Password Resets

Password Resets

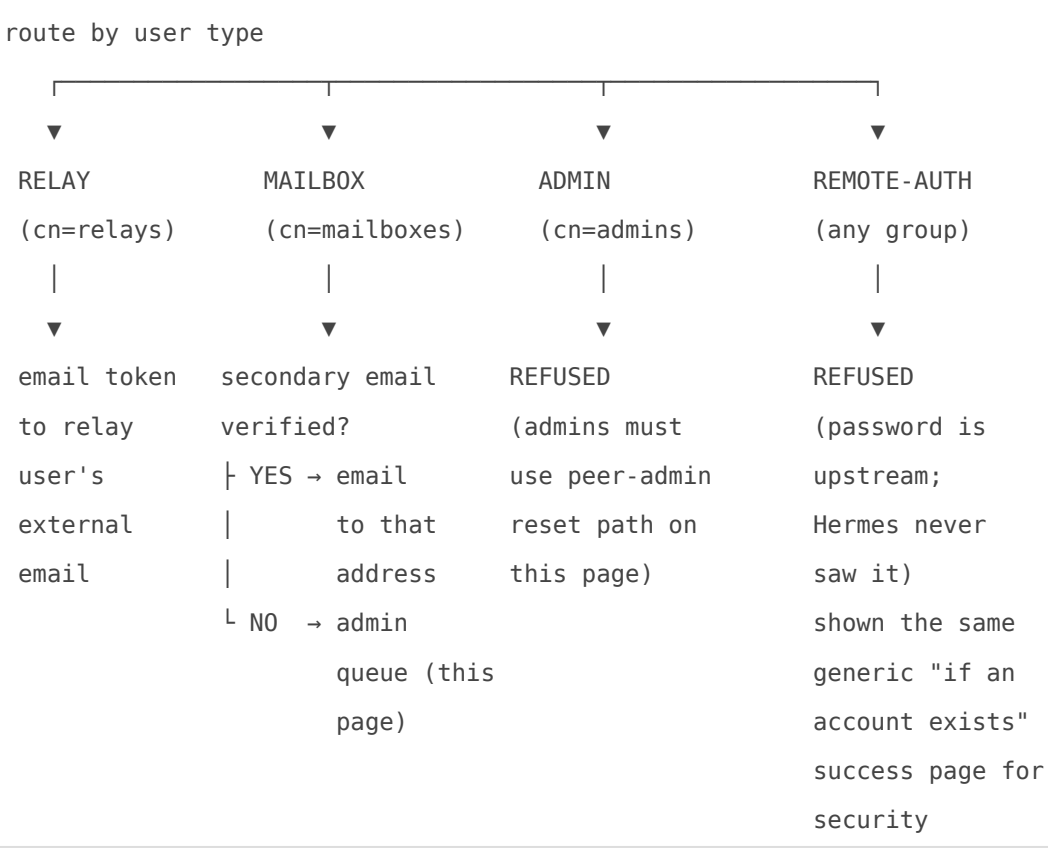
Admin path: **System > Password Resets** (`view_password_reset_requests.cfm`, `inc/process_admin_password_reset.cfm`, `inc/cancel_password_reset_requests.cfm`, `inc/check_hibp.cfm`).

This is the admin-side **queue** for password-reset requests that users have submitted from the public **Forgot Password** page (`/user-auth/forgot_password.cfm`). Most requests resolve themselves via email or Pushover and never need admin attention — the requests that land on this page are the ones that **couldn't** be self-served.

The page is also where an admin can **manually reset** any user's password (mailbox or relay) regardless of how the request arrived — it is the single tool for forcing a password change.

Where a request comes from

```
End user opens /user-auth/forgot_password.cfm
        |   (link from the /users portal login page; same page
        |    serves admin and user portals at the public URL)
        ▼
fills in email + CAPTCHA
        |
        ▼
process_password_reset_request.cfm runs:
  1. honeypot check (hidden field "fax_number_ext" must be empty)
  2. CAPTCHA validation (built-in math OR reCAPTCHA OR
     hCaptcha OR Turnstile — configured globally)
  3. 15-minute rate limit: refuse if a pending request for this
     email exists less than 15 minutes old
  4. LDAP lookup: find the user, determine type from group membership
        |
        ▼
```



The route the request takes determines whether it ever shows up on this page:

Request shape	Lands here?
Relay user with valid email	**No** — email is sent automatically with a 15-minute reset link
Mailbox user with a verified secondary email	**No** — email is sent automatically to the secondary address
Mailbox user with no verified secondary email	**Yes** — admin must reset manually
Mailbox user with Pushover enabled	**No** — Pushover notification sent automatically
Admin self-service	**Never accepted** — admins must be reset by another admin from this page
RemoteAuth user (`auth_type = 'remote'`)	**Never accepted** — Hermes does not own the password (see below)

> **By design.** Admin self-service password reset is blocked because a compromised admin email is an easy lateral-movement vector and the blast radius is the whole console. The forgot-password page shows the same generic "if an account exists, instructions have been sent" message for blocked admins as for blocked RemoteAuth users and for unknown emails — bots probing for admin usernames learn nothing.

RemoteAuth requests are never accepted

For users with `recipients.auth_type = 'remote'` (or, in the future, `mailboxes.auth_type = 'remote'`), the request flow short-circuits at step 4 with the same generic success message as for unknown emails. Hermes does **not** store, hash, or have any way to update the user's password — it lives in the customer's upstream AD/LDAP.

These users must use their organization's own password-reset workflow (self-service portal, helpdesk ticket, etc.). See LDAP RemoteAuth and Credential Model § Local-auth users vs. remote-auth users.

Database schema — `password_reset_requests`

Column	Purpose
`id`	PK
`email`	The address the user typed into the form
`ldap_username`	The `cn` resolved from LDAP at submission time
`user_type`	`relay`, `mailbox`, or `admin` (admin rows shouldn't exist in practice — the flow blocks them at submit)
`token`	64-char random — the secret in the reset link emailed to the user
`notification_method`	`email`, `pushover`, or `admin` — how the user was notified
`status`	`pending`, `completed`, `expired`, `cancelled`
`requested_at`	When the user submitted the form
`expires_at`	NOW + 15 min for `email`/`pushover` methods; NULL for `admin` method (no link to expire)
`completed_at`	When the admin (or self-service flow) resolved it
`completed_by`	The admin username, or the system user that auto-resolved

Auto-cleanup runs on every page load

The page does **not** rely on a scheduled job for housekeeping. Two DELETE queries run at the top of every request:

```
-- Cull expired pending requests (the reset link is dead anyway)
DELETE FROM password_reset_requests
 WHERE status = 'pending'
   AND expires_at < NOW();

-- Cull completed requests older than 30 days (audit window)
DELETE FROM password_reset_requests
 WHERE status = 'completed'
   AND completed_at < DATE_SUB(NOW(), INTERVAL 30 DAY);
```

This keeps the table bounded with no admin intervention. The 30-day audit window is hardcoded — if you need longer retention for compliance, that's a code change, not a configuration knob.

The page surface

Column	Notes
(checkbox)	Only renders for `pending` rows
Email	The user's submitted address
User Type	Badge: relay (info-blue), mailbox (primary), admin (warning)
Method	Icon + label: email envelope, Pushover bell, admin shield
Requested	Submission timestamp
Expires	NULL for admin-method rows; for time-bound rows, shows the timestamp + an "Expired" red badge if past and still pending
Status	pending (yellow), completed (green), expired (gray), cancelled (red)
Completed By	Admin username + timestamp once resolved

Two action buttons sit above the table:

- **Reset Password** — opens the reset modal for the single selected pending row (alerts if zero or more than one is selected)
- **Cancel Request(s)** — opens a confirmation modal that hard-deletes every selected pending row

Why notify-user is shown only for relay rows

The reset modal shows a **Notify user via email** checkbox **only** when the selected row is a relay user. Mailbox and admin users have their primary email == their mailbox address, which won't deliver because the admin is about to change their login credential to a mail-protocol component that's part of the same auth chain. Relay users hold an external email address, so sending them a "your password was reset" notification to that external address works.

Admin reset flow

When the admin clicks **Reset Password** and confirms the modal, `process_admin_password_reset.cfm` runs:

```
1. Form validation: passwords match, length >= 8, request_id present
2. (optional) HIBP check via api.pwnedpasswords.com — k-anonymity
   prefix lookup; reject on match
3. Lookup the row — must still be status='pending'
4. docker exec hermes_ldap slappasswd \
        -o module-load=argon2.la -h {ARGON2} \
        -s <new_password>
        --> returns {ARGON2}$argon2id$...
5. Render /opt/hermes/templates/ldap_modifyuserpassword.ldif
   (THE_USERNAME, THE_OU=users, THE_PASSWORD placeholders)
   to /opt/hermes/tmp/<token>_modifyuserpassword.ldif
6. docker exec hermes_ldap ldapmodify -Y EXTERNAL \
        -H ldapi:///... -f /opt/hermes/tmp/<token>_modifyuserpassword.ldif
7. Delete the temp LDIF
8. If the user has a Nextcloud account (mailboxes.nextcloud_enabled=1):
        docker exec -e OC_PASS=<new> -u www-data hermes_nextcloud \
          php /var/www/html/occ user:resetpassword \
          --password-from-env <email>
   (sync NC's local password column — see Credential Model for why
    NC keeps a local password that no human knows)
9. UPDATE password_reset_requests
        SET status='completed', completed_at=NOW(), completed_by=<admin>
10. UPDATE password_reset_requests SET status='expired'
```

```
        WHERE email=<email> AND status='pending' AND id != <this one>
    (clears stale pending duplicates the user may have submitted)
11. If notify_user checked (relay rows only):
        cfmail via hermes_postfix_dkim:10026 — generic "your password
        was reset by an administrator" template with the console URL
```

Two non-obvious bits:

- **Two hashing tools, one outcome.** This page uses `slappasswd` with the OpenLDAP argon2 module loaded; System Users uses the Authelia CLI image. Both produce `{ARGON2}$argon2id$...` hashes that the same OpenLDAP overlay validates. They are interchangeable; the difference is historical (this page predates the Authelia-image hashing pattern). Either is correct.
- **Nextcloud password sync via temp shell script.** Step 8 writes a shell script to `/opt/hermes/tmp/` and runs it instead of `cfexecute`ing `docker exec` directly. The script wrapper exists because Lucee's `cfexecute` mishandles stderr, quoting, and `OC_PASS` env-var injection on commands of this shape, and the temp-script pattern is the established Hermes workaround.

Cancel flow

`cancel_password_reset_requests.cfm` performs a hard `DELETE` against every selected `pending` row. There is no soft-delete — the row is gone, the user must submit a new request if they still need help. This is the right shape because the request never carried valuable data; it's just a "please help me" signal.

The admin username doing the cancel is **not** recorded — only completions record `completed_by`. If audit trail matters for cancellations, that's a planned schema extension.

CAPTCHA — the public side

The forgot-password page picks a CAPTCHA provider from `system_settings` at runtime. Four providers are supported today:

`captcha_provider`	What appears on the page
`builtin` (default)	Math word-problem ("What is three plus seven?") — no third-party JS, no cookie, no API key required. ~225 unique combinations across addition (1-10), subtraction (1-10, positive result), and small multiplication (1-5).
`recaptcha`	Google reCAPTCHA v2 — site key + secret key required

`captcha_provider`	What appears on the page
`hcaptcha`	hCaptcha — site key + secret key required
`turnstile`	Cloudflare Turnstile — site key + secret key required

All four use the same flow: client-side widget posts a token with the form, server-side `process_password_reset_request.cfm` validates the token (for external providers, via HTTPS POST to the provider's `siteverify` endpoint). Failed validation always redirects back with reason code `9` ("invalid CAPTCHA"). For external providers, if the provider's API is unreachable from Hermes, the page treats the request as invalid — failing closed is the right call on a brute-force defense surface.

A **honeypot** field (named `fax_number_ext`, hidden via CSS) runs **before** the CAPTCHA check. Real users never see or fill it; bots that submit the entire form are silently rejected with the same generic success page so they can't tell their submission was discarded.

Rate limiting — the 15-minute window

`process_password_reset_request.cfm` queries for any `pending` row with the same email submitted in the last 15 minutes; if one exists, the new submission is refused with reason `8`. The window is per-email, not per-IP — a malicious actor enumerating addresses can still hit many emails in parallel, but cannot spam any single one.

The window is hardcoded; if you need longer cool-down for a high-noise environment, that's a code change.

Token security

For email and Pushover methods, the user receives a link of shape:

```
https://<console>/user-auth/reset_password.cfm?token=<64-char-random>
```

- The token is 64 hex chars from `inc/generate_customtrans.cfm` — cryptographically strong, single-use.
- It expires after **15 minutes** (`expires_at` column).
- It is **single-use**: when the user successfully completes the reset, the row's `status` flips to `completed`, and the reset_password.cfm endpoint rejects further use.
- Submitting a new request invalidates any earlier pending request for the same email (step 10 of the admin reset above; the user-side reset endpoint does the equivalent).

For the `admin` method (the rows that show up on this page), the token still exists in the row but the **expires_at is NULL** — there is no email link to expire because no email was sent. The admin resolves the request when they get to it; the queue serves as the notification channel.

What this page does NOT do

Concern	Lives on
Admin's own password change	They sign in to `/admin/`, go to **My Settings** (or have another admin reset it from System Users's edit modal)
Configuring CAPTCHA provider + keys	Configured via `system_settings` rows; admin UI for this is planned. Defaults to `builtin` math CAPTCHA.
Configuring the rate-limit window	Hardcoded 15 minutes — code change required
Configuring the token TTL	Hardcoded 15 minutes — code change required
Pushover credentials per-user	Set on the user portal's **Account Settings** page; this page just consumes them
The reset email template / branding	Hardcoded in `process_password_reset_request.cfm` and `process_admin_password_reset.cfm`; uses `hermes_logo_new_orange2.png` as a CID attachment
2FA device deletion	System Users's **Delete 2FA Devices** button — runs `authelia storage user totp delete`

Failure semantics

What breaks	What happens
`hermes_ldap` down during admin reset	The `slappasswd` and `ldapmodify` calls fail; the admin sees the raw error, the request row stays `pending`, no password change. Retry after LDAP recovers.
`hermes_postfix_dkim` down during user-initiated email request	The cfmail throws; `process_password_reset_request.cfm` catches, flips the request row to `status='failed'`, and shows reason `6` ("Unable to send password reset").
HIBP API unreachable	Server-side check silently passes (the JavaScript on the modal already warned the user; defense-in-depth pattern). The reset still completes.
Token guessed / brute-forced	Computationally infeasible at 64 hex chars (256 bits of entropy).

What breaks	What happens
`hermes_nextcloud` down during admin reset step 8	LDAP password is already updated; the NC sync step fails silently (caught in a non-fatal cftry). The user can log in to `/users` immediately; webmail and DAV will work as soon as NC is back.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_password_reset_requests.cfm`	`hermes_commandbox`	Page (table + 2 modals + auto-cleanup queries)
`config/hermes/var/www/html/admin/2/inc/process_admin_password_reset.cfm`	`hermes_commandbox`	Admin reset handler (LDAP + NC sync + audit + optional notify)
`config/hermes/var/www/html/admin/2/inc/cancel_password_reset_requests.cfm`	`hermes_commandbox`	Hard-deletes selected pending rows
`config/hermes/var/www/html/user-auth/forgot_password.cfm`	`hermes_commandbox`	Public-facing request entry point (CAPTCHA + honeypot + LDAP lookup)
`config/hermes/var/www/html/user-auth/inc/process_password_reset_request.cfm`	`hermes_commandbox`	Rate-limit check + token mint + INSERT + route to email/Pushover/admin
`config/hermes/var/www/html/user-auth/inc/ldap_get_user_groups.cfm`	`hermes_commandbox`	Determines user type from LDAP group membership
`config/hermes/var/www/html/user-auth/reset_password.cfm`	`hermes_commandbox`	Token-consuming endpoint that actually changes the password (user side)
`/opt/hermes/templates/ldap_modifyuserpassword.ldif`	`hermes_commandbox`	LDIF template for the password-replace operation
`/opt/hermes/tmp/<token>_modifyuserpassword.ldif`	`hermes_commandbox`, `hermes_ldap`	Ephemeral rendered LDIF; deleted after `ldapmodify`
`/opt/hermes/tmp/<token>_nc_pwd_update.sh`	`hermes_commandbox`	Ephemeral shell script for the NC `occ user:resetpassword` step
`password_reset_requests` table	`hermes_db_server` (`hermes` DB)	The queue itself

Every shell-out uses `docker exec hermes_ldap …`, `docker exec hermes_nextcloud …`, or the standard `hermes_postfix_dkim:10026` re-injection port per the canonical Hermes pattern.

Related documentation

- System Users — admin-account CRUD; password changes for admins happen there, not on this page
- Credential Model — why mailbox users carry both a web-login password (reset here) and separate per-device app passwords (reset elsewhere)
- LDAP RemoteAuth — why remote-auth users cannot be reset through this page
- Authentication Settings — the Authelia JWT secret used for the reset-link signature on the user-side reset endpoint
- Console Settings — the console hostname embedded in the reset-link emails
- Intrusion Prevention — Fail2ban `authelia` jail; layered defense against brute-force on the login surface this page protects

---

Revision #8
Created 2026-05-31 12:52:00 UTC by Dino Edwards
Updated 2026-05-31 14:01:05 UTC by Dino Edwards