System Users

System Users

Admin path: **System > System Users** (`view_system_users.cfm`, `inc/system_user_actions.cfm`, `inc/ldap_add_user.cfm`, `inc/ldap_add_user_remoteauth.cfm`, `inc/ldap_add_user_groups.cfm`, `inc/ldap_modify_user.cfm`, `inc/ldap_modify_user_password.cfm`, `inc/ldap_change_user_access_control.cfm`, `inc/ldap_delete_user.cfm`, `inc/delete_system_user.cfm`, `inc/delete_system_user_devices.cfm`, `inc/generate_ldap_password.cfm`, `inc/check_hibp.cfm`).

This page manages **admin console operators** — the accounts that can sign in at `/admin/`. Mailbox users (Email Server) and relay recipients (Email Relay) are not managed here even though they share the same underlying LDAP tree; they have their own admin pages.

Each row written by this page lands in **two** stores: the `system_users` table (Hermes DB — UI metadata, auth-type flag, `applied`/`ldap_synced` status), and an LDAP entry under `ou=users,dc=hermes,dc=local` whose group memberships in `ou=groups` give the user actual access. Authelia binds against LDAP for every console login; the DB row exists so the admin UI has something to display and edit.

What this page creates — and what it doesn't

Creates	Doesn't create
Console admin accounts (`cn=admins` group membership)	Mailbox accounts (those go through **Email Server > Mailboxes**, populate `mailboxes` + `cn=mailboxes`)
LDAP entry + DB row in lockstep	Relay-recipient accounts (those go through **Email Relay > Recipients**, populate `recipients` + `cn=relays`)
Local-auth (password lives in Hermes LDAP) **or** RemoteAuth (password lives in upstream AD/LDAP) admins	Authelia-side rows (Authelia is stateless against LDAP — no per-user provisioning needed)
`cn=one_factor` or `cn=two_factor` group membership at create time	The MFA enrolment itself — the user still has to enrol TOTP/WebAuthn/Duo from the user portal's Account Settings page once they sign in

> **Operational consequence.** Every account this page creates is an admin. There is no "create a reader-only admin" or "create an auditor" path today. Granular role assignment is a planned extension; the current model is binary — either you're an admin (full console access) or you're not. The `access_control` column gates one-factor vs. two-factor at the **login gate**, not at the privilege level.

How LDAP membership is structured

System users live under the same OU as every other identity in Hermes, and the user's role is determined by **which groups contain their DN** in the `member` attribute (see Credential Model for the full architecture).

```
dc=hermes,dc=local
├── ou=groups
│   ├── cn=admins              <-- every System User is added here
│   ├── cn=mailboxes           <-- mailbox users (not this page)
│   ├── cn=relays              <-- relay recipients (not this page)
│   ├── cn=one_factor          <-- access_control = one_factor
│   └── cn=two_factor          <-- access_control = two_factor
└── ou=users
    ├── cn=admin               <-- the install-time built-in admin
    ├── cn=jsmith              <-- example local-auth System User
    └── cn=corp_user           <-- example remote-auth stub entry
```

`inc/ldap_add_user_groups.cfm` adds the new System User's DN to **both** `cn=admins` and the chosen access-control group in a single LDIF operation. The LDIF template `/opt/hermes/templates/ldap_addusergroup.ldif` contains two `changetype: modify` blocks that both reference the same `THE_USERNAME` placeholder.

Database schema — `system_users`

Column	Purpose

`id`	PK
`username`	LDAP `cn` / `uid`. Immutable after create (the edit modal renders this field read-only).
`email`	`mail` LDAP attribute; also where forgotten-password notifications would go (but admin self-service reset is **disabled** for security — see Password Resets below)
`first_name`, `last_name`	`givenName`, `sn`
`password`	Argon2id hash with the `{ARGON2}` prefix that OpenLDAP's argon2 overlay expects. Empty string for RemoteAuth users (their password is upstream).
`access_control`	`one_factor` or `two_factor` — drives Authelia's access-control policy at login
`auth_type`	`local` or `remote` — drives the entire create/edit flow
`remoteauth_domain`	For `auth_type = 'remote'`, the `domain_name` key into `remoteauth_mappings`. NULL for local-auth.
`system`	`1` = install-time built-in admin (delete-protected). `2` = admin-created.
`applied`	`1` = current state synced to LDAP. `2` = pending sync (transient during a save).
`ldap_synced`	`1` = LDAP entry exists. `0` = DB row exists but LDAP entry doesn't (a half-sync state the edit handler explicitly detects and tries to repair).
`pushover_user_key`, `pushover_enabled`	Optional Pushover notifications for admin alerts

Local-auth user create flow

```
Admin clicks Create System User
        │
        ▼
form validation: username regex, email format,
first/last name regex, password length 8-64
        │
        ▼ (optional)
HIBP check: SHA-1 prefix sent to api.pwnedpasswords.com
        │   reject if hash suffix matches a known breach
        ▼
generate_ldap_password.cfm
        │   docker run --rm authelia/authelia:VERSION \
        │     authelia crypto hash generate argon2 \
```

```
        |     --password <plaintext>
        |   returns: {ARGON2}$argon2id$v=19$m=...$...$...
        ▼
INSERT INTO system_users (..., password='{ARGON2}...')
        |
        ▼
ldap_add_user.cfm  -- builds adduser LDIF from template,
        |             docker exec hermes_ldap ldapadd
        |             writes entry to ou=users with userPassword
        ▼
ldap_add_user_groups.cfm  -- adds DN to cn=admins
        |                    + cn=<one_factor|two_factor>
        ▼
UPDATE system_users SET ldap_synced = 1
        ▼
session.m = 20  ("System User was created successfully")
```

The Authelia hash generator runs as a **one-shot** `docker run --rm` against the same Authelia image the platform already runs — zero host dependency, format guaranteed to match what Authelia validates at login. The hashing happens in `inc/generate_ldap_password.cfm`.

RemoteAuth user create flow

When the **Authentication Type** dropdown is set to `Remote`, the form shape changes: the password fields disappear and a **RemoteAuth Domain** dropdown becomes required (populated from `remoteauth_mappings` where `enabled = 1`). This option only appears when (a) the install has a Pro license, (b) `remoteauth_settings.enabled = 1`, and (c) at least one enabled mapping exists.

```
INSERT INTO system_users (..., password='', auth_type='remote',
                          remoteauth_domain='<key>')
        |
        ▼
ldap_add_user_remoteauth.cfm  -- writes a stub entry with NO password,
        |                        with seeAlso pointing at the upstream
        |                        DN (expanded from the mapping's
        |                        remote_dn_pattern) and associatedDomain
        |                        set to the mapping key
        ▼
ldap_add_user_groups.cfm  -- adds DN to cn=admins
```

```
+ cn=<one_factor|two_factor>
```

At login, Authelia binds locally against the stub. Hermes's `slapo-remoteauth` overlay sees the `associatedDomain`, finds the matching upstream URI, and rebinds as the `seeAlso` DN. The local entry has no `userPassword` to validate against — the upstream bind is the only decision. See LDAP RemoteAuth for the overlay mechanics.

> **Username uniqueness is global.** The `system_users.username` column is checked for collision across **both** auth types. If your upstream AD already has a user named `dedwards` and Hermes already has a local-auth admin named `dedwards`, the second account cannot be created with the same username. The form's error message suggests `username@domain` or `username.domain` as a workaround.

Edit flow — what can and cannot change

Two fields are **immutable** after create and rendered read-only in the edit modal:

Field	Why immutable
Username	It's the LDAP RDN (`cn=`). Renaming would require a `modrdn` plus updating every group's `member` attribute that references the old DN. The "delete and recreate" path is simpler and safer.
Authentication Type	Switching local-to-remote or remote-to-local would change the LDAP entry's objectClass set (loses or gains a password attribute) and break the `seeAlso`/`associatedDomain` overlay reference. Recreate the user instead.

Everything else is editable: email, first/last name, access-control policy (one/two factor), and — for local-auth users only — the password (via the **Set User Password = YES** toggle which reveals the password fields). The password edit re-runs the same HIBP check and Argon2 hash flow as create.

The access-control change is non-trivial: switching `one_factor` to `two_factor` (or vice versa) means removing the DN from the old group and adding it to the new one. `inc/ldap_change_user_access_control.cfm` handles both ops in sequence.

Half-synced repair

If a previous save crashed between the DB INSERT and the LDAP write (`ldap_synced = 0`, no LDAP entry exists), the edit handler refuses to save the row in a "NO password change" mode — there's no password to push into LDAP. Alert code `16` surfaces the explicit instruction: "set **Set User Password** to YES and enter a new password" so the sync can complete on the next save attempt. The user's stored password is **not** re-pushed because the DB column holds an Argon2 hash, not a plaintext.

Built-in admin protection — the `system` column

The install script seeds a single built-in admin row (the username chosen at install time) with `system = 1`. The page's UI rules:

- **Delete button** is hidden on the row.
- **Cannot delete self**: the row matching `session.userid` also hides its Delete button (a separate check).

Both gates are also enforced server-side in `system_user_actions.cfm`'s `deleteuser` branch — the SQL lookup explicitly filters `system <> '1' AND id <> <session.userid>` so a crafted POST cannot bypass the hidden button.

Delete flow

Soft-delete is **not** the model — the row is physically removed.

```
1. DB lookup: refuse if system='1' or id=session.userid
2. ldap_delete_user.cfm:
     docker exec hermes_ldap ldapdelete \
       cn=<username>,ou=users,dc=hermes,dc=local
     (this auto-removes the DN from any group's member attribute via
      the OpenLDAP referential-integrity overlay)
3. delete_system_user.cfm:
     DELETE FROM system_users WHERE id = <id>
4. delete_system_user_devices.cfm:
     docker exec hermes_authelia authelia storage user totp delete \
       <username> --config /config/configuration.yml
     docker exec hermes_authelia authelia storage user webauthn delete \
       <username> --config /config/configuration.yml --all
```

```
5. session.m = 1  ("System User was deleted successfully")
```

> **Duo Push devices do NOT delete here.** Duo enrolment lives on Duo's cloud servers, not Authelia's database. If the deleted user was Duo-enrolled, the admin must also remove them from the Duo Admin Panel — both the delete and the 2FA-only modals say so explicitly. See Authentication Settings § Duo Security.

Delete 2FA Devices — without deleting the user

The **yellow key** button on each row opens a dedicated **Delete 2FA Devices** modal that runs only step 4 of the delete flow above. Use this when:

- A user reports they've lost their phone / hardware key
- A user is stuck in a 2FA loop after a session expiry
- A user needs to re-enrol with a new TOTP app

After running this, the user is back to a one-factor login state for the next sign-in, then can re-enrol from their Account Settings page. The page waits 5 seconds before redirecting to give Authelia time to flush the credential cache before the success banner appears.

> **Note on Authelia config path.** The two `authelia storage` commands reference `--config /config/configuration.yml`. That is the in-container path, which differs from where you'd expect to find the file from the host's perspective. Authelia's working config inside the container is `/config/configuration.yml`, NOT `/etc/authelia/`. See Authentication Settings § Storage backend — MySQL, not SQLite for why the MariaDB `authelia` database is what actually gets cleaned when these commands run.

have-i-been-pwned (HIBP) check

The **Check Password Against haveibeenpwned.com** toggle (YES/NO, default YES) sends only the first 5 hex chars of the password's SHA-1 to `api.pwnedpasswords.com/range/<prefix>` (k-anonymity: the full hash is never transmitted) and rejects the password if the remaining 35 hex

chars appear in the returned breach list.

If `api.pwnedpasswords.com` is unreachable (no outbound 443, DNS broken, etc.) the create fails with alert `100` — the admin must either restore outbound connectivity or disable the check explicitly on the form. Silently skipping a security check on network failure would be the wrong default.

What this page does NOT do

Concern	Lives on
Mailbox creation	Email Server > Mailboxes — separate table, separate LDAP group
Relay-recipient creation	Email Relay > Relay Recipients — separate table, separate LDAP group
Per-user MFA enforcement (admin-policy flag)	The mailbox / relay-recipient detail pages set `enforce_mfa` for those user classes. System Users use `access_control` instead; if you set it to `two_factor`, Authelia challenges every login. There is no separate "encourage but don't require" middle state for admins — see Authentication Settings § MFA enforcement is decoupled from the cn=two_factor LDAP group.
Password reset queue (admin processes user-initiated requests)	Password Resets
Authelia session length, brute-force throttle, Duo / OIDC	Authentication Settings
Upstream AD/LDAP mapping for RemoteAuth admins	LDAP RemoteAuth — must exist + be enabled before this page's Remote dropdown appears
Pushover token (per-admin alert notifications)	Set on the per-admin notification configuration page; the `pushover_user_key` column on `system_users` is populated there, not here

Failure semantics

What breaks	What happens
`hermes_ldap` container down	Create + Edit fail at the LDAP step. The DB INSERT has already run, so the row exists with `ldap_synced = 0`. Recovery: restart LDAP, edit the user with **Set User Password = YES** to retry the sync (alert `16` will prompt for this on first reload).

What breaks	What happens
`hermes_authelia` container down	Create + Edit + Delete still succeed at the DB + LDAP level; the user can't actually log in until Authelia is back. Delete 2FA Devices fails silently (caught and swallowed in the cftry block) — the next attempt after Authelia recovers will succeed.
HIBP API unreachable with HIBP check ON	Create + password-change Edit refuse to save (alert `100`). The admin must either fix outbound connectivity or set HIBP to NO.
RemoteAuth domain dropdown empty / RemoteAuth disabled	The Remote option doesn't appear in the dropdown at all. To restore: enable a mapping on LDAP RemoteAuth and click Apply Settings.
Username collision	Alert `13` with the suggested `username@domain` or `username.domain` workaround.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_system_users.cfm`	`hermes_commandbox`	Page (table + 4 modals)
`config/hermes/var/www/html/admin/2/inc/system_user_actions.cfm`	`hermes_commandbox`	Action router (create / edit / delete / deletedevices)
`config/hermes/var/www/html/admin/2/inc/generate_ldap_password.cfm`	`hermes_commandbox`	`docker run --rm authelia/authelia ... crypto hash generate argon2`
`config/hermes/var/www/html/admin/2/inc/ldap_add_user.cfm`	`hermes_commandbox`	LDIF render + `ldapadd` for local-auth entries
`config/hermes/var/www/html/admin/2/inc/ldap_add_user_remoteauth.cfm`	`hermes_commandbox`	Stub-entry LDIF render + `ldapadd` for remote-auth entries
`config/hermes/var/www/html/admin/2/inc/ldap_add_user_groups.cfm`	`hermes_commandbox`	Adds DN to `cn=admins` + access-control group
`config/hermes/var/www/html/admin/2/inc/ldap_change_user_access_control.cfm`	`hermes_commandbox`	Moves DN between `cn=one_factor` and `cn=two_factor`
`config/hermes/var/www/html/admin/2/inc/ldap_delete_user.cfm`	`hermes_commandbox`	`ldapdelete` of the user entry
`config/hermes/var/www/html/admin/2/inc/delete_system_user_devices.cfm`	`hermes_commandbox`	`authelia storage user totp delete` + `webauthn delete --all`
`config/hermes/var/www/html/admin/2/inc/check_hibp.cfm`	`hermes_commandbox`	HTTPS GET to `api.pwnedpasswords.com`
`/opt/hermes/templates/ldap_adduser.ldif`	`hermes_commandbox`	Add-user LDIF (placeholder-substituted)

Path	Owner	Role
`/opt/hermes/templates/ldap_adduser_remoteauth.ldif`	`hermes_commandbox`	Stub-user LDIF
`/opt/hermes/templates/ldap_addusergroup.ldif`	`hermes_commandbox`	Two-block LDIF for `cn=admins` + access-control group add
`system_users` table	`hermes_db_server` (`hermes` DB)	Admin metadata + LDAP sync state
`cn=admins,ou=groups,dc=hermes,dc=local`	`hermes_ldap`	Source of truth for who can sign in at `/admin/`

Related documentation

- Credential Model — full four-credential architecture; this page's accounts use only the web-login credential
- LDAP RemoteAuth — required prerequisite for creating remote-auth System Users; covers mappings, DN patterns, TLS settings
- Authentication Settings — Authelia's session lifetime, login regulation, MFA capability vs. enforcement model
- Password Resets — the admin queue for user-initiated reset requests; the page's note on why admin self-service reset is blocked
- Console Settings — `/admin/` hostname, cert, and the IP allowlist that layers above this page's access control

---

Revision #14
Created 2026-05-31 12:52:08 UTC by Dino Edwards
Updated 2026-06-13 12:30:09 UTC by Dino Edwards