next sign-in. "User lost their phone" recovery.
Nuclear *(checkbox)*	Also moves the user from `cn=two_factor` back to `cn=one_factor`. Admin override; if `enforce_mfa` is still 1 the next Edit Options save will reverse the LDAP move.

> **Does not affect Duo Push.** Duo enrollments live on Duo's cloud servers. Use the Duo Admin Console.

Send Mobile Setup Profile

Per-mailbox action that emails the user a signed iOS / iPadOS mobileconfig profile pre-wired with IMAP + Submission + CalDAV + CardDAV + the appropriate account name and email. The link in the email expires in 30 minutes and works only once.

Handler is `inc/admin_resend_mobile_setup_action.cfm`. The mobileconfig generator itself is shared with the user-portal Setup Your Devices wizard.

Rotate NC Internal Password

Visible only when `mailboxes.nextcloud_enabled = 1`. Generates a new random local password for the Nextcloud user via `docker exec hermes_nextcloud occ user:resetpassword` and the displayed value is **never shown** — it is purely defense-in-depth.

Background: the Nextcloud internal password was historically set to the user's real password, which silently allowed CalDAV/CardDAV to accept the org password and defeat the app-password isolation boundary (closed in #197 Phase 1). The internal password is now random and unused by anything user-facing — users reach NC via OIDC, and DAV/IMAP go through app passwords. This admin action lets the admin re-randomize on demand without touching the user's actual credentials.

Delete

Cascading delete that mirrors the create pipeline in reverse, with the same cleanup discipline as Relay Recipients (the goal is zero-orphan rows). Per mailbox:

```
For the selected mailbox ID:
1. Read mailboxes row + user_settings (for ldap_username)
2. Remove LDAP from cn=mailboxes  (before delete_internal_recipients
                                   runs ldap_delete_user_relay)
3. (If NC enabled) Remove from cn=nextcloud LDAP group
4. delete_internal_recipients.cfm
     - docker exec hermes_authelia authelia storage user totp delete
     - docker exec hermes_authelia authelia storage user webauthn delete --all
     - LDAP user entry delete
     - cert_generation_queue cancel + recipient_certificates clear
     - recipient_keystores + Ciphermail keystore clear
```

```
     - wblist, mailaddr, password_reset_requests cancel
5. DELETE mailboxes WHERE id = <id>
6. DELETE sender_login_maps WHERE login_user = <email>
7. DELETE user_settings (if not already cleared by step 4)
8. Re-sync any shared mailbox vfile ACLs the user was a member of
   (so the deleted user vanishes from sharer lists)
9. DELETE app_passwords WHERE username = <email>
10. (If NC enabled AND admin did NOT check "Keep Nextcloud data")
    docker exec hermes_nextcloud occ user:delete <user>
11. signature_regen_map.cfm (rebuild body milter map without this user)
```

The Nextcloud user/data preservation is opt-in via the `Keep Nextcloud account data` checkbox surfaced when toggling NC off in Edit Options — deletion from this page asks the same question.

> **Dovecot mailbox data on disk is NOT deleted.** `/mnt/vmail/<domain>/<user>/` survives the delete. If you intend to permanently retire the mailbox, remove the directory from the host after the delete completes. This matches the per-domain behavior on Domains.

Local-auth vs RemoteAuth — the credential split

Identical model to relay recipients. See Email Relay > Relay Recipients § Local-auth vs RemoteAuth and Authentication Settings for the full four-credential architecture.

For mailboxes specifically: app passwords are always Hermes-issued regardless of `auth_type`. RemoteAuth mailbox users' upstream directory password is exposed only to the web gate (via the LDAP overlay's pass-through bind) — never to Dovecot or the Nextcloud Mail profile.

> **Known forward-looking gap (#102).** RemoteAuth mapping deletion validation in `view_remoteauth.cfm` and `edit_remoteauth_mapping.cfm` currently only checks `system_users` and `recipients`. When RemoteAuth-for-mailboxes activity grows, the validation must add a third query against `mailboxes` so an in-use mapping cannot be stranded. See LDAP RemoteAuth § Deletion validation.

Failure semantics

What breaks	What happens
Quota not a positive number	`session.m = 15`, redirect, no DB write
Missing required form fields	`session.m = 20`, redirect, no DB write
Mailbox not found (Edit/Delete)	`session.m = 21`, redirect, no DB write
Password under 12 characters	`session.m = 22`, redirect, no DB write
Password found in HIBP breach	`session.m = 99`, redirect, no DB write
HIBP API unavailable	`session.m = 100`, warning banner, mailbox still rejected (fail-closed)
Enabling NC for existing user without setting a password	`session.m = 51`, redirect, no DB write
Mobile setup profile email failed but profile staged	`session.m = 83`, warning banner, link still works
Duplicate email (against recipients / mailboxes / aliases / virtual_recipients)	redirect to `add_mailbox.cfm` with appropriate alert
LDAP add fails after DB inserts succeed	DB row exists; subsequent IMAP/SMTP login fails until the LDAP entry is created (admin can re-save Edit Options or delete and re-add)
Nextcloud `occ user:add` fails	Mailbox creation succeeds; NC toggle effectively becomes a no-op until re-toggled
`cert_generation_queue` row stuck in `processing`	Surfaces in the Add Recipient / Add Mailbox alert banner via Pending S/MIME or PGP generation; retry via the same Retry Failed Jobs button on the Relay page

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_mailboxes.cfm`	`hermes_commandbox`	Main page + Edit Options / Edit Encryption / Reset 2FA / Delete modals
`config/hermes/var/www/html/admin/2/add_mailbox.cfm`	`hermes_commandbox`	Add page (single mailbox, full per-recipient stack)
`config/hermes/var/www/html/admin/2/inc/add_mailbox_action.cfm`	`hermes_commandbox`	Add handler — orchestrates DB + LDAP + NC + cert queue + welcome email
`config/hermes/var/www/html/admin/2/inc/edit_mailbox_action.cfm`	`hermes_commandbox`	Edit Options handler

Path	Owner	Role
`config/hermes/var/www/html/admin/2/inc/edit_mailbox_encryption_action.cfm`	`hermes_commandbox`	Edit Encryption handler + cert/keyring queue insertion
`config/hermes/var/www/html/admin/2/inc/edit_mailbox_access_control_action.cfm`	`hermes_commandbox`	Reset 2FA Devices handler (TOTP + WebAuthn clear + optional nuclear move)
`config/hermes/var/www/html/admin/2/inc/delete_mailbox_action.cfm`	`hermes_commandbox`	Delete cascade
`config/hermes/var/www/html/admin/2/inc/get_mailbox_json.cfm`	`hermes_commandbox`	AJAX hydrator for Edit Options
`config/hermes/var/www/html/admin/2/inc/get_dept_options.cfm`	`hermes_commandbox`	Per-domain department datalist (typeahead)
`config/hermes/var/www/html/admin/2/inc/ldap_add_user_mailbox.cfm` / `ldap_add_user_mailbox_remoteauth.cfm`	`hermes_commandbox`	Local / remote LDAP entry creation
`config/hermes/var/www/html/admin/2/inc/ldap_add_user_groups_mailbox.cfm`	`hermes_commandbox`	Group assignment: `cn=mailboxes`, `cn=one_factor` / `cn=two_factor`, `cn=nextcloud`
`config/hermes/var/www/html/admin/2/inc/ldap_delete_user_mailbox.cfm`	`hermes_commandbox`	LDAP entry removal on delete
`config/hermes/var/www/html/admin/2/inc/nextcloud_provision_user.cfm`	`hermes_commandbox`	NC user creation, random internal password, Mail app profile, initial app password
`config/hermes/var/www/html/admin/2/inc/rotate_nc_password_action.cfm`	`hermes_commandbox`	On-demand NC internal password rotation
`config/hermes/var/www/html/admin/2/inc/admin_resend_mobile_setup_action.cfm`	`hermes_commandbox`	Mobile setup profile generation + email
`config/hermes/var/www/html/admin/2/inc/send_mailbox_welcome_email.cfm` / `send_mailbox_welcome_email_remoteauth.cfm`	`hermes_commandbox`	Welcome email (local: reset link; remote: org-password instructions)
`config/hermes/var/www/html/admin/2/inc/signature_regen_map.cfm`	`hermes_commandbox`	Body milter `signature_by_sender` map + `sender_data.json` rebuild
`mailboxes`, `recipients`, `user_settings`, `maddr`, `sender_login_maps`, `app_passwords`, `recipient_certificates`, `recipient_keystores`, `cert_generation_queue`, `mailbox_aliases`, `shared_mailbox_permissions`, `wblist`, `password_reset_requests`	`hermes_db_server`	The mailbox row group
`cn=<user>,ou=users,dc=hermes,dc=local`	`hermes_ldap`	Per-mailbox LDAP entry (with `userPassword` Argon2id hash for local-auth or `seeAlso` for remote)

Path	Owner	Role
`cn=mailboxes`, `cn=one_factor` / `cn=two_factor`, `cn=nextcloud` in `ou=groups`	`hermes_ldap`	Group memberships set at create-time
`/mnt/vmail/<domain>/<user>/`	`hermes_dovecot` (mounted)	Mailbox directory tree — auto-created on first LMTP delivery / IMAP login; NOT removed on delete
Authelia `totp_configurations` + `webauthn_devices`	`hermes_authelia` storage backend	Cleared on delete + Reset 2FA Devices
`hermes_nextcloud` container	—	`occ user:add` / `user:delete` / `user:resetpassword` / `group:add` (the latter from Domains)

Every shell-out uses `docker exec ...` per the standard Hermes pattern.

Related

- Domains — mailbox-domain registration. A mailbox is meaningless without a domain row of `type='mailbox'`. Domain defaults (default quota, Nextcloud enabled, 2FA required) pre-fill the Add Mailbox form for new mailboxes; toggling the per-domain default does NOT cascade to existing mailboxes.
- Settings — global Dovecot config: TLS profile, compression, encryption at rest, quota warning thresholds. The warning thresholds measure against the per-mailbox quota set here.
- Aliases — alias addresses that resolve to mailboxes (with optional silent-discard mode). Add aliases AFTER the target mailbox exists.
- Shared Mailboxes — shared-namespace mailboxes with per-user ACLs. Distinct from regular mailboxes — they live in the same `mailboxes` table but with `mailbox_type='shared'` .
- Mailbox Rules — server-side Sieve rules per mailbox. Sieve is always-on at the protocol level via Settings.
- SAN Management — SAN prefixes that gate client auto-discovery for every mailbox domain.
- Authentication Settings — Authelia config, OIDC, the four-credential architecture (web vs IMAP/SMTP vs DAV vs Nextcloud) that mailbox app passwords slot into.
- LDAP RemoteAuth — required prerequisite for `auth_type='remote'` mailboxes. The Add form surfaces only mappings with `enabled=1`.
- Password Resets — admin-driven password reset for local-auth mailboxes (the user-facing flow uses the link in the welcome email).

- System Users — distinct from mailboxes; covers console admins / readers, which use the `system_users` table rather than `mailboxes`.
- Email Relay > Relay Recipients — the relay-topology equivalent. Mailbox users are delivered locally; relay recipients are forwarded downstream. Don't confuse the two.
- Organizational Signatures *(Pro)* — consumer of the Personal Information fields on the Edit Options modal (plus the domain's Organization Information fields).

SAN Management

SAN Management

Admin path: **Email Server > SAN Management** (`view_mailbox_sans.cfm`, `inc/san_actions.cfm`, `inc/sync_mailbox_sans.cfm`, `inc/acme_request_san_certificate.cfm`, `inc/smtp_sni_generate_config.cfm`, `inc/generate_nginx_configuration.cfm`, `schedule/acme_validate_ip.cfm`).

This page maintains the **global list of SAN (Subject Alternative Name) prefixes** that Hermes cross-joins with every mailbox-hosting domain to produce the actual SANs on each domain's TLS certificate. The prefix `mail` plus the domain `example.com` produces the SAN `mail.example.com`; doing it once here lets Hermes mint one certificate per mailbox domain that covers IMAP/POP/Submission, autoconfig/autodiscover, ManageSieve, CalDAV/CardDAV, and any additional client-facing hostnames in a single cert.

Pairs tightly with System Certificates (the certificate store these SANs are stamped into) and Domains (the mailbox-domain rows the prefixes are multiplied against). This page is the **only** input UI for the mailbox-cert SAN list — both the CSR generator on System Certificates and the ACME SAN request path read from `additional_sans` to build the `-d` flag list.

What the page edits

```
additional_sans                          domains (type='mailbox')
+----+---------------+--------+          +----+----------------+
| id | san           | system |          | id | domain         |
+----+---------------+--------+          +----+----------------+
|  1 | autoconfig    |   1    |          |  9 | example.com    |
|  2 | autodiscover  |   1    |          | 10 | acme.org       |
|  3 | mail          |   2    |          +----+----------------+
|  4 | imap          |   2    |
+----+---------------+--------+
              |                                    |
              +--- sync_mailbox_sans.cfm cross-joins ---+
                                   |
                                   v
```

```
mailbox_sans  (one row per prefix x domain)
+----+-------------+-------------------------+------+------+------+
| id | certificate | subdomain               | ip   | dns  | acme |
+----+-------------+-------------------------+------+------+------+
| 50 |     12      | autoconfig.example.com  | YES  | YES  |  1   |
| 51 |     12      | autodiscover.example.com| YES  | YES  |  1   |
| 52 |     12      | mail.example.com        | YES  | YES  |  1   |
| 53 |     12      | imap.example.com        | NO   | NO   |  1   |
| 54 |     12      | autoconfig.acme.org     | YES  | YES  |  1   |
| ...
```

Two storage rows per change:

Table	Role
`additional_sans`	One row per global prefix. `san` is the subdomain label; `system` is `1` for installer-seeded prefixes (`autoconfig`, `autodiscover`) that cannot be deleted, `2` for admin-added prefixes. There is no `enabled` flag — the row's mere presence means active.
`mailbox_sans`	One row per `additional_sans.san` x `domains` (`type='mailbox'`) combination. Carries the cert FK (`certificate`), the full FQDN (`subdomain`), and the per-SAN validation state (`ip` / `dns` = `YES` / `NO`, plus `*_result_datetime`, `*_result_msg`). `acme = 1` for ACME-managed certs, `2` for imported certs.

The page itself only writes to `additional_sans`. The cross-join into `mailbox_sans` is performed by `sync_mailbox_sans.cfm`, which is also called from the Domains page on add/edit (so adding a new mailbox domain populates its SAN rows immediately).

How a prefix becomes a live SAN

```
form submit (Add SAN Prefix)  ──► san_actions.cfm
                                      |
                                      |  validate:
                                      |    - prefix not blank
                                      |    - matches ^[a-z][a-z0-9-]{0,62}$
                                      |      (DNS label rules: lowercase, starts
                                      |       with letter, <= 63 chars)
                                      |    - not already in additional_sans
                                      |
```

```
                        |  INSERT additional_sans (san, system=2)
                        |
                        v
              sync_mailbox_sans.cfm
                  |
                  |  for each (prefix x mailbox-domain):
                  |     if FQDN missing in mailbox_sans:
                  |        INSERT (cert from mailbox_domains,
                  |                subdomain=fqdn, ip='NO', dns='NO',
                  |                acme=1|2 per cert type)
                  |     if FQDN exists with wrong cert binding:
                  |        UPDATE certificate + acme
                  |        (PRESERVE ip/dns validation state —
                  |         resetting would break nginx vhost
                  |         generation until the next validator
                  |         pass)
                  |  for each existing mailbox_sans row whose
                  |     subdomain is no longer in the cross-join:
                  |        DELETE
                  |
                  v
          Validator picks up the new rows on its next pass
          (schedule/acme_validate_ip.cfm @every 1h)
                  |
                  |  POST encrypted subdomain to
                  |    https://verify.hermesseg.io
                  |    -> returns expected IP for the host
                  |  Compare against the SAN's resolved A record
                  |    -> ip = YES/NO with timestamped result_msg
                  |  Resolve DNS for the SAN's CNAME/A chain
                  |    -> dns = YES/NO with timestamped result_msg
                  |
                  v
    All SANs on a cert at dns=YES + ip=YES?
                  |
                  v
acme_request_san_certificate.cfm (Pro)
docker run --rm certbot/certbot:latest \
  certonly --webroot --cert-name <domain> --expand \
    -d example.com -d autoconfig.example.com \
```

```
                -d autodiscover.example.com -d mail.example.com ...
                           |
                           v
            smtp_sni_generate_config.cfm   (Postfix SNI map)
            generate_nginx_configuration.cfm (per-SAN nginx vhosts)
```

Delete reverses the same path: removing a prefix from `additional_sans` calls `sync_mailbox_sans.cfm`, which deletes the corresponding `mailbox_sans` rows for every mailbox domain. The certificate itself is **not** re-issued automatically on delete — the next renewal cycle picks up the smaller SAN set when it runs.

The two seed prefixes

A fresh install seeds two `system = 1` rows:

Prefix	Required for
`autoconfig`	Thunderbird and K-9 Mail auto-configuration. Clients fetch `https://autoconfig.<domain>/mail/config-v1.1.xml`.
`autodiscover`	Outlook and iOS Mail auto-configuration. Clients POST to `https://autodiscover.<domain>/autodiscover/autodiscover.xml`.

Both rows have **Delete** suppressed and the System badge displayed. The action handler re-checks `system = 1` server-side and refuses with error 13 if a crafted POST tries to bypass the missing button. Removing either prefix would break client auto-discovery globally across every mailbox domain — they are non-optional.

Prefix validation rules

The Add form enforces DNS-label syntax both client-side (`pattern="[a-z][a-z0-9-]*"` + `maxlength="63"`) and server-side (`REFind("^[a-z][a-z0-9-]{0,62}$", ...)`):

- **Lowercase letters, numbers, and hyphens only.** No uppercase, no underscores, no dots. Each prefix is a **single** DNS label; multi-label SANs (`internal.mail.example.com`) are not supported here.
- **Must start with a letter.** Leading digits and leading hyphens are rejected per the DNS label spec.
- **Max 63 characters.** Each DNS label is capped at 63 octets.
- **Lowercased on save.** Submitting `Mail` stores as `mail`.

Suggested prefixes from the placeholder text: `mail`, `imap`, `smtp`, `pop`, `webmail`. Pick whichever match the client-facing hostnames you've published in DNS; the prefix only does work if a matching DNS A/CNAME record exists pointing at this server.

The Let's Encrypt budget callout

The page surfaces a live calculation of the cert budget per domain:

```
Let's Encrypt SAN limit: Each domain certificate supports a maximum
of 100 SANs. With <N> prefixes configured, each domain's certificate
uses <N + 1> SANs (1 for the domain + N prefixes), leaving room for
up to <99 - N> additional prefixes.
```

The +1 accounts for the bare domain itself, which is always included on the cert regardless of prefix list (this is hardcoded in the ACME request path).

Other Let's Encrypt rate limits that don't show on this page but still apply:

Limit	Value
SANs per certificate	100
Certificates per registered domain per week	50
Duplicate certificates per week	5
Failed validation attempts per account, per hostname, per hour	5

A misconfigured DNS record (SAN row stuck at `dns = NO`) does **not** burn the duplicate-cert budget because the certbot run is gated on the validator marking every SAN ready first. The validator's failed DNS probes are free and run on Hermes-side resolvers, not Let's Encrypt's.

Validation challenge mechanics

ACME issuance uses **HTTP-01** by default. The certbot container mounts `<repo>/config/hermes/var/www/html` at `/var/www/certbot` so the challenge file lands where the live nginx vhost for the domain already serves `/.well-known/acme-challenge/`. The domain's nginx vhost (generated by `generate_nginx_configuration.cfm`) is therefore required to be up and serving HTTP on port 80 of the public IP that the SAN resolves to.

DNS-01 (TXT-record validation) is **not** wired into this UI. The underlying certbot container supports it but the request path here hardcodes `--webroot`. Internal-only / DNS-only SANs (subdomains that

resolve to an internal IP but should still be on the public cert) need either a manual certbot invocation or a public split-DNS record pointing at the gateway's WAN address — there is no DNS-challenge bypass on this page.

The validator's `ip = YES` check is **separate from** the ACME challenge — it confirms that the SAN's DNS A record points at this gateway's expected IP (which is what `https://verify.hermesseg.io` returns when probed). It exists to catch broken DNS before burning a Let's Encrypt rate-limit slot, not to perform the ACME challenge itself.

How SAN status surfaces elsewhere

This page edits the prefix list; the per-SAN validation state and the per-cert SAN sub-table show up on other pages:

Where	What it shows
Domains **Cert Status** column	Per-domain aggregate: `Verified` (all SANs ip+dns=YES), `Partial`, `Awaiting Cert`, `Pending`, `DNS Failed`, `No SANs`, `No Cert`. Imported certs always render `Imported` regardless of probe state because probes are informational only for those.
System Certificates expanded row **Mailbox SAN Validation** sub-table	Per-cert listing: every SAN bound to the cert, with its `ip_result_msg` / `dns_result_msg` / timestamps. Read-only here.
System Certificates § Generate CSR — Mailbox certificate purpose	The CSR generator pre-fills the SAN list from `additional_sans` x the chosen mailbox domain. Refuses to generate a mailbox CSR if `additional_sans` is empty (impossible in practice because the two system prefixes can't be deleted).
`smtp_sni_generate_config.cfm` (run from Email Server > Settings)	Reads `mailbox_sans WHERE dns = 'YES'`, builds Postfix's `sni_maps`, runs `postmap -F`. Postfix then serves the per-domain cert on `:25` / `:587` via SNI based on the client's TLS SNI extension.
`generate_nginx_configuration.cfm` (run from Domains)	Reads validated `mailbox_sans` rows to write per-SAN nginx `server` blocks (autoconfig, autodiscover, DAV).

Failure semantics

What breaks	What happens

Prefix blank	`session.m = 10`, redirect, no DB write
Prefix fails DNS-label regex	`session.m = 11`, redirect, no DB write
Prefix already in `additional_sans`	`session.m = 12`, redirect, no DB write
Delete attempted on a `system = 1` prefix	`session.m = 13`, redirect, no DB write
Delete with non-numeric `delete_san_id`	`session.m = 20`, redirect
`sync_mailbox_sans.cfm` fails mid-cross-join	Partial `mailbox_sans` state possible; re-saving any mailbox domain or re-adding the same prefix triggers another sync that converges
Validator can't reach `verify.hermesseg.io`	`mailbox_sans.ip` stays at the previous value; cert request gated until next successful probe. Validator runs hourly.
`acme_request_san_certificate.cfm` fails (DNS, port 80, rate limit)	Postmaster email sent with certbot stderr; SAN rows retain validation state; admin can re-trigger by toggling the cert binding on Domains
`smtp_sni_generate_config.cfm` finds zero validated SANs	Deletes `/etc/postfix/sni_maps` and `.db` — Postfix falls back to its default cert on every connection. Non-fatal but clients lose per-domain SNI.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_mailbox_sans.cfm`	`hermes_commandbox`	Page + Add card + Delete modal + LE budget callout
`config/hermes/var/www/html/admin/2/inc/san_actions.cfm`	`hermes_commandbox`	Add / Delete handler — validates, writes `additional_sans`, calls sync
`config/hermes/var/www/html/admin/2/inc/sync_mailbox_sans.cfm`	`hermes_commandbox`	Cross-joins prefixes x mailbox domains into `mailbox_sans`; idempotent
`config/hermes/var/www/html/admin/2/inc/acme_request_san_certificate.cfm`	`hermes_commandbox`	Pro — runs ephemeral certbot container for SAN-bearing certs
`config/hermes/var/www/html/admin/2/inc/smtp_sni_generate_config.cfm`	`hermes_commandbox`	Pro — builds Postfix `sni_maps` from validated SANs
`config/hermes/var/www/html/admin/2/inc/generate_nginx_configuration.cfm`	`hermes_commandbox`	Per-domain nginx vhost generator (called from Domains; consumes validated SANs)
`config/hermes/var/www/html/schedule/acme_validate_ip.cfm`	`hermes_commandbox` (Ofelia)	Pro — hourly validator; probes each SAN's IP via `verify.hermesseg.io` and updates `mailbox_sans.ip` / `dns`
`additional_sans` table	`hermes_db_server` (`hermes` DB)	The prefix list this page edits

Path	Owner	Role
`mailbox_sans` table	`hermes_db_server` (`hermes` DB)	Per-SAN rows with validation state and cert binding
`system_certificates` table	`hermes_db_server` (`hermes` DB)	Per-cert metadata referenced via `mailbox_sans.certificate`
`/etc/letsencrypt/live/<domain>/`	`hermes_commandbox` (bind-mounted from `config/certbot/conf/`)	Issued SAN certs
`/etc/postfix/sni_maps` + `.db`	`hermes_postfix_dkim` (mounted)	Live SNI map — Postfix serves per-domain cert based on this
`/etc/postfix/sni/*.pem`	`hermes_postfix_dkim` (mounted)	Combined key + fullchain PEM per cert, referenced from `sni_maps`
Per-SAN nginx vhost files	`hermes_nginx` (mounted)	One vhost per validated SAN
`certbot/certbot:latest` image	docker.io	Pulled on demand for SAN cert issuance + renewal
`verify.hermesseg.io`	external (Pro)	Returns expected IP for a given SAN to gate ACME issuance

Every certbot invocation is `docker run --rm` against the public `certbot/certbot:latest` image — same pattern as the single-domain ACME path on System Certificates. The container shares the host network (`--network host`) so the HTTP-01 challenge can reach port 80 on the public IP.

Related

- System Certificates — the certificate store these SANs land on. The Mailbox certificate purpose on Generate CSR auto-fills its SAN list from this page; Pro's auto-managed ACME path mints SAN certs from the same source.
- Domains — per-mailbox-domain Cert Status column summarizes the per-SAN validation state this page's prefixes drive. Adding a domain calls `sync_mailbox_sans.cfm`, so new SANs appear immediately under existing prefixes.
- Mailboxes — mailbox users hit IMAP/Submission via the `imap` / `mail` / `smtp` prefixes configured here. Apple iOS and Outlook reach autodiscover via the system prefixes.
- Settings — Dovecot IMAP/POP TLS is gated on the validated mailbox cert; the SNI map for Postfix is generated from the same `mailbox_sans` table this page populates.
- Aliases / Shared Mailboxes — both ride on the same per-domain cert; no separate SAN entries needed.
- SMTP TLS Settings — binds the **single** cert Postfix presents on the public SMTP banner. The SNI map this page feeds into is an **additional** layer that overrides the banner cert when the client sends a matching SNI hostname.

- Email Relay > Relay Recipients — relay recipients use Submission via the same `mail.<domain>` hostnames as local mailboxes; the SAN prefixes here cover both topologies.

Settings

Settings

Admin path: **Email Server > Settings** (`view_email_server_settings.cfm`, `inc/email_server_settings_action.cfm`, `inc/generate_dovecot_configuration.cfm`, `inc/generate_mail_crypt_keys.cfm`).

This page is the **global configuration surface for the Email Server topology** — the half of Hermes where Hermes is itself the destination MTA, delivering inbound mail into Dovecot mailboxes on `/mnt/vmail` and serving IMAP/POP3/Submission/Sieve back to end users. Per-domain addressing lives on Email Server > Domains, per-mailbox quotas and personal info on Mailboxes, and aliases on Aliases; this page handles everything that applies across all mailboxes regardless of domain — the Dovecot TLS profile, mail compression and encryption-at-rest, which protocols are exposed, quota warning thresholds, connection limits, debug logging, the Nextcloud login-form mode that gates webmail SSO, and the master toggle for shared mailboxes and folder sharing.

Most pages save and run a small handful of `docker exec` commands. This page saves and re-renders the entire Dovecot configuration from a template; the next inbound LMTP delivery sees the new settings.

What this page does — and what it doesn't

This page configures	This page does NOT configure
Dovecot TLS certificate, profile, ciphers, min protocol	LDAP authentication backend (hard-coded against `hermes_ldap`)
Mail compression (LZ4 / Zstd / Zlib)	Per-mailbox quota size (set on Mailboxes)
Mail encryption at rest (mail_crypt plugin + ECC key pair)	Per-domain delivery / acceptance (handled by Domains)
IMAP and POP3 enable/disable	Submission, Sieve, LMTP enable (always on — required for core operation)

This page configures	This page does NOT configure
Quota warning thresholds (medium / high / critical / trash overage)	Default new-mailbox size (set per-mailbox; see Mailboxes)
Per-service client limit + per-user-per-IP connection cap	Postfix-side recipient validation (handled by Postfix `relay_recipient_maps`)
Dovecot debug logging	Authelia session timing, MFA enrollment, SMTP notifier (Authentication Settings)
Mailbox sharing master toggle (Shared/ namespace + user folder shares)	Per-user shared mailbox access (handled by Shared Mailboxes)
Nextcloud login form mode (auto-redirect / SSO-only / full form)	Nextcloud OIDC client itself (Authentication Settings)

Configuration storage

Almost every setting on this page is keyed into `parameters2` under `module = 'dovecot'` and read back by both the page and `generate_dovecot_configuration.cfm` at render time. A handful of adjacent concerns live in sibling modules:

Settings group	Storage
All Dovecot directives (compression, encryption, protocols, quota, connections, logging, sharing, TLS profile/ciphers)	`parameters2` rows where `module = 'dovecot'`, keyed by dotted names like `mail.compression_algorithm`, `quota.warning_critical`, `ssl.min_protocol`
TLS certificate selection	`parameters2` row `module = 'certificates'`, `parameter = 'mail.certificate'`, value = `system_certificates.id`
Nextcloud login-form mode	`parameters2` row `module = 'nextcloud'`, `parameter = 'oidc.auto_redirect'`, value = `auto_redirect` / `sso_only` / `full_form` (legacy `true` / `false` strings normalized on read)
Mail encryption key pair	Files at `/opt/hermes/keys/ecprivkey.pem` and `/opt/hermes/keys/ecpubkey.pem` on the Docker host
Live Dovecot config	`/etc/dovecot/dovecot.conf` (regenerated from `/opt/hermes/templates/dovecot.conf` on every save)

`parameters2` is keyed by the `module + parameter` pair. The action handler uses an upsert pattern (`checkDovParam` → UPDATE-or-INSERT) so fresh installs that haven't yet had the schema seeded with every row land cleanly on first save.

How a save propagates

```
form submit  ──► email_server_settings_action.cfm
                         │
                         │  1. validate + sanitize (whitelist enums,
                         │     clamp numeric ranges, normalize booleans)
                         │
                         │  2. Nextcloud login-form mode
                         │     - UPDATE/INSERT parameters2 (oidc.auto_redirect)
                         │     - docker exec hermes_nextcloud occ
                         │         config:app:set user_oidc
                         │         allow_multiple_user_backends = 0|1
                         │     - docker exec hermes_nextcloud occ
                         │         config:system:set/delete hide_login_form
                         │
                         │  3. Dovecot TLS cert
                         │     - verify system_certificates row exists
                         │     - UPDATE/INSERT parameters2 (mail.certificate)
                         │
                         │  4. Mail encryption key generation (if enabled
                         │     AND keys missing OR zero-byte)
                         │     - cfinclude generate_mail_crypt_keys.cfm
                         │     - openssl ecparam + ec via docker exec
                         │     - writes /opt/hermes/keys/ecprivkey.pem
                         │              /opt/hermes/keys/ecpubkey.pem
                         │
                         │  5. Dovecot settings batch upsert
                         │     - loop the dovSettings struct
                         │     - UPDATE-or-INSERT each parameters2 row
                         │
                         │  6. cfinclude generate_dovecot_configuration.cfm
                         │     - reads /opt/hermes/templates/dovecot.conf
                         │     - substitutes placeholders from parameters2
                         │     - writes /etc/dovecot/dovecot.conf
                         │     - docker exec hermes_dovecot dovecot reload
                         │
                         v
              cflocation → session.m = 1 (success) or 10 (per-step errors)
```

Validation lives entirely in the action handler. Each step is wrapped in its own `cftry` so a failure in (e.g.) the Nextcloud `occ` step accumulates into `session.saveErrors` but doesn't abort the Dovecot save. Step 6 — the Dovecot regen — gates on `NOT saveError` so a broken upstream step doesn't

push a half-rendered config file.

Cards on the page

Nextcloud Webmail Settings

Single dropdown that controls the Nextcloud login page behavior. Three modes — chosen because two underlying Nextcloud knobs (`user_oidc.allow_multiple_user_backends` and the system-wide `hide_login_form`) compose into three meaningful states:

Mode	`allow_multiple_user_backends`	`hide_login_form`	User experience
Auto-redirect to SSO (default)	`0`	(unset)	Clicking "Login to Webmail" silently bounces through Authelia OIDC and lands the user in Nextcloud already authenticated. True SSO — no Nextcloud login page is ever shown.
SSO button only	`1`	`true`	The Nextcloud login page is shown but with the username/password fields hidden — only the SSO button is visible. Good when you want users to know SSO is required but don't want to auto-redirect.
Show full form	`1`	(unset)	Both the username/password form and the SSO button are shown. Use temporarily for local Nextcloud admin maintenance.

The legacy storage key `oidc.auto_redirect` is reused as the slot for this three-way value so existing installs don't need a migration. The read path in `view_email_server_settings.cfm` normalizes legacy `true`/`false` strings to `auto_redirect` / `full_form`.

Nextcloud Maintenance Mode card

Below the Webmail Settings card sits a second card that controls the local-admin escape hatch. As of #262 there is **no permanent bypass URL** — the operator toggles OIDC on/off from this card

when they need to administer Nextcloud as the local admin (separate identity from the Authelia/LDAP users that normally SSO in).

State	What it means
`OIDC ENABLED` (green)	Normal operation. Mailbox users SSO into Nextcloud via Authelia. The local NC admin **cannot** log in.
`MAINTENANCE MODE` (yellow)	Click "Enter Maintenance Mode" ran `occ app:disable user_oidc`. Mailbox-user SSO is offline. The local NC admin can now log in via Nextcloud's own form at `/nc/`.

Maintenance procedure:

1. Click **Enter Maintenance Mode**. The card status flips to yellow, mailbox-user SSO goes offline, and a success banner appears at the top of the page.
2. Click the **Open Nextcloud** button that appears below the toggle — it opens `https://<console-host>/nc/` in a new tab (`target="_blank"`) so the Hermes admin tab stays put for step 7.
3. In the Nextcloud tab, log in as the NC local admin. Username is shown on the card; password is also in `/opt/hermes-seg-container-gl/INSTALL_SUMMARY.txt` on the host.
4. On first login Nextcloud prompts for TOTP enrollment via its own UI — scan the QR code with any TOTP authenticator app.
5. **First login only — generate backup codes immediately**. Click your avatar (top-right) → **Personal settings** → **Security**, scroll to **Two-Factor backup codes**, click **Generate backup codes**. Save the 10 single-use codes somewhere safe (password manager, printed copy in a safe, etc.). These codes are the ONLY recovery path if you lose your TOTP authenticator — without them, recovery requires shell access. Done once per admin; codes persist across sessions until used.
6. Do your admin work in Nextcloud.
7. Switch back to the Hermes admin tab and click **Exit Maintenance Mode**. SSO is restored for mailbox users.

The button uses `fetch()` to call `inc/edit_nc_oidc_action.cfm` (`occ app:disable user_oidc` or `enable`), bypassing the outer settings form so the toggle doesn't collide with a normal Save submission. `redirect: 'manual'` on the fetch prevents the action handler's `cflocation` from being auto-followed and consuming the `session.m` flash before the page can render it.

Operators who need to use this often can ignore step 2's helper link and just type `/nc/` — the helper link exists to make first-time use obvious.

Why the toggle pattern and not a permanent bypass URL:

Earlier attempts at a permanent local-admin URL (the `/nc-admin-login` path) were architecturally infeasible. The Authelia session created by gating that URL fueled `user_oidc` silent OIDC re-auth on every post-form `/nc/` request, overriding whatever local-admin session the form submission had just established. Removing the Authelia gate didn't help either because `user_oidc` itself force-redirects `/login?direct=1` to OIDC under several conditions. The toggle is the only path that

reliably wins against `user_oidc`, and it's what most NC operators in OIDC-fronted deployments use anyway. See #262 for the full diagnostic trace.

Recovery if the NC local admin loses their TOTP authenticator:

1. **Preferred — backup codes** (generated at TOTP enrollment time per step 5 of the maintenance procedure above). At the TOTP prompt during login, click **"Use backup code"** (or **"Try another method"**, wording varies by NC version), paste one of the saved codes. Each code is single-use, so re-generate a new set after recovery via Personal → Security → Two-Factor backup codes.
2. **Fallback — disable enforcement via shell** (only if backup codes are also lost or were never generated):

   ```
   docker exec hermes_nextcloud php occ twofactorauth:enforce --off
   # log in, re-enroll TOTP via NC UI, generate fresh backup codes, then:
   docker exec hermes_nextcloud php occ twofactorauth:enforce --on
   ```

   This requires shell access to the Hermes host. If you don't have shell access, the only recovery is restoring `/mnt/data/dbase/` from a backup taken when the admin still had TOTP access, which is a significantly more disruptive operation. Generating backup codes at enrollment time is much cheaper.

Mailbox Sharing

Single dropdown — Enabled or Disabled. Stored as `sharing.enabled` in `parameters2`.

State	Dovecot effect
Enabled	Shared mailbox support is compiled into the Dovecot config (`acl`, `imap_acl`, `imap_quota` plugins and the `Shared/` namespace). Per-mailbox shares are then managed under Shared Mailboxes. Folder-level user-managed shares work in IMAP clients that support them.
Disabled	The shared namespace is not declared in the Dovecot config and IMAP clients won't see a `Shared/` folder. Existing per-mailbox ACL entries are preserved in their backing files but are inactive until sharing is re-enabled.

Toggling this is the master switch. The per-mailbox setup work happens on Shared Mailboxes.

TLS / SSL Settings

The cert that Dovecot presents on every IMAPS / POP3S / submission connection. Driven by:

Field	Notes
Mail Server Certificate	Autocomplete against `system_certificates` (via `inc/getcertificates.cfm`). Selecting a row populates the four read-only fields below and writes the cert `id` into `parameters2`. Manage certificates on System Certificates.
TLS Security Profile	`Modern` (TLS 1.3 only) / `Intermediate` (TLS 1.2+, recommended) / `Legacy` (TLS 1.2+, broad compatibility) / `Custom`. Presets follow Mozilla Server Side TLS guidance.
Minimum TLS Version	Auto-set by profile (read-only) when a preset is selected; editable in Custom mode.
SSL Cipher List	Auto-set by profile (read-only) when a preset is selected; editable in Custom mode. The page's JS form-submit hook re-enables disabled fields before submit so their values are POSTed. The action handler's `cfswitch` then re-derives the canonical preset values defensively so the saved values always match the named profile.

`Intermediate` is the default and the only profile that ships with a non-empty cipher list. `Modern` deliberately leaves the cipher field empty because OpenSSL picks TLS 1.3 ciphers automatically.

Mail Storage — Compression

Field	Notes
Mail Compression	Enabled / Disabled. When Disabled, the algorithm and level fields are JS-disabled.
Algorithm	`LZ4` (fastest, good compression) / `Zstandard` (balanced) / `Zlib/Deflate` (best ratio, slowest). LZ4 is the default.
Compression Level	Numeric. Hidden for LZ4 (no level knob). 1–22 for Zstandard (default 3), 1–9 for Zlib (default 6). The handler enforces the Zlib ceiling — Zlib with level > 9 is clamped to 6.

Compression is mailbox-format aware: only newly delivered or saved messages are compressed, existing messages remain readable, and Dovecot auto-detects the format per message on read. Changing or disabling compression never breaks existing mail; mailboxes safely contain a mix of uncompressed, LZ4, and Zstandard messages.

Mail Storage — Encryption at Rest

Dovecot's `mail_crypt` plugin with an EC-curve key pair stored on the Docker host. **This is irreversible-ish — back up the keys.**

Field	Behavior

Encryption at Rest	Disabled (default) / Enabled. Saving with Enabled and no key pair triggers `generate_mail_crypt_keys.cfm`, which runs `openssl ecparam` + `openssl ec` via `docker exec hermes_dovecot` to write `/opt/hermes/keys/ecprivkey.pem` and `ecpubkey.pem`.
Elliptic Curve	`prime256v1` / `secp384r1` / `secp521r1`. Selectable only when no keys exist yet — once keys are generated the field is rendered as a read-only display because changing curves with mismatched keys would render existing encrypted mail unreadable.
Algorithm	Always `AES-256-GCM`. Not configurable.
Key Status	Badge: `Keys Present` (green), `Keys Empty` (red — files exist but zero-byte from a failed previous attempt; delete from the host to regenerate), or `No Keys` (gray — auto-generated on enable).

> **Operational consequence.** Only newly delivered mail is encrypted. Disabling encryption later does not affect existing encrypted messages — they remain readable as long as the keys are present. If the keys are lost there is no recovery mechanism; encrypted mail becomes permanently unreadable. The two PEM files belong in every system backup. The system-backup script collects `/opt/hermes/keys/` automatically, but operators running off-Hermes backup tooling must include this directory explicitly.

Protocols & Connections — Protocols

Per-protocol enable/disable for the end-user-facing services. **Submission, Sieve, and LMTP are always enabled** — Submission for authenticated outbound and vacation responder, Sieve for mail filter rules, LMTP for Postfix-to-Dovecot delivery — and surface in the UI as read-only `Always Enabled` fields.

Protocol	Ports	Knob
IMAP	993 / 143	`protocol.imap` — Enabled / Disabled
POP3	995 / 110	`protocol.pop3` — Enabled / Disabled
Submission	587	Always on
Sieve / LMTP	4190 / 24	Always on

Disabling IMAP or POP3 takes effect on the next Dovecot reload — the service is dropped from `protocols = ...` in `dovecot.conf` and the listener stops.

Protocols & Connections — Connection Limits

Field	Default	Notes
Login Service Client Limit	`1000`	Max concurrent connections per login service (IMAP, POP3, Submission, ManageSieve). Clamped 100–10000. Increase for installs with many simultaneous users.
Max Connections per User per IP	`20`	Per-user-per-source-IP cap. Stops a runaway client from consuming the global pool. Clamped 1–1000. Bump for users with many devices / many open folders.

Quota Settings — Warning Thresholds

When a mailbox crosses these usage thresholds, Dovecot's quota-warn hook sends an email notification. A "back under quota" notice is always sent when usage drops below 100% — that one is not configurable. **Per-mailbox quota sizes are set per-mailbox** on Mailboxes; this card only controls the warning bands.

Field	Default	Range
Critical Warning	`99` %	1–100. Triggers the "Mailbox Full" notification.
High Warning	`95` %	1–100. Triggers the "Nearly Full" notification.
Medium Warning	`80` %	1–100. Triggers the first warning notification.
Trash Quota Overage	`110` %	100–200. The Trash folder is allowed this percentage of the user's quota so users can still delete messages when they're at 100%. Default leaves 10% headroom in Trash.

Logging

Field	Notes

Debug Logging	Disabled (production, default) / Enabled (troubleshooting). When Enabled, Dovecot's `mail_debug = yes` and `auth_debug = yes` are emitted. Output lands in `/logs/dovecot-debug.log` inside the container. Significant log volume — leave off in production.

Failure semantics

What breaks	What happens
Nextcloud `occ` step fails (container down, OIDC app not installed)	Per-error message appended to `session.saveErrors`, banner shown at top of page, **other steps still run**
TLS cert id doesn't match a `system_certificates` row	`parameters2 mail.certificate` is not updated; Dovecot keeps using whatever cert was previously selected
`generate_mail_crypt_keys.cfm` fails	Per-error message appended; encryption may be enabled in DB but keys missing — admin sees the Keys Empty badge on the next page load, must clear the partial files and retry
Dovecot config regen fails (template missing, substitution error)	`session.m = 10`, error banner with the cfcatch message; **the previous `dovecot.conf` is still on disk** because the template renderer writes to a temp path and atomically moves only on success
`dovecot reload` fails	The new config is on disk but the running Dovecot is still on the old config. Recovery is `docker exec hermes_dovecot dovecot reload` from the host or a container restart.
Encryption keys deleted from host while encryption is enabled	New incoming mail cannot be encrypted; Dovecot logs the failure and the LMTP delivery is deferred. Existing encrypted mail remains unreadable until the keys are restored from backup.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_email_server_settings.cfm`	`hermes_commandbox`	Page + cards
`config/hermes/var/www/html/admin/2/inc/email_server_settings_action.cfm`	`hermes_commandbox`	Save handler
`config/hermes/var/www/html/admin/2/inc/generate_dovecot_configuration.cfm`	`hermes_commandbox`	Template-to-`dovecot.conf` renderer + `dovecot reload`
`config/hermes/var/www/html/admin/2/inc/generate_mail_crypt_keys.cfm`	`hermes_commandbox`	EC key pair generator

Path	Owner	Role
`config/hermes/var/www/html/admin/2/inc/getcertificates.cfm`	`hermes_commandbox`	Autocomplete for the Mail Server Certificate field
`/opt/hermes/templates/dovecot.conf`	`hermes_commandbox`	Dovecot template
`/etc/dovecot/dovecot.conf`	`hermes_dovecot` (volume-mounted)	Live Dovecot config (regen target)
`/opt/hermes/keys/ecprivkey.pem`, `ecpubkey.pem`	`hermes_dovecot` (volume-mounted)	mail_crypt key pair
`parameters2` rows where `module IN ('dovecot','certificates','nextcloud')`	`hermes_db_server`	Settings storage
`system_certificates`	`hermes_db_server`	TLS certificate lookup
`hermes_nextcloud` container	—	`occ config:app:set user_oidc allow_multiple_user_backends`, `occ config:system:set/delete hide_login_form`

Every shell-out uses `docker exec hermes_dovecot ...` or `docker exec hermes_nextcloud ...` per the standard Hermes pattern.

Related

- Domains — per-domain configuration for the mailbox topology. Add a domain there first; this page's settings then apply to every mailbox on every domain.
- Mailboxes — per-mailbox quota size, personal info, encryption opt-in. The quota size set per-mailbox is what the warning thresholds on this page measure against.
- Aliases — alias addresses that resolve to local mailboxes. The Email Server alternative to Email Relay > Virtual Recipients.
- Shared Mailboxes — per-mailbox shared-access configuration. The master switch on this page must be on for any shared mailbox to function.
- Mailbox Rules — server-side Sieve rules per mailbox; Sieve is always-on at the protocol level via this page.
- SAN Management — Subject Alternative Names on the Dovecot TLS certificate. The cert selected on this page is the one SAN Management edits.
- System Certificates — managing the certificate inventory that the Mail Server Certificate autocomplete draws from.
- Authentication Settings — Authelia, the OIDC client, and the Nextcloud-side session-lifetime knobs that complement the login-form mode dropdown on this page.

Shared Mailboxes

Shared Mailboxes

Admin path: **Email Server > Shared Mailboxes** (`view_shared_mailboxes.cfm`, `inc/shared_mailbox_actions.cfm`, `inc/sync_shared_mailbox_acl_file.cfm`, `inc/sync_user_folder_acl_file.cfm`, `inc/get_shared_mailbox_permissions_json.cfm`).

This page manages **mailboxes that several users can read from and write to** — typically role addresses like `info@`, `support@`, or `sales@`. A shared mailbox is a real Dovecot mailbox in its own Maildir, but it has no login of its own; users access it through their own credentials and the rights granted on this page. The **master switch** for the entire shared-mailbox feature lives on Email Server > Settings (Mailbox Sharing card) — when that switch is off, the rows on this page are preserved but inactive, and the Add / Manage Permissions / Rebuild buttons are disabled.

Per-member rights are stored in the `shared_mailbox_permissions` table and projected to Dovecot's on-disk `dovecot-acl` files via the vfile driver, which is the only per-mailbox ACL driver shipped with Dovecot 2.4 (the SQL rights driver was a non-upstream Hermes carry that was removed in the 2.4 rewrite).

How a shared mailbox is wired

A shared mailbox is more than just an ACL — six tables and a Maildir are stitched together on creation:

Component	Storage	Role
Mailbox row	`mailboxes` with `mailbox_type = 'shared'`	Gives Dovecot a userdb entry so the mailbox has a quota, a Maildir, and a sender identity
Shared mailbox row	`shared_mailboxes`	UI metadata: address, display name, auto-subscribe flag, owning domain
Per-member rights	`shared_mailbox_permissions`	Authoritative permission matrix per (shared mailbox, user mailbox) pair
On-disk ACL	`/srv/mail/<domain>/<local>/dovecot-acl`	Dovecot vfile driver enforcement file — projected from `shared_mailbox_permissions`

Component	Storage	Role
Shared namespace visibility	`dovecot_acl_shared` (`acl_sharing_map`)	Tells Dovecot's `Shared/` namespace which users should see this mailbox in their folder list
Recipient policy	`recipients` (Amavis SVF policy + `recipient_type = 'shared'`)	Allows mail addressed to the shared address to pass the Amavis recipient gate
Sender identity	`sender_login_maps`	Lets the shared address be used as a From: by itself (anchor row) and by each member with Send-As granted
Maildir	`/srv/mail/<domain>/<local>/`	The actual on-disk message store. Bootstrapped via `doveadm mailbox create -u <addr> INBOX` so members see it immediately rather than waiting for first delivery

The add handler creates all of these in a single `cftry` block. If any step fails the catch sets `session.m = 30` and the operation fails-loud rather than leaving a partial mailbox.

Permission model — seven flags, projected to IMAP ACL letters

The UI surfaces seven permission flags. Six are IMAP ACL rights enforced by Dovecot; one (Send-As) is a Postfix sender-identity grant.

UI flag	DB column	Dovecot vfile rights	IMAP ACL meaning
Read	`can_read`	`lrs`	`lookup` (see mailbox), `read` (read messages), `write-seen` (set/clear \Seen flag)
Write	`can_write`	`wt`	`write` (set/clear flags except \Seen and \Deleted), `write-deleted` (set/clear \Deleted)
Delete	`can_delete`	`e`	`expunge` (permanently remove messages)
Insert	`can_insert`	`i`	`insert` (append/copy messages into mailbox)
Post	`can_post`	`p`	`post` (submit messages via the post address — rarely used)

UI flag	DB column	Dovecot vfile rights	IMAP ACL meaning
Admin	`can_admin`	`a`	`admin` (modify the ACL itself from an IMAP client)
Send-As	`send_as`	—	Inserts `(sender = shared, login_user = member)` into `sender_login_maps` so the member can use the shared address as From:

The vfile letters are concatenated into a single token per user (e.g., `lrswtie` for read+write+delete+insert). Dovecot 2.4's vfile parser reads each character as a separate right, so the full-word form (`lookup read write-seen ...`) does NOT work — the parser would treat `o` in `lookup` as an unknown right. The `sync_shared_mailbox_acl_file.cfm` include knows this and emits the single-letter form.

The `dovecot_acl` SQL table is still written by the action handlers for legacy/audit reasons, but Dovecot 2.4 no longer reads it. `sync_shared_mailbox_acl_file.cfm` writes the on-disk file every time permissions change, and the **Rebuild ACL Files** button on the page regenerates every file from scratch — used after upgrading to a new Dovecot release or when an admin reports a member can't see a mailbox they should have rights on.

How a save propagates

```
Add Shared Mailbox  ──► shared_mailbox_actions.cfm (add_shared_mailbox)
                                │
                                │  1. Feature guard (Mailbox Sharing = enabled)
                                │  2. Validate prefix + domain + display name + quota
                                │  3. Four-way conflict check
                                │     (recipients, mailboxes, mailbox_aliases,
                                │      virtual_recipients)
                                │  4. INSERT into recipients (Amavis SVF policy)
                                │     + maddr (Amavis address tracking)
                                │  5. INSERT into mailboxes (mailbox_type='shared')
                                │  6. INSERT into shared_mailboxes
                                │  7. INSERT into sender_login_maps (anchor row)
                                │  8. docker exec hermes_dovecot doveadm mailbox
                                │     create -u <addr> INBOX  (bootstrap Maildir)
                                │  9. For each initial member:
                                │     - INSERT shared_mailbox_permissions
                                │     - INSERT dovecot_acl (legacy)
                                │     - INSERT dovecot_acl_shared (namespace)
```

```
                        |     - INSERT sender_login_maps if Send-As
                        | 10. cfinclude sync_shared_mailbox_acl_file.cfm
                        |     → writes /srv/mail/<dom>/<local>/dovecot-acl
                        |       via temp shell script + docker exec -i
                        |       (heredoc pattern; vmail:vmail 0660)
                        v
                cflocation → session.m = 1
```

Add / Edit / Remove permission flows follow the same shape but only touch the rows for one member, then re-call `sync_shared_mailbox_acl_file.cfm` to rebuild that mailbox's `dovecot-acl` file in place. The sync include uses the **temp shell script + heredoc + `docker exec -i`** pattern (it has to — Lucee `cfexecute` argument quoting can't reliably ship multiline content with embedded special characters through `docker exec`).

Cards and modals on the page

Add Shared Mailbox modal

Field	Notes
Domain	Dropdown of mailbox-type domains (`domains.type = 'mailbox'`). The Address Prefix suffix updates live to show the full address.
Address Prefix	Local-part of the email. Validated against `^[a-z0-9._-]+$` — only lowercase letters, digits, dots, hyphens, underscores.
Display Name	Free-form text shown as the mailbox's `name` and in the table. Required.
Quota (GB)	Mailbox quota. Accepts decimals (e.g., `0.5`). Stored as bytes via `Round(quota_gb * 1024^3)`.
Auto-Subscribe	When `Yes` (default), the shared mailbox appears automatically in each member's IMAP folder list. When `No`, members have to manually subscribe to `Shared/<address>` in their client.
Initial Members	Checkbox list of user mailboxes in the selected domain (filtered live as the Domain dropdown changes). Optional — you can grant access later.
Default Permissions	Seven checkboxes applied uniformly to every selected initial member. Defaults are Read + Write + Insert checked.

The address-prefix suffix and the member-list filter both run client-side when the Domain dropdown changes. Cross-domain members are excluded from the picker even before form submit; the server-side handler re-enforces the same-domain rule with error 26 if a forged post tries to bypass it.

Shared Mailboxes table

DataTables surface — searchable, sortable, paginated, `stateSave: true`.

Column	Source
Actions	Manage Permissions (opens modal) / Delete (opens confirmation modal)
Address	`shared_mailboxes.address`
Display Name	`shared_mailboxes.display_name`
Domain	`domains.domain`
Members	Count of `shared_mailbox_permissions` rows for this shared mailbox
Quota	`mailboxes.quota` divided into GB (1-decimal for whole GB, 2-decimal otherwise)
Auto-Subscribe	YES / NO badge
Status	`Active` (sharing on + mailbox active) / `Inactive` (sharing on + mailbox disabled) / `Inactive (Sharing Off)` (master switch off)

A Domain filter dropdown narrows the visible rows to one domain.

Manage Permissions modal

Opens via the per-row action button. Two sections:

1. **Current Members** — table of every `shared_mailbox_permissions` row for this shared mailbox, with per-right YES/NO badges and Edit / Remove buttons per row. Loaded via AJAX from `get_shared_mailbox_permissions_json.cfm`.
2. **Add Member** — Tom Select user picker (filtered to the same domain as the shared mailbox) + the seven permission checkboxes
   - an Add button.

The Edit Member sub-modal opens on top of the Manage Permissions modal, lets you toggle the seven flags for an existing member, and re-syncs the on-disk ACL file on save. Changes take effect immediately; the member does not need to reconnect their mail client.

Rebuild ACL Files modal

A maintenance action that walks **both** admin-managed shared mailboxes AND user-managed folder shares and regenerates every `dovecot-acl` file from the current state of the database.

> **When to use Rebuild ACL Files.**
>
> - After upgrading to a new Dovecot 2.4 release — backfills the vfile files for any shared mailboxes created before the upgrade.
> - When a member reports they cannot see or access a shared mailbox or shared folder they should have rights on (recovery / drift heal).
> - After manually editing `shared_mailbox_permissions` or `user_folder_shares` in the database.
>
> Safe to run anytime — it rebuilds files from the database and never modifies the permission rows themselves. Per-mailbox failures are non-fatal; the operation continues to the next.

The success banner reports a count of shared mailboxes rebuilt and a separate count of user folder shares rebuilt, so the admin can confirm the operation covered everything they expected.

Delete Shared Mailbox modal

A confirmation modal that lists exactly what will be removed:

- All member permissions and ACL entries
- Sender login maps (send-as permissions)
- Dovecot shared folder subscriptions
- Amavis policy entry

With an optional **Also delete all email messages from the server** checkbox (default checked) that, when set, runs `docker exec hermes_dovecot rm -rf /srv/mail/<domain>/<local>` to remove the Maildir. The DB rows are deleted regardless of that checkbox; only the on-disk messages are conditional. Maildir deletion is wrapped in a non-fatal `cftry` — failure leaves the messages on disk for an admin to clean up later, but the DB state is correct.

User-initiated folder shares — same engine, different page

Individual users can share folders from their own mailbox with other users via the User Portal (`/users/2/`), and those shares land in `user_folder_shares` rather than `shared_mailbox_permissions`. They are projected to `dovecot-acl` files by `sync_user_folder_acl_file.cfm` using the same vfile driver. The **Rebuild ACL Files** button on this page rebuilds both types of share in one pass, so admins don't have to think about the distinction when troubleshooting.

The two share types are otherwise independent:

	Admin-managed shared mailbox	User-initiated folder share
Surface	This page	User Portal > Folder Sharing
Storage	`shared_mailboxes` + `shared_mailbox_permissions`	`user_folder_shares`
Underlying mailbox	A dedicated `mailboxes` row with `mailbox_type='shared'`	The owner's existing mailbox + a named folder path
Visibility namespace	`Shared/<address>/INBOX`	`Shared/<owner>/<folder_path>`
ACL file path	`/srv/mail/<dom>/<local>/dovecot-acl`	`/srv/mail/<owner-dom>/<owner-local>/<folder>/dovecot-acl`
Cleanup on member removal	This page's Remove Permission	Owner removes the share from User Portal

Cross-domain members — not supported, enforced server-side

A shared mailbox on `company.com` can only be shared with users whose mailboxes are also on `company.com`. The same-domain rule is enforced in three places:

1. **Add Shared Mailbox modal** — the Initial Members list is filtered client-side to the selected domain.
2. **Manage Permissions modal** — the Tom Select picker is repopulated on open to only show users in the shared mailbox's domain.
3. **`add_permission` action handler** — compares `getUserMailbox.domain_id` against `getShared.domain_id` and returns error 26 on mismatch, so a forged form post can't bypass the UI filter.

The Dovecot shared namespace itself does not enforce this — the `acl_sharing_map` query keys on username, not domain — so the rule is a UX contract, not a Dovecot constraint. If you need a single inbox readable across multiple domains, the workable pattern is one shared mailbox per domain with a virtual recipient fan-out feeding both.

Nextcloud Mail caches the folder tree per account

Nextcloud Mail (the NC webmail app) caches each connected account's IMAP folder tree the first time the account is added and refreshes it lazily. **A user who is newly granted access to a shared mailbox via this page will NOT see it in Nextcloud Mail until they remove and re-add their NC mail account.** Standalone IMAP clients (Thunderbird, Outlook, Apple Mail) refresh the folder tree on the next IDLE cycle or manual sync, so they don't have this gotcha.

This is upstream NC Mail behavior, not a Hermes setting. The workaround is documented for end-users in the User Portal documentation; for admins, the remediation is to tell the affected user to re-add their NC mail account once the share is in place.

Feature-disabled behavior

When the Mailbox Sharing master switch on Settings is **off**:

- The Add / Rebuild / Manage Permissions buttons render disabled with a tooltip pointing back to Settings.
- An amber banner at the top of the page explains the state and links to Settings.
- Existing shared mailboxes appear in the table with status badge `Inactive (Sharing Off)` so the admin can see what would resume when the switch is flipped back on.
- The Delete button still works — admins can clean up rows while the feature is off.
- The `add_shared_mailbox`, `add_permission`, `edit_permission`, and `sync_all_acl_files` action handlers all check the master switch at entry and return error 31 if it's off, so a stale tab can't silently bypass the guard.

Dovecot itself does not declare the `Shared/` namespace when the master switch is off, so IMAP clients won't see shared folders even if the on-disk ACL files exist. Existing ACL files are preserved and re-activate as soon as the switch is flipped back on.

Failure semantics

What breaks	What happens
Master switch off + Add / Edit / Sync attempted	error 31, no DB write
Blank address prefix	error 10

What breaks	What happens
Address prefix has invalid characters	error 11
Domain missing or not mailbox-type	error 12
Address collides with mailbox / alias / virtual recipient / existing shared mailbox	error 13
Quota not numeric or `<= 0`	error 14
Blank display name	error 15
Stale shared_mailbox_id (deleted between page load and submit)	error 21
Invalid user_mailbox_id	error 22
User already has permissions on this shared mailbox	error 23
Stale permission_id (Edit / Remove)	error 24
Add / Edit Permission with all seven flags off	error 25
Cross-domain member attempt	error 26
Any database operation throws inside the cftry	error 30, no rows committed
`doveadm mailbox create` fails	non-fatal — Maildir bootstraps via LMTP on first delivery instead
`sync_shared_mailbox_acl_file.cfm` fails	non-fatal — DB is the source of truth; the next permission change retries the sync, or admin can use Rebuild ACL Files
Maildir `rm -rf` on delete fails	non-fatal — DB rows are removed regardless; admin can manually clean up `/srv/mail/<domain>/<local>`

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_shared_mailboxes.cfm`	`hermes_commandbox`	Page + table + Add / Manage / Delete / Rebuild modals
`config/hermes/var/www/html/admin/2/inc/shared_mailbox_actions.cfm`	`hermes_commandbox`	Dispatcher for all six actions (add / delete / add_permission / edit_permission / remove_permission / sync_all_acl_files)
`config/hermes/var/www/html/admin/2/inc/sync_shared_mailbox_acl_file.cfm`	`hermes_commandbox`	Rebuilds one `dovecot-acl` file from `shared_mailbox_permissions`
`config/hermes/var/www/html/admin/2/inc/sync_user_folder_acl_file.cfm`	`hermes_commandbox`	Same engine for user-initiated folder shares

Path	Owner	Role
`config/hermes/var/www/html/admin/2/inc/get_shared_mailbox_permissions_json.cfm`	`hermes_commandbox`	AJAX endpoint for the Manage Permissions table
`/srv/mail/<domain>/<local>/dovecot-acl`	`hermes_dovecot` (vmail:vmail 0660)	Per-mailbox vfile ACL file — Dovecot 2.4's enforcement source
`/srv/mail/<domain>/<local>/`	`hermes_dovecot`	The Maildir itself
`/opt/hermes/tmp/<token>_sync_shared_acl.sh`	`hermes_commandbox`	Throwaway shell script used to ship the ACL payload through `docker exec -i` via heredoc
`shared_mailboxes`, `shared_mailbox_permissions`, `user_folder_shares`, `mailboxes`, `recipients`, `maddr`, `sender_login_maps`, `dovecot_acl`, `dovecot_acl_shared`, `parameters2`	`hermes_db_server`	Storage
`hermes_dovecot` container	—	`doveadm mailbox create` (bootstrap), `rm -rf` (delete), and the in-container `mkdir / cat / chown / chmod` invoked by the sync helper

Related

- Settings — the Mailbox Sharing master switch. Must be on for shared mailboxes to actually function at the IMAP layer. Also the Dovecot TLS profile and connection limits that all shared-mailbox access goes through.
- Mailboxes — the user mailbox list. Members granted permission on this page must already exist there.
- Domains — the mailbox domain list. A shared mailbox is anchored to exactly one domain; cross-domain sharing is not supported.
- Aliases — if you want one inbound address to deliver into one mailbox (rather than be visible to several users), an alias is the lighter-weight option. Aliases have no ACL surface at all.
- Email Relay > Virtual Recipients — the relay-side fan-out pattern. Sometimes a virtual recipient feeding two shared mailboxes (one per domain) is the right tool when a single role address needs to be visible to users on more than one mailbox domain.
- Mailbox Rules — Sieve rules can be configured on shared mailboxes the same way as on user mailboxes; the authentication path is the granting user, not the shared address.
- Authentication Settings — Submission-port auth that the Send-As flag piggybacks on, plus the LDAP backend that Dovecot looks up members against.