Internal CA

Internal CA

Admin path: **Encryption > Internal CA** (`view_internal_ca.cfm`, `inc/download_ca_file.cfm`, `inc/create_certificate.cfm`, `inc/send_smime_certificate.cfm`, `inc/delete_smime_certificate.cfm`).

This is the **gateway's built-in Certificate Authority** for issuing S/MIME certificates to local users and relay recipients. Each CA row here corresponds to a private CA cert + key on disk under `/opt/hermes/CA/<directory>/root_ca/` and a matching `roots`-store entry in the CipherMail (`djigzo`) trust list. Per-recipient S/MIME certs minted from a CA on this page are stored in `recipient_certificates` and listed on Email Server > Relay Recipients (and Email Server > Mailboxes when S/MIME is enabled on a mailbox).

This page is **distinct from** System Certificates:

	System Certificates	Internal CA
What it stores	Operator-uploaded TLS leaf certs (nginx, Postfix, Dovecot)	Private CAs that *mint* S/MIME end-user certs
Trust direction	Hermes presents these to clients	Hermes issues certs that *recipients* present
Backing store	`system_certificates` table + `/opt/hermes/ssl/` or `/etc/letsencrypt/`	`ca_settings` table + `/opt/hermes/CA/<dir>/` + CipherMail `cm_certificates` (`roots` store) + `cm_ctl` trust list
Typical lifetime	90 d (ACME) or 1-3 yr (commercial)	5 yr root (recommended), extendable in place
Lifecycle owner	nginx / Postfix / Dovecot via TLS handshake	CipherMail S/MIME signer / encryptor for outbound; per-recipient cert issuance for inbound encrypt

The two ingest paths

The page exposes two collapsing cards (**Create Internal CA**, **Import External CA**) plus a DataTable of existing CAs. Both paths land a row in `ca_settings` and register the cert in

CipherMail's `cm_certificates` table as a root (`cm_store_name = 'roots'`) plus an entry in `cm_ctl` (Certificate Trust List) flagged `whitelisted`.

1. Create Internal CA

Operator fills the DN fields, picks a key size (2048 / 4096) and a validity (1-5 years; 5 years recommended). Hermes:

1. Validates inputs (regex-restricted character set per field, 2-char ISO country code, uniqueness against `ca_settings.ca_commonname`).
2. Materializes a per-CA on-disk skeleton at `/opt/hermes/CA/<sanitized-cn>/root_ca/` with the standard OpenSSL layout (`certs/`, `crl/`, `newcerts/`, `private/`, `requests/`, `PFX/`, `serial`, `index.txt`, `crlnumber`).
3. Materializes an `openssl.cnf` from `/opt/hermes/templates/rootca_openssl.cnf` with the directory placeholder substituted.
4. Snapshots `cm_certificates` into `cm_certificates_tmp`, runs the OpenSSL root-CA generation script as a one-shot temp script (`/opt/hermes/scripts/<token>_create_ca.sh`), then diffs to find the new cert.
5. Marks the new CipherMail row `cm_store_name = 'roots'`, inserts a `cm_ctl` row with status `whitelisted` and `allowExpired = false`, and back-fills `ca_settings.ca_djigzo_id` + `ca_djigzo_subject`.

2. Import External CA

For organizations that already have a private CA (commercial issuer, internal PKI, prior Hermes install). Operator uploads the **CA cert** (PEM) and the **CA private key** (PEM, unencrypted). Hermes:

1. Lands the files at `/opt/hermes/CA/<sanitized-cn>/root_ca/certs/cacert.pem` and `.../private/cakey.pem`.
2. Runs an OpenSSL validation script that checks:
   - Cert parses as X.509 (`openssl x509 -modulus`)
   - Key parses as RSA (`openssl rsa -modulus`)
   - Cert and key moduli match (private key matches public key)
   - Cert has `CA:TRUE` basic constraint
3. On any check failure the upload directory is removed and the operator gets a specific error alert (m=48 / 49 / 50 / 51).
4. Generates `openssl.cnf` from the template + `cachain.pem` = copy of the cert (needed for later PFX export of per-user certs).
5. Pipes the cert into CipherMail via `docker exec -i hermes_ciphermail /usr/bin/java -cp '/usr/share/djigzo/lib/*' mitm.application.djigzo.tools.CertStore --import-certificates` and back-fills the `ca_djigzo_id` exactly as the Create path does.

The Import path is the only way to migrate a CA that already has issued certs in the wild — re-creating a CA from scratch with the same DN does NOT reproduce the original key material, so previously issued certs would not chain to it.

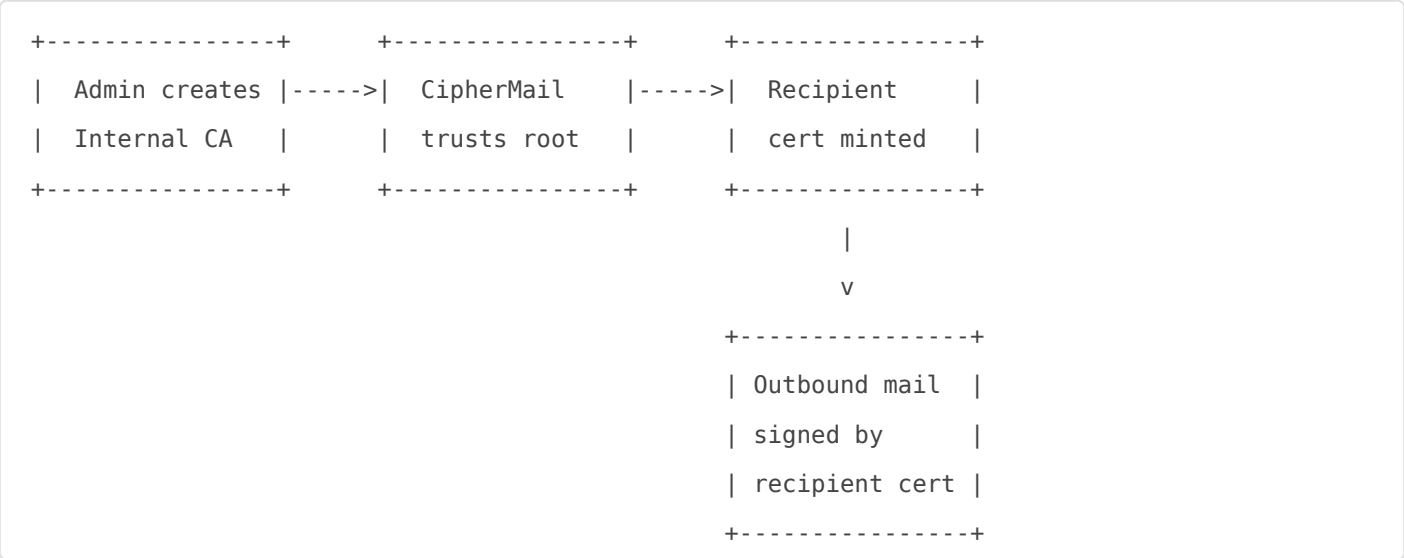
Default CA flag (`default2`)

Exactly one row in `ca_settings` has `default2 = '1'`; all others have `'2'`. The default CA is the one Hermes mints from when an admin clicks **Create Certificate** for a recipient on Email Server > Relay Recipients (or the mailbox equivalent) without explicitly choosing a CA. The page enforces single-default by:

- Setting all rows to `'2'` before flipping the new row to `'1'`
- Forcing the first-ever CA to default regardless of the checkbox
- Disabling the **Make Default** checkbox on the Create card when no default exists (forced default is implicit)

The DataTable Default column renders a green **YES** badge for the default row and a one-click **set default** button for the others.

CA lifecycle workflow



Stage	Where the data lives	Trigger
CA root created	`ca_settings` + `/opt/hermes/CA/<dir>/` + `cm_certificates` (`roots`) + `cm_ctl` (`whitelisted`)	Create / Import buttons on this page

Stage	Where the data lives	Trigger
Per-recipient cert minted	`recipient_certificates` (or `external_recipient_certificates`) + CipherMail user store	**Create Certificate** button on a recipient page; uses `default2 = '1'` CA unless overridden
Cert self-introduction	Bundled into the first signed outbound message the recipient sends	Automatic on first S/MIME-signed send
Cert revocation	`delete_smime_certificate.cfm` removes the row + CipherMail entry; CRL is maintained by CipherMail's own scheduled job	**Delete** button on the recipient cert row
CA renewal	Re-sign the existing cert + key with `openssl x509 -days <N>` and re-import into CipherMail; `ca_settings.expires` updated	**Renew** button (sync icon) on the CA row
CA deletion	Refused if any `recipient_certificates.ca_id` row references it; otherwise removes DB row + CipherMail `cm_certificates` / `cm_ctl` + on-disk tree	**Delete** button (only enabled when zero issued certs)

CA Renewal: 5-year extension in place

Clicking the **Renew** (sync) icon does NOT generate a new key pair — it re-signs the existing CA cert against its own key with an extended `notAfter`. The math:

```
new_expires = current_expires + 5 years
days_param  = max(1825, days_from_now_to_new_expires)
openssl x509 -in cacert.pem -days <days_param> -out cacert.pem.new -signkey cakey.pem
mv cacert.pem.new cacert.pem
cp cacert.pem cachain.pem
cat cacert.pem | docker exec -i hermes_ciphermail \
  /usr/bin/java -cp '/usr/share/djigzo/lib/*' \
  mitm.application.djigzo.tools.CertStore --import-certificates
```

Because the key stays the same, every previously issued recipient cert **still chains to a valid CA cert** — there is no need to re-mint recipient certs after a CA renewal. This is the operator-friendly path: recipients on the outside who already trust the CA root continue to trust it transparently.

The old CipherMail row is deleted and the renewed cert re-imported so the `cm_certificates`/`cm_ctl` rows reflect the new validity window (otherwise CipherMail would keep enforcing the old expiry).

Trust distribution to external recipients

A Hermes-issued S/MIME cert is signed by a private CA that **no operating system or mail client trusts by default**. External recipients see Hermes-signed mail as "signed by an unknown CA" until they explicitly install the Internal CA root in their trust store.

Two practical paths:

Path	Effort	Reach
Operator distributes the CA root out-of-band (download from this page, email or publish on a portal, recipient installs in Outlook / macOS Keychain / iOS Profile / Thunderbird)	Manual per recipient	Small fixed counterparty set (B2B, partner orgs)
Issue recipient certs from a publicly-rooted CA (commercial S/MIME issuer signs your CA, or you buy per-user S/MIME certs from a public issuer)	One-time cross-sign or per-user cost	Every MUA on the planet trusts the chain

For most Hermes deployments the Internal CA is the right answer (per-user public S/MIME costs $20-$80/yr/user); for high-volume B2C senders the publicly-rooted route is sometimes worth the cost.

Hermes does not generate a CRL distribution URL on this page; CipherMail maintains the revocation list internally and applies it when verifying inbound S/MIME from local recipients. External recipients have no automatic way to consume the CRL — revocation is effectively local-only unless the operator publishes the CRL manually.

CA file downloads (gated)

Each row's action column exposes a **Download Certificate** and **Download Private Key** button. These are **disabled by default** — downloading a CA private key off a web console is a high-risk operation. To enable, set

```
ALLOW_CA_DOWNLOAD=yes
```

in `/opt/hermes/config/security.conf` on the host filesystem. This is the same toggle pattern used by System Certificates (`ALLOW_CERT_DOWNLOAD`) — read on every page load, surfaced as a disabled-button + tooltip when off. When enabled, downloads stream via a hidden iframe (`<iframe id="caDownloadFrame">`) so the page preloader doesn't get stuck.

Body-modification interaction with S/MIME

CipherMail-side S/MIME signing happens **after** the `hermes_body_milter` disclaimer / signature / banner insertion (see Disclaimers — Behavior with S/MIME, PGP, and DKIM-signed mail). That means outbound mail signed by an Internal-CA-minted recipient cert covers the **final body** the recipient sees — including any disclaimer or banner Hermes appended. The body milter passes already-S/MIME-signed mail through untouched, so end-to-end MUA-signed mail (Outlook + per-user S/MIME) is never re-signed or invalidated.

This is the same ordering rationale that drives ARC sealing placement (see ARC Settings — Container and milter placement): the cryptographic envelope is the last thing applied so it always matches the bytes the recipient sees.

Container and database touch-points

Component	Container / path	Role
Page	`config/hermes/var/www/html/admin/2/view_internal_ca.cfm` (`hermes_commandbox`)	CRUD + DataTable + action router
CA tree	`/opt/hermes/CA/<sanitized-cn>/root_ca/` (`hermes_commandbox` bind mount)	OpenSSL working tree per CA
Templates	`/opt/hermes/templates/rootca_openssl.cnf` + `/opt/hermes/scripts/create_ca.sh`	Placeholder-substituted at create time
Trust store	`cm_certificates` + `cm_ctl` + `cm_ctl_cm_name_values` in `djigzo` DB (`hermes_db_server`)	CipherMail's view of the root CA

Component	Container / path	Role
Engine	`hermes_ciphermail` (Java; CipherMail Community 5.x branded `djigzo`)	Signing / encryption / decryption engine; reached via `docker exec -i hermes_ciphermail /usr/bin/java -cp '/usr/share/djigzo/lib/*' mitm.application.djigzo.tools.CertStore`
Recipient certs	`recipient_certificates` + `external_recipient_certificates` in `hermes` DB	One-row-per-user, joined to a CA via `ca_id`
Security toggle	`/opt/hermes/config/security.conf` on host	`ALLOW_CA_DOWNLOAD=yes` to expose cert/key download buttons

Every CipherMail interaction is **temp-script + docker exec** rather than direct invocation — the `hermes_commandbox` container has no JVM of its own; the CipherMail Java tooling lives in `hermes_ciphermail` and is reached over the docker socket.

Related

- Encryption Settings — outbound encryption policy (force / opportunistic / off); decides whether Hermes signs at all and whether it falls back to plaintext when no recipient key is available
- External Recipients — per-counterparty key store; `external_recipient_certificates` rows pair with Internal-CA-issued or externally-issued S/MIME chains
- PGP Key Servers — sibling page for the PGP side of recipient key distribution
- System Certificates — distinct TLS cert store for nginx / Postfix / Dovecot
- Disclaimers — body-modification ordering vs the S/MIME signer
- ARC Settings — same milter-ordering pattern applied to inbound chain sealing
- **Advanced Settings** (sidebar link to `/ciphermail/`) — CipherMail's own admin UI, exposed for deep operations (CRL publishing, per-user policy tuning) not surfaced in the Hermes admin

---

Revision #48
Created 2026-05-31 12:52:37 UTC by Dino Edwards
Updated 2026-06-20 13:33:21 UTC by Dino Edwards