Console Firewall

Console Firewall

Pro Edition feature. Maps to **System > Console Firewall** (`view_console_firewall.cfm`, `inc/firewall_action.cfm`, `inc/generate_nginx_configuration.cfm`).

Console Firewall is a **static IP allowlist** for the two admin surfaces of the gateway: the Hermes admin console (`/admin/` and `/admin/2/`) and the Ciphermail web admin (`/ciphermail/`). When enabled, nginx returns `403 Forbidden` to any request for those paths from a source IP not on the list. This is enforced at the nginx layer before Authelia ever sees the request — it's a perimeter filter, not an authentication filter.

How it differs from IPS

Both pages live under System and both touch nginx and ban traffic, so admins routinely confuse them. The distinction is reactive vs. preventative:

	Console Firewall	IPS
Model	Static allowlist (default-deny)	Dynamic blocklist (default-allow)
Layer	nginx `allow`/`deny` directives	iptables drop rules via fail2ban
Scope	`/admin/`, `/admin/2/`, `/ciphermail/` only	All exposed surfaces: SMTP/IMAP, Authelia SSO
Trigger	Admin adds an IP to the list	Failed-auth threshold tripped in a log
Audience	Internal admins / known office IPs	Anyone on the public internet
Storage	`firewall` table + `parameters2.firewall_status`	`intrusion_prevention_jails` + `fail2ban_ips`
Apply	Auto: regen nginx + preload restart on every save	Manual: admin clicks Apply Settings after edits

Both layers stack. A request to `/admin/` from a non-allowlisted IP is rejected by Console Firewall (nginx 403) before fail2ban ever sees an Authelia auth event. A request from an allowlisted IP that then fails login five times still gets the IPS ban from the `authelia` jail.

What's behind the page

```
Browser request to https://<console>/admin/
        │
        ▼
   hermes_nginx  (sites-enabled/<console>_hermes-ssl.conf)
        │
        ├─►  location /admin/ {
        │       allow 10.0.0.5;       ◄── from `firewall` table where hermesadmin='yes'
        │       allow 192.168.1.0/24;
        │       deny all;
        │       ...auth_request /authelia...
        │       proxy_pass http://hermes_commandbox:8888/admin/;
        │    }
        ▼
   Authelia (if allowed)
        ▼
   hermes_commandbox
```

The firewall is **purely an nginx allow/deny block** rendered into the per-console-host vhost. When `firewall_status = enabled`, the rules are present. When `disabled`, the placeholder is rendered as an empty string and nginx falls back to its default allow-all behavior for that location.

Database schema

Table / Column	Role
`firewall.ip`	Single IP address (no CIDR — see the validation note below)
`firewall.hermesadmin`	`'yes'` / `'no'` — include this IP in the `/admin/` allow list
`firewall.ciphermailadmin`	`'yes'` / `'no'` — include this IP in the `/ciphermail/` allow list
`firewall.note`	Free-text annotation surfaced in the table
`firewall.datetime`	Last-modified timestamp
`parameters2` row where `parameter='firewall_status' AND module='firewall'`	Master switch — `enabled` or `disabled`

The schema (`hermes_install.sql` line 812) defines `ip` as `varchar(50)` but the validator at `inc/validate_ip_address.cfm` is a single-address IPv4 regex — there is no CIDR support and no IPv6 support on this page. A 24-bit range needs 256 rows, one per host. For larger ranges, install an upstream firewall instead.

The auto-apply flow

Every action handler in `inc/firewall_action.cfm` (`addip`, `editip`, `deleteip`, `setfirewall`) ends the same way:

1. Update the `firewall` table (or `parameters2.firewall_status` for the master switch).
2. Set a numeric `session.m` alert code (1–7 for errors, 33–37 for success).
3. **Always** include `generate_nginx_configuration.cfm` at the bottom of the file — re-render every per-console vhost from `/opt/hermes/templates/hermes-ssl.conf` with current firewall rules baked in.
4. `cflocation` to `/admin/2/preload_restart_nginx.cfm?returnUrl=/admin/2/view_console_firewall.cfm`.

There is **no "Apply Settings" button** on this page. The Save & Apply button on the master-status card and the row-level edit/delete buttons are themselves the apply — every individual change triggers a full nginx regen and a restart. This is the opposite of the IPS page's batched pending-changes model.

> **Operational consequence.** A burst of edits (adding ten allowed IPs one at a time) triggers ten back-to-back nginx regens, each ending in a restart. The `preload_restart_nginx.cfm` pattern bridges this — the page renders a static "please wait" before the restart fires, then polls until nginx is back, so the admin's own session doesn't `ERR_CONNECTION_REFUSED` mid-redirect. There is no batch-add path; bulk imports are an `INSERT INTO firewall ...` SQL job followed by one manual Save & Apply on the status card.

Template placeholders

`generate_nginx_configuration.cfm` queries `firewall` twice and renders two placeholder substitutions into the per-vhost rendered file:

Template token	Substituted with	Used in

`hermes_fw_hermes`	`allow <ip>;` lines for every `firewall` row where `hermesadmin='yes'`, terminated by `deny all;`	`location /admin/ { ... }` block (template line 157)
`hermes_fw_ciphermail`	`allow <ip>;` lines for every `firewall` row where `ciphermailadmin='yes'`, terminated by `deny all;`	`location /ciphermail/ { ... }` block (template line 287)

When the firewall is disabled, both placeholders are blanked out — the `location` blocks render without any `allow`/`deny` and nginx falls back to its default allow-all. When the firewall is enabled but **no row** has the relevant flag set to `yes`, the recordcount-zero branch in the generator also blanks the placeholder. There is no "deny everyone" mode that locks the page from itself; see the safety checks below.

The `/users/`, `/nc/`, `/main/`, `/plugins/`, and `/web/` locations are **not** firewalled by this page — they have no `hermes_fw_*` placeholder. Mailbox users, Nextcloud users, and Ciphermail end-user portal users hit Authelia directly with no IP allowlist. This is deliberate: those are end-user surfaces, not admin surfaces.

Safety checks — the four guardrails

Without protection, an admin could trivially lock themselves out of the gateway by deleting their own IP, editing it to something wrong, or enabling the firewall before adding their own IP. `inc/firewall_action.cfm` carries four guard rules (each tied to its own alert code):

Guard	When it fires	Alert
Can't delete own IP while firewall enabled	`getip.ip = ClientIP AND firewall_status = enabled` on `deleteip`	`m=3`
Can't edit own IP while firewall enabled (unless the new IP is also the client's IP)	Same condition on `editip` with a different new IP	`m=4`
Can't enable firewall unless current IP is in the list with `hermesadmin='yes'`	`setfirewall` to `enabled` with no matching `firewall` row for `ClientIP`	`m=5`
Duplicate IP rejected on add/edit	Unique-IP check by query	`m=2`, `m=6`

`ClientIP` is set in `Application.cfc` from the `X-Forwarded-For` header (nginx sets it from `$remote_addr`). When testing behind a load balancer or VPN, what the page considers "your IP" may not match what your laptop reports — verify with the per-row table what nginx is actually seeing before clicking the master enable.

The recovery path when locked out

If a misconfiguration locks the admin out anyway (forgotten to add the new office IP, master flipped before the row was saved, browser using an unexpected egress IP), the recovery sequence is shell-level on the Docker host:

```
# Disable the firewall directly in the DB
docker exec hermes_db_server mariadb -u root hermes -e \
    "UPDATE parameters2 SET value2='disabled' \
     WHERE parameter='firewall_status' AND module='firewall'"

# Add the new admin IP
docker exec hermes_db_server mariadb -u root hermes -e \
    "INSERT INTO firewall (ip, hermesadmin, ciphermailadmin, note) \
     VALUES ('<your-ip>', 'yes', 'yes', 'Recovery add')"

# Trigger a manual nginx regen by hitting the page from inside the CommandBox container
docker exec hermes_commandbox curl -s
http://localhost:8888/admin/2/inc/generate_nginx_configuration.cfm

# Reload nginx
docker exec hermes_nginx nginx -s reload
```

The MariaDB call uses unix-socket auth (root via the container) — no password, by design. Once back in, re-enable the firewall from the UI so the lockout-guard alerts are restored.

A planned Hermes CLI Management Console (`scripts/hermes-cli.sh`) will wrap this recovery into a menu option. Until it ships, the docker-exec sequence above is the supported recovery path.

Interaction with Console Settings

The console hostname change (`edit_console_settings.cfm`) regenerates the same per-console nginx vhost from the same template — meaning a hostname change automatically picks up the current Console Firewall state. The Firewall rules carry over to the new vhost transparently; the admin does not need to revisit this page after a hostname change.

The reverse is not true: editing the Firewall does not change the hostname. But because `firewall_action.cfm` always calls `generate_nginx_configuration.cfm`, which always renders every active console vhost, a stale-vhost scenario (where an old hostname's vhost still exists alongside

the new one) gets both vhosts re-rendered on a Firewall save. This is fine in practice; it's been the established behavior since the AdminLTE 4 refactor (`a348e73f`).

License gating

The page is wrapped in the standard Pro check:

```
<cfif NOT isDefined("session.edition") OR session.edition NEQ "Pro">
    <cfset proFeatureName = "Admin Console Firewall">
    <cfinclude template="./inc/license_pro_required.cfm">
    <cfabort>
</cfif>
```

Community installs see the gating panel. The `firewall` table and `parameters2.firewall_status` row exist anyway (they're seeded); pre-existing rules continue to render into the nginx vhost as long as `firewall_status='enabled'`. Switching from Pro to Community does **not** auto-disable the firewall — if it was on when the license downgraded, it stays on. To turn it off, an admin needs to either reactivate Pro or use the recovery path above.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_console_firewall.cfm`	`hermes_commandbox`	Main page + modals
`config/hermes/var/www/html/admin/2/inc/firewall_action.cfm`	`hermes_commandbox`	All add/edit/delete/status handlers; auto-applies via the nginx regen include
`config/hermes/var/www/html/admin/2/inc/generate_nginx_configuration.cfm`	`hermes_commandbox`	Renders `hermes_fw_hermes` and `hermes_fw_ciphermail` placeholders
`config/hermes/var/www/html/admin/2/inc/validate_ip_address.cfm`	`hermes_commandbox`	IPv4 single-address regex (no CIDR, no IPv6 on this page)
`config/hermes/var/www/html/admin/2/preload_restart_nginx.cfm`	`hermes_commandbox`	Pre-restart splash + polling rejoin so the admin's session survives the reload
`config/hermes/opt/hermes/templates/hermes-ssl.conf`	`hermes_commandbox`	nginx vhost template with the `hermes_fw_*` tokens
`config/nginx/etc/nginx/sites-available/<token>_hermes-ssl.conf`	`hermes_nginx` (mounted)	Live rendered vhost — what nginx actually serves

Related

- IPS — the reactive blocklist that complements this preventative allowlist
- Console Settings — hostname changes regenerate the same vhost and pick up Firewall state automatically
- Authentication Settings — Authelia runs after Console Firewall passes; both layers stack
- LDAP RemoteAuth — RemoteAuth admins still hit Console Firewall first; the upstream LDAP bind only matters once the request reaches Authelia

---

Revision #12
Created 2026-05-31 12:51:54 UTC by Dino Edwards
Updated 2026-06-11 15:04:14 UTC by Dino Edwards