External Recipients

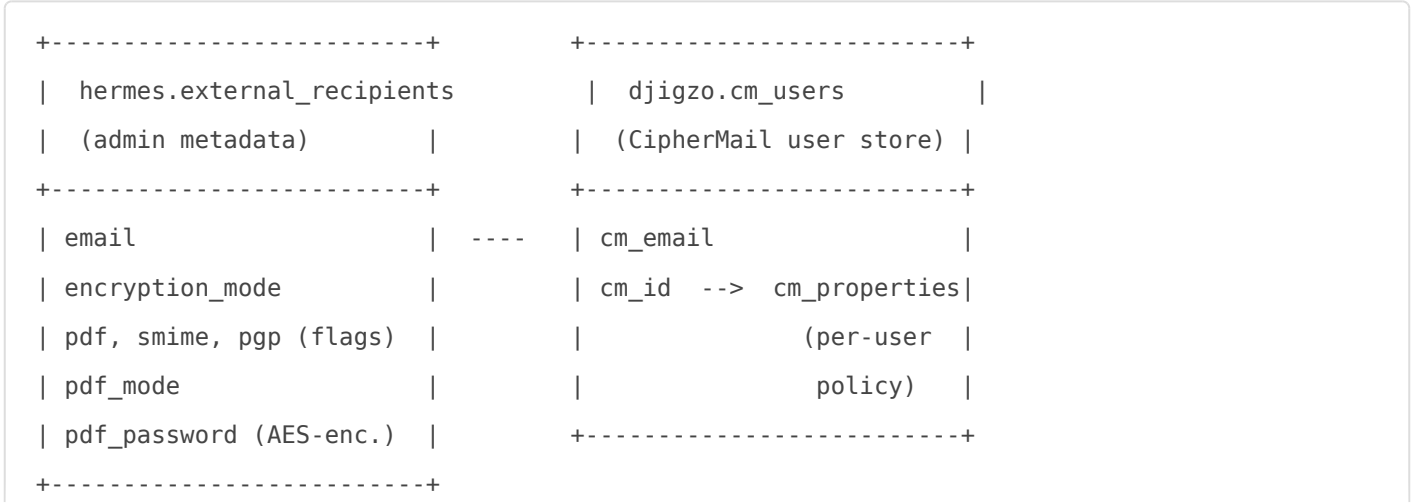
External Recipients

Admin path: **Encryption > External Recipients** (`view_ext_rec_encryption.cfm`, `view_create_ext_recipient.cfm`, `view_ext_smime_certificates.cfm`, `view_ext_pgp_keyrings.cfm`, `view_ext_add_smime_cert.cfm`, `view_ext_add_pgp_keyring.cfm`, `inc/create_ext_recipient.cfm`, `inc/delete_ext_recipient.cfm`, `inc/reset_pdf_password.cfm`, `inc/reset_portal_password.cfm`).

This is the **per-counterparty encryption policy and key store** for external (non-managed) email addresses. Each row binds a single external email to one of three protocols (PDF / S/MIME / PGP) and to one of two trigger modes (Mandatory / By Subject). It is the page where the policy referenced by Encryption Settings actually takes effect — the global page chooses the **mechanism** (subject trigger keyword, shared secrets, PDF reply sender); this page chooses the **policy** for every external recipient the gateway encrypts to.

The DataTable is the master view across **both** Hermes-side metadata (`external_recipients` in the `hermes` DB) and CipherMail's own user table (`cm_users` in the `djigzo` DB), joined on email address. Rows are tagged **Admin-Configured** (explicitly created on this page, with a matching `external_recipients` row) or **Auto-Discovered** (materialized by CipherMail during message processing, no `external_recipients` row).

Schema: two tables, one view



```
            |
            v
   Page renders Admin badge
            |
+--------------------------+
| If NO matching row,      |
| recipient is "Auto" with |
| inferred policy from     |
| cm_certificates_email /  |
| cm_keyring_email         |
+--------------------------+
```

The page never N+1's against CipherMail — three batch queries build struct lookups (`adminLookup`, `smimeLookup`, `pgpLookup`) and the row loop reads from those instead of per-row queries. That matters at any scale beyond a few hundred recipients.

`external_recipients` columns:

Column	Purpose
`id`	PK
`email`	External email address (joined to `cm_users.cm_email`)
`encryption_mode`	`pdf_mandatory` / `pdf_by_subject` / `smime_mandatory` / `smime_by_subject` / `pgp_mandatory` / `pgp_by_subject`
`pdf` / `smime` / `pgp`	Flag (1 / NULL) indicating which protocol is the active one for this recipient
`pdf_mode`	For PDF only: `static` / `random` / `backtosender`
`pdf_password`	AES-encrypted (with `/opt/hermes/keys/hermes.key`) copy of the static PDF password — for admin re-display only; CipherMail holds its own copy
`smime_mode` / `pgp_mode`	Reserved for parity; populated identically to `encryption_mode` for the matching protocol

Encryption modes

The 6 encryption modes map cleanly onto two axes (protocol × trigger):

Mode	CipherMail `user.encryptMode`	CipherMail `user.pdf.encryptionAllowed`	CipherMail `user.sMIMEEnabled`	CipherMail `user.pgp.enabled`
`pdf_mandatory`	`mandatory`	`true`	`false`	`false`

Mode	CipherMail `user.encryptMode`	CipherMail `user.pdf.encryptionAllowed`	CipherMail `user.sMIMEEnabled`	CipherMail `user.pgp.enabled`
`pdf_by_subject`	`allow`	`true`	`false`	`false`
`smime_mandatory`	`mandatory`	`false`	`true`	`false`
`smime_by_subject`	`allow`	`false`	`true`	`false`
`pgp_mandatory`	`mandatory`	`false`	`false`	`true`
`pgp_by_subject`	`allow`	`false`	`false`	`true`

"By Subject" requires **Encryption Settings > Trigger Encryption by Subject = Enabled** plus the configured keyword (default `[encrypt]`) in the message subject. See Encryption Settings — Subject Trigger for the decision tree.

PDF mode: three sub-policies

PDF encryption is the lowest-friction protocol (recipient needs only a PDF reader and a password — no certs, no keys, no portal account required up front), so it ships with three independent password-distribution sub-modes:

`pdf_mode`	How the password reaches the recipient	When to use
`random`	CipherMail auto-generates a one-time password per message and pushes it through the Secure Email Portal (`https://<console>/web/portal`); recipient self-registers on first use	Default. Best for ad-hoc / first-time external recipients
`static`	Admin sets a fixed password once (minimum 12 chars); recipient must already know it via out-of-band channel	Long-term partners who have agreed on a shared secret
`backtosender`	CipherMail generates a per-message password and emails it back to the **original internal sender** for them to relay to the recipient	Compliance scenarios where the sender must explicitly hand the password to the recipient (auditable trail)

For `backtosender`, two extra fields are configurable per recipient:

Field	Range	Purpose
Password Age (minutes)	15-240	How long the random password is valid

Field	Range	Purpose
Password Length	16-bit / 20-bit	Bit-strength of the generated random password

Bulk vs single create

The **Create External Recipient** page (`view_create_ext_recipient.cfm`) exposes a Single / Bulk toggle:

Mode	Protocol options	Use case
Single	PDF, S/MIME, PGP (all three modes available)	One-off precise configuration including S/MIME / PGP recipients that need a cert/key uploaded afterward
Bulk	**PDF only** (Mandatory or By Subject)	Mass-onboard a list of external addresses, one per line; the UI auto-hides S/MIME and PGP because those protocols need per-recipient cert/key material that has no bulk equivalent

The bulk path validates and skips per-row (invalid format / internal domain / already-exists rows are reported but do not abort the batch); session variables `bulk_created`, `bulk_skipped`, `bulk_failed` feed a partial-success alert on return.

Both paths refuse internal domains. The check is a `COUNT(*) FROM domains WHERE domain = <recipient-domain>` — if Hermes is the authoritative MX for that domain, the recipient is a local mailbox or relay recipient, not an external recipient, and per-mailbox encryption policy belongs on Email Server > Mailboxes instead.

Auto-Discovered recipients

When CipherMail processes mail to an address it has never seen, it materializes a `cm_users` row with the global defaults. These recipients show up here with **Source = Auto** and no `external_recipients` row backing them. They:

- Use the global Subject Trigger policy (from Encryption Settings)
- Have no per-recipient password mode (PDF random is the CipherMail default)
- Display only the cert / keyring counts CipherMail actually holds
- Cannot be edited from this page (no Admin badge, no action buttons for cert / PGP / password reset) — managing them means **either** promoting them to Admin-Configured by creating an explicit row, **or** dropping into CipherMail's own admin UI at `/ciphermail/`

The Source dropdown defaults to **Admin-Configured** on page load — operators most often want to see what they explicitly configured, not the long tail of mail CipherMail has touched.

Per-row actions

The action column varies by what the recipient is configured for:

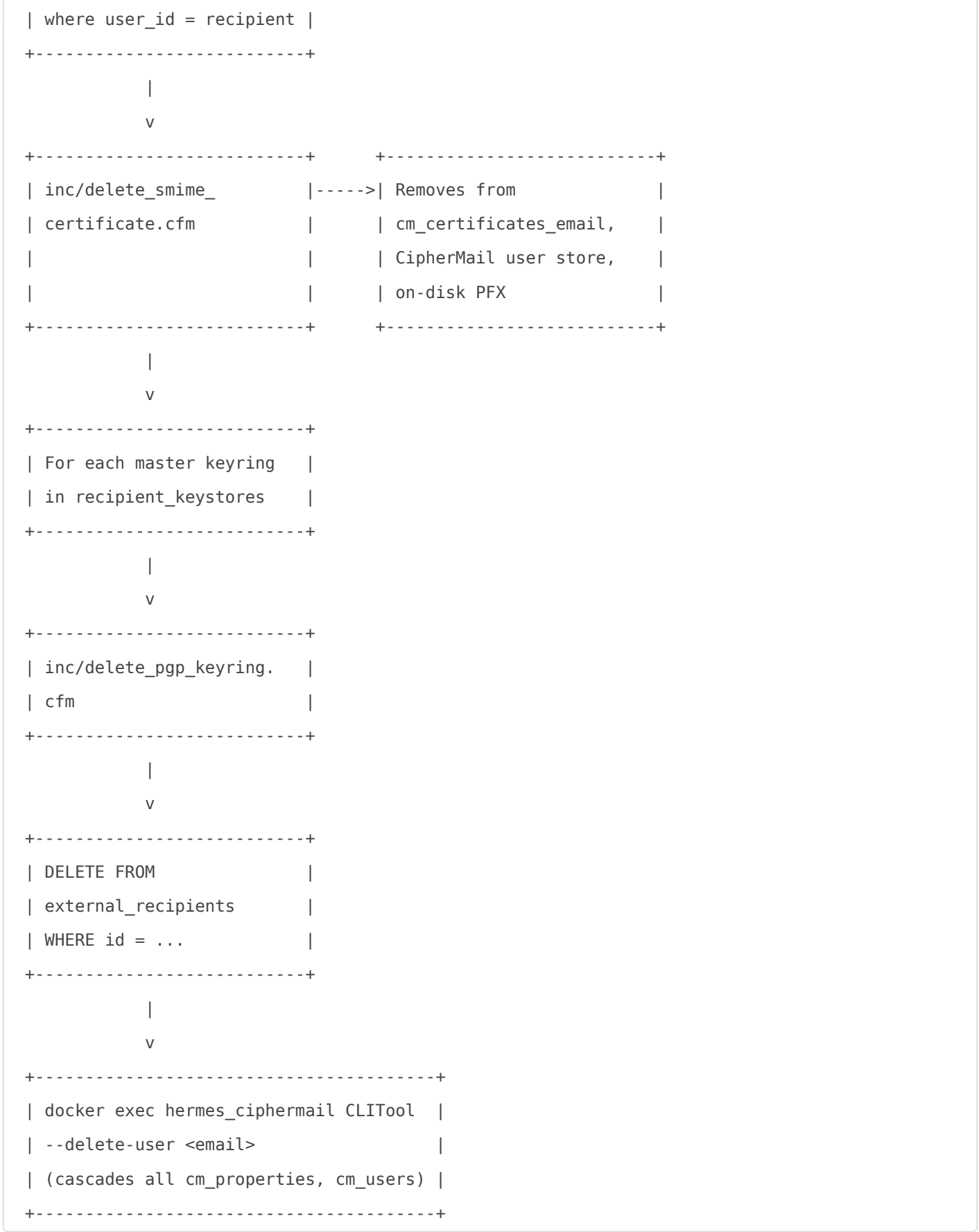
Action	Icon	Visible when	What it does
S/MIME Certificates	`fa-certificate` (green)	Admin row, `smime = 1`	Links to `view_ext_smime_certificates.cfm?email=...` for cert add / delete / send
PGP Keyrings	`fa-key` (blue)	Admin row, `pgp = 1`	Links to `view_ext_pgp_keyrings.cfm?email=...` for keyring add / delete / publish
Reset PDF Password	`fa-file-pdf` (yellow)	Admin row, `pdf = 1` AND `pdf_mode = static`	Opens modal; auto-generates a 16-char mixed-case-alphanumeric password client-side via `generatePassword(16)`; submits to `inc/reset_pdf_password.cfm`
Reset Portal Password	`fa-lock` (grey)	Admin row, `pdf = 1` AND `pdf_mode = random`	Opens modal; same 16-char generator; submits to `inc/reset_portal_password.cfm` (two-step: encode via `--encode-password`, then set `user.portal.password`)
Delete Recipient	`fa-trash-alt` (red)	Every row	Confirms, then submits to `delete_recipient` handler

The Cert Expiry column derives from a batch join of `cm_certificates_email + cm_certificates`, picking the **earliest** `cm_not_after` across all certs for that recipient. Color coding: red bold (already expired), yellow bold (within 30 days), grey muted (more than 30 days).

Delete cascade

Deleting an external recipient is a multi-table operation handled by `inc/delete_ext_recipient.cfm`:

```
+--------------------------+
| For each row in          |
| recipient_certificates   |
```



On success the page surfaces a callout reminding the operator that any **Sender Checks Bypass** mapping tied to this recipient must be re-created — that relationship is not auto-cascaded.

Password reset specifics

PDF static password reset (`inc/reset_pdf_password.cfm`):

1. Writes a one-liner `CLITool --set-property user.password --value <newpass> --encrypt --email <recipient>` to `/opt/hermes/tmp/<token>_reset_pdf_password.sh`.
2. `chmod +x`, executes (240s timeout), deletes.
3. AES-encrypts the new password with `/opt/hermes/keys/hermes.key` and UPDATEs `external_recipients.pdf_password` so the admin re-display path still works.

Portal password reset (`inc/reset_portal_password.cfm`) is **two-step** because CipherMail's portal password is stored as an encoded value, not the raw string:

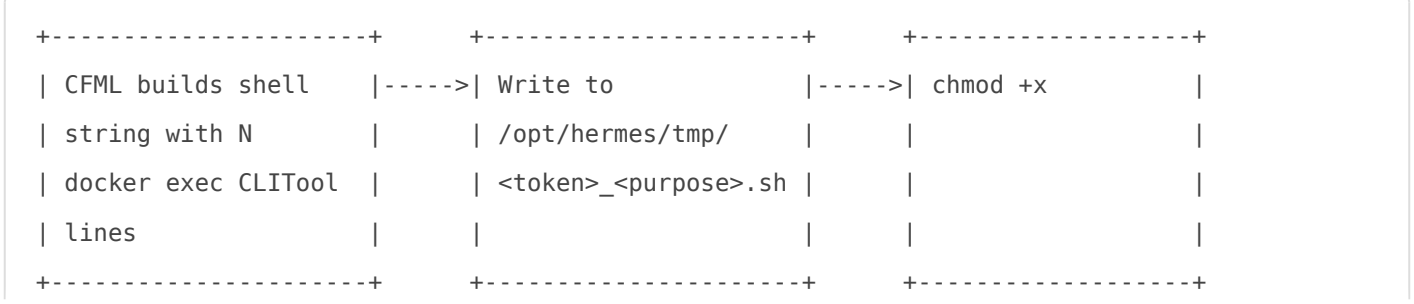
1. **Step 1 — encode:** runs `CLITool --encode-password <newpass>`, captures stdout to `/opt/hermes/tmp/<token>_portal_password`, reads that file back into CFML, deletes the temp file.
2. **Step 2 — set:** runs `CLITool --set-property user.portal.password --encrypt --email <recipient> --value <encoded>` to push the encoded value into CipherMail.

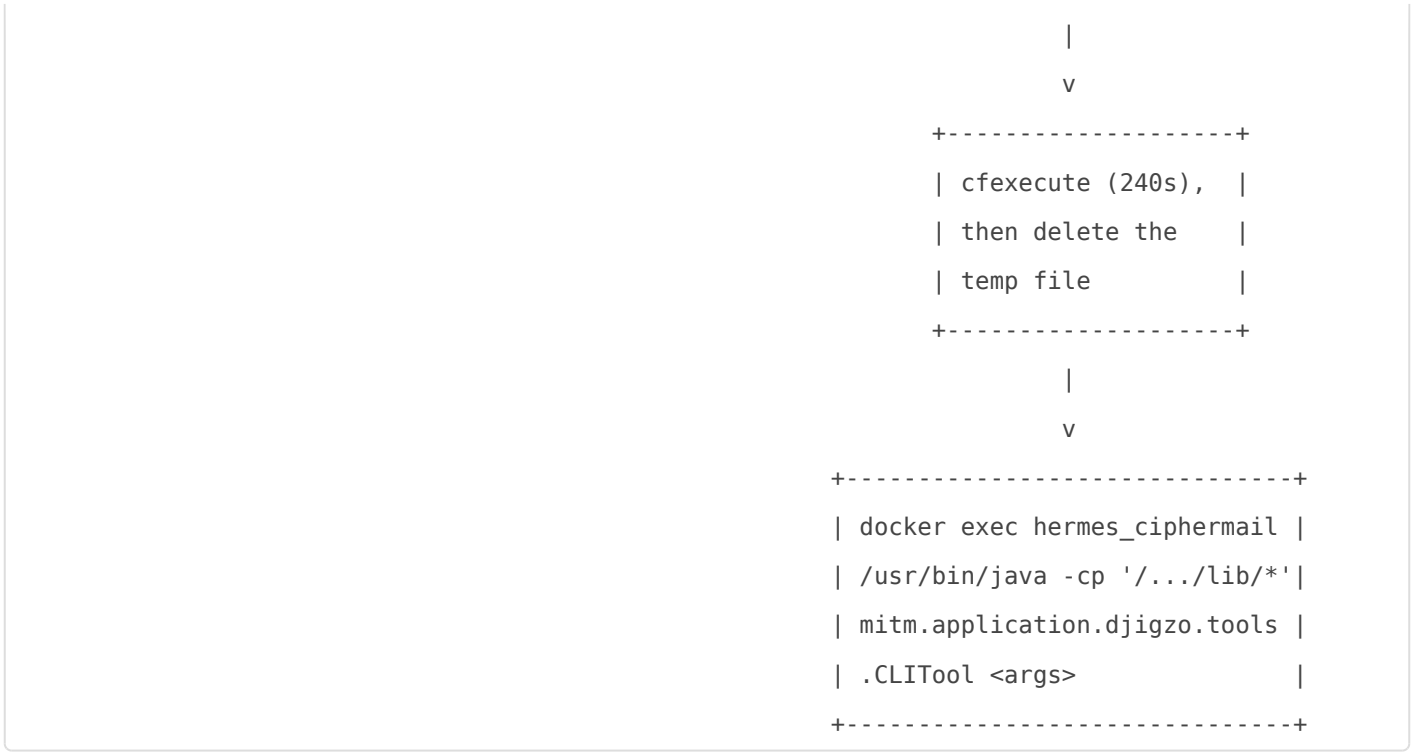
Both modals auto-generate a 16-character mixed-case-alphanumeric password client-side and pre-populate the hidden confirm field; the operator can regenerate or type-in their own. Min length 12 is enforced server-side; the regenerator produces 16.

The modal text explicitly notes that **unencrypted voice calls and texts are NOT considered secure** for relaying the password to the recipient — operators are expected to use Signal, an in-person exchange, or a separately-encrypted channel.

CipherMail integration: every action is docker exec

Every CipherMail-side mutation on this page uses the same pattern documented across the Hermes admin:





The Hermes app container (`hermes_commandbox`) holds no JVM and no CipherMail libraries; everything reaches into `hermes_ciphermail` over the docker socket via `CLITool`. The temp-script pattern (write + chmod + execute + delete) survives the Lucee `cfexecute` quirks around stderr and quoting that would otherwise make a direct inline invocation unreliable.

What's NOT on this page

Expectation	Where it actually lives
Per-recipient cipher / algorithm selection (AES-128 vs AES-256, RSA / EC)	CipherMail Advanced Settings (`/ciphermail/`); per-recipient overrides live in `cm_properties` directly
Auto-lookup of recipient PGP keys from a keyserver at send time	Not implemented; see PGP Key Servers — that page is publish-only. Keys must be uploaded manually on the PGP Keyrings sub-page
Auto-lookup of recipient S/MIME certs via LDAP / public directory	Not implemented; certs must be uploaded manually on the S/MIME Certificates sub-page, OR minted from an Internal CA row and sent to the recipient
Per-recipient subject-trigger keyword override	Not implemented; the keyword is global (one row in `encryption_settings`)
Recipient-side enrollment / self-service for their own keys	The Secure Email Portal handles recipient password registration for PDF-random mode; there is no self-service cert / PGP upload UI
Bulk import from CSV with mixed protocols	Bulk path is PDF-only by design (S/MIME / PGP need per-recipient material that doesn't bulk-import cleanly)

Expectation	Where it actually lives
Sender-side "force encrypt for this thread" UI	Senders use the subject trigger; there is no per-mailbox sender UI

Container and database touch-points

Component	Container / path	Role
Page	`config/hermes/var/www/html/admin/2/view_ext_rec_encryption.cfm` (`hermes_commandbox`)	List, filter, password resets, delete
Create page	`view_create_ext_recipient.cfm` + sub-pages for cert / keyring management	Single + bulk insertion
Action includes	`inc/create_ext_recipient.cfm`, `inc/delete_ext_recipient.cfm`, `inc/reset_pdf_password.cfm`, `inc/reset_portal_password.cfm`	One-liner CLITool dispatchers via temp script
Admin metadata	`external_recipients` in `hermes` DB (`hermes_db_server`)	Per-recipient policy choices + AES-encrypted static PDF password copy
CipherMail user store	`cm_users`, `cm_properties` in `djigzo` DB	Authoritative per-recipient state
CipherMail cert / key index	`cm_certificates_email`, `cm_certificates`, `cm_keyring_email` in `djigzo` DB	Joined batch into `smimeLookup` / `pgpLookup` for column rendering
Encryption engine	`hermes_ciphermail` (Java; CipherMail Community 5.x branded `djigzo`)	Actual S/MIME / PGP / PDF encryption + portal back-channel
AES key	`/opt/hermes/keys/hermes.key` (`hermes_commandbox` bind mount)	Encrypts `pdf_password` for re-display
Secure Email Portal	`https://<console.host>/web/portal/` (served by `hermes_ciphermail`)	Recipient-facing landing page for PDF random + portal account flows

Related

- Encryption Settings — global Subject Trigger, PDF reply sender, three shared secrets; the policy mechanism this page applies per recipient
- Internal CA — where the private CAs that can mint S/MIME certs for these recipients live (operator-issued S/MIME chain delivered out-of-band)

- PGP Key Servers — the publish list for the Publish action on the PGP Keyrings sub-page (note: publish-only, not lookup)
- System Certificates — distinct TLS cert store; not related to message-content S/MIME
- Disclaimers — body-mod ordering vs the CipherMail encryption pass (disclaimer is appended before encryption wraps the message)
- Organizational Signatures — same milter ordering applies to signature injection
- ARC Settings — same milter-ordering pattern applied to inbound chain sealing
- **Advanced Settings** (sidebar link to `/ciphermail/`) — CipherMail's own admin UI for everything not surfaced here (per-recipient cipher tuning, custom DLP, direct `cm_properties` editing)

Revision #8
Created 2026-05-31 12:52:36 UTC by Dino Edwards
Updated 2026-05-31 14:01:27 UTC by Dino Edwards