PGP Key Servers

PGP Key Servers

Admin path: **Encryption > PGP Key Servers** (`view_pgp_key_servers.cfm`, `inc/publish_pgp_keyring.cfm`).

This page maintains the **HKP keyserver publish list** — the set of public OpenPGP keyservers Hermes will push (`gpg --send-keys`) recipient public keys to when an admin clicks **Publish** on a keyring row in **Encryption > External Recipients**. Each row is a hostname only (no scheme, no port, no path) stored in the `pgp_keyservers` table.

> **Important: publish, not lookup.** Despite the page name, the keyserver list is currently **outbound-only**. Hermes does NOT auto-query these servers to fetch a recipient's PGP key at send time — recipient keys must be imported manually (paste-in or file upload) on **Encryption > External Recipients > PGP Keyrings**. The keyservers configured here are used solely by the **Publish** action in `inc/publish_pgp_keyring.cfm`, which pushes a key the operator already holds (typically the local CipherMail server's public key or a recipient's key that was imported and now needs broader distribution).

What the page does

The page is a thin CRUD over a 3-column table:

`pgp_keyservers` column	Purpose
`id`	PK
`keyserver`	Hostname only, e.g. `keys.openpgp.org`
`note`	Free-text label, e.g. "Primary keyserver"

Three actions:

Action	Form value	Effect

Add	`action=add`	Validates hostname via `IsValid("email", "bob@" & ks)` (rejects URLs and `host:port`), checks for duplicate `keyserver`, INSERTs the row
Single delete	`action=delete` with `delete_id`	DELETE one row by id
Bulk delete	`action=bulk_delete` with `selected_ids` (CSV)	DELETE every selected id in a loop

The existing-servers card is a DataTable with select-all + per-row checkboxes + a **Delete Selected** button. There is no per-row enable flag, no protocol/port column, no priority ordering — every row in the table is offered as a publish target in the modal on the keyring page, indexed by `id`.

What "publish" actually runs

When the operator clicks **Publish** on a keyring row at **External Recipients > PGP Keyrings**, the `publish_pgp_keyring.cfm` include does the following for each selected keyserver:

```
/usr/bin/gpg --homedir /opt/hermes/.gnupg/ \
             --keyserver <hostname-from-pgp_keyservers> \
             --send-keys <recipient-PGP-key-id>
```

The temp script is written to `/opt/hermes/tmp/<token>_publish_pgp_key.sh`, chmod'd, executed, and deleted. The standard Hermes temp-script pattern. The keyserver hostname is substituted via `REReplace` of the `THE-KEY-SERVER` placeholder in `/opt/hermes/scripts/publish_pgp_key.sh`.

GPG itself picks the protocol — `gpg` defaults to `hkps://` (HKP over TLS on tcp/443) for a bare hostname when the local `dirmngr` is configured for it; otherwise it falls back to `hkp://` (tcp/11371). Hermes does not pass an explicit scheme.

Failure modes the include recognizes (sets `session.m` and redirects):

GPG stderr fragment	Meaning	`session.m`
`Server indicated a failure`	Keyserver rejected the upload (rate limit, policy, malformed key)	22
`No name`	Local GPG keyring has no user-id matching the requested key id	23
`Not found`	Local GPG keyring does not hold the requested key id	24
`Not a key ID`	The key id parameter was malformed	25

A successful publish returns no recognized fragment and falls through to the success branch.

Recommended seed list

The default install seeds one row:

Hostname	Note
`keyserver.ubuntu.com`	Ubuntu SKS OpenPGP Public Key Server

Practical 2026 replacements / additions the operator should consider:

Hostname	Network	Caveats
`keys.openpgp.org`	Identity-verified standalone (Hagrid)	Strips third-party signatures (no web-of-trust); requires email verification before a key becomes searchable by email address; **does not distribute revocation certificates the way SKS did**
`keyserver.ubuntu.com`	SKS-style federated	Was the last reliable SKS-network bridge; survives but is no longer broadly federated
`pgp.mit.edu`	Legacy SKS	Largely defunct in 2026 — uploads may not propagate; leave off unless legacy compatibility is required
`<your-org-keyserver>`	Internal HKP daemon (e.g. Hagrid)	Useful if the operator runs an authoritative keyserver for their own domain — same publish path

The page does NOT validate keyserver reachability at add time; an unreachable host simply produces a publish failure when the operator clicks Publish later.

What is NOT on this page

Several things an operator might reasonably expect from a "PGP Key Servers" page that are intentionally elsewhere or absent:

Expectation	Where it actually lives
Per-server enable/disable toggle	Not implemented — every row is a publish target
Search-order priority	Not applicable — publish iterates the explicit selection from the modal, not the full list

Expectation	Where it actually lives
Inbound recipient-key auto-lookup at send time (`gpg --search-keys` / `recv-keys`)	Not implemented anywhere in Hermes; recipient keys must be imported manually on **External Recipients > PGP Keyrings**
Automatic refresh of imported keys (re-fetch + merge updates)	Not implemented; operators must re-import a key if a recipient rotates
DANE `OPENPGPKEY` DNS lookup	Not currently surfaced in the Hermes admin or CipherMail engine config
WKD (Web Key Directory) discovery at `https://<domain>/.well-known/openpgpkey/...`	Not currently surfaced in the Hermes admin or CipherMail engine config
HKP port override	Not on this page; GPG picks the port
Encryption policy decisions ("fail closed vs send plaintext if no key")	Encryption Settings, not here

The page is deliberately scoped to one job: **a list of HKP endpoints the publish flow can push to.**

When the operator should populate this list

Two practical scenarios:

1. **The organization wants its own gateway PGP key to be publicly discoverable.** Add the operator's preferred public keyserver(s), then publish the local CipherMail key from **External Recipients > PGP Keyrings**. External counterparties running `gpg --recv-keys` against the same keyserver can then pull it for encrypting mail back to Hermes-served users.
2. **A specific recipient has asked for their key (which the operator already holds locally) to be pushed somewhere centralized.** Less common — usually recipients self-publish — but the workflow supports it.

If the deployment never publishes keys outward (typical Community deployments that use S/MIME exclusively, or PGP deployments that exchange keys out-of-band via attachment), this page can remain empty with no functional impact.

Container and database touch-points

Component	Location	Role
Page	`config/hermes/var/www/html/admin/2/view_pgp_key_servers.cfm` (`hermes_commandbox`)	CRUD UI
Publish include	`config/hermes/var/www/html/admin/2/inc/publish_pgp_keyring.cfm` (`hermes_commandbox`)	Builds + runs the temp `gpg --send-keys` script
Template script	`/opt/hermes/scripts/publish_pgp_key.sh`	Single line: `/usr/bin/gpg --homedir /opt/hermes/.gnupg/ --keyserver THE-KEY-SERVER --send-keys THE_KEY_ID 2>&1`
GPG home	`/opt/hermes/.gnupg/` (bind-mounted into `hermes_commandbox`)	Local GPG keyring holding the keys eligible for publish
Storage	`pgp_keyservers` in `hermes` DB (`hermes_db_server`)	The list itself
Engine	`hermes_ciphermail` (separate from publish — handles actual signing/encryption at send time)	NOT touched by this page; this page only manages the GPG outbound-publish list

The publish flow runs **gpg on `hermes_commandbox`** (which has the `/opt/hermes/.gnupg/` keyring bind-mounted) — not inside `hermes_ciphermail`. CipherMail keeps its own per-recipient PGP store in the `djigzo` DB for actual encryption/decryption operations.

Related

- External Recipients — per-counterparty key store; the **Publish** action that consumes this list lives on the keyring sub-page there
- Encryption Settings — outbound encryption policy that decides whether absence of a recipient PGP key blocks the message or falls through to plaintext
- Internal CA — sibling page for the S/MIME side of recipient key issuance and trust
- **Advanced Settings** (sidebar link to `/ciphermail/`) — CipherMail's own admin UI for the deep PGP keyring operations the Hermes admin does not surface

---

Revision #8
Created 2026-05-31 12:52:38 UTC by Dino Edwards
Updated 2026-05-31 14:01:27 UTC by Dino Edwards