System Update

System Update

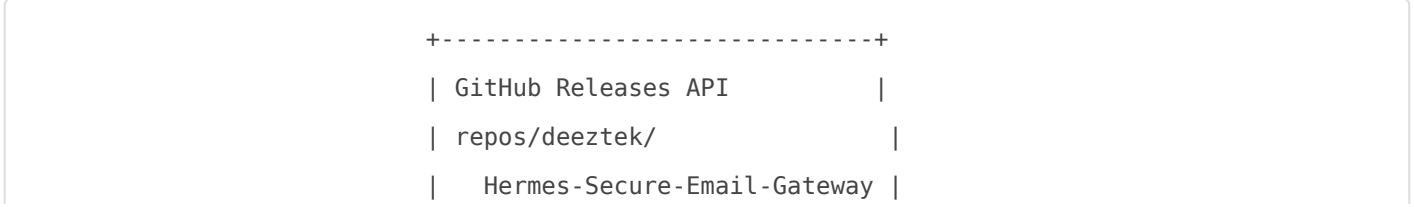
Admin path: **System > System Update** (`view_system_updates.cfm`). Update infrastructure: `config/hermes/var/www/html/schedule/check_for_update.cfm` (daily GitHub Releases poll), `config/hermes/var/www/html/admin/2/inc/check_system_update.cfm` (dashboard cache-file reader), `scripts/system_update_docker.sh` (the update orchestrator), `config/ofelia/config.ini` (the cron schedule that triggers the daily check).

This page tells an admin **whether a new Hermes release is available and how to apply it**. It is intentionally thin: every detail of how upgrades actually work — the artifact taxonomy, the orchestrator's five phases, the idempotency rules, the release-cut procedure — lives in Release and Update Methodology, which is the canonical reference. This page documents the **admin surface** that sits on top of that methodology.

> **Update is currently CLI-driven.** The page itself displays a notice that points at the docs and the release-and-update methodology; the actual upgrade is run on the Docker host via SSH using `scripts/system_update_docker.sh`. A future revision will move the launch button into the page itself; until it does, the CLI is the supported path.

How an admin knows there is an update

Three independent surfaces converge on the same answer:



All three are downstream of one cached value — the `/opt/hermes/updates/check_system_update.txt` file. The dashboard does not call GitHub on page load; the email is not sent on page load; only the once-a-day Ofelia job actually hits the API.

Daily update check

`config/ofelia/config.ini` schedules a single `job-exec` against the `hermes_commandbox` container:

```
[job-exec "hermes-update-check"]
schedule =  0 30 04 * * *
container = hermes_commandbox
command = /opt/hermes/schedule/update_check.sh
```

The shell wrapper resolves to a `curl --silent http://localhost:8888/schedule/check_for_update.cfm` against the internal Lucee port — no auth dance, no X-Token header, same convention as `hermes-message-cleanup`

, `hermes-quarantine-notify`, and every other Hermes scheduled job. The CFML target does the actual work:

1. Read current `build_no` from `system_settings`.
2. `GET https://api.github.com/repos/deeztek/Hermes-Secure-Email-Gateway/releases/latest` with a 30s timeout.
3. On HTTP 200, parse `tag_name` and compare to the local build via simple string comparison (`vYYMMDD` sorts correctly as a string because the format is fixed-width calendar versioning — see Release and Update Methodology § Calendar versioning).
4. Write `/opt/hermes/updates/check_system_update.txt` regardless of outcome — the dashboard reader needs **something** to display.
5. On `UPDATEFOUND`, send one notification email to `admin_email`.

Cache file format

The file is a single `@`-delimited line. The format is preserved from the pre-#218 legacy update server (`updates.deeztek.com`) for backward-compat with the dashboard reader; for Docker installs, several fields are unused.

Position	Field	Docker meaning
1	status	`SUCCESS` (update available), `NOUPDATE`, or `UPDATE CHECK UNAVAILABLE`
2	build	The new tag (e.g. `v260601`) on `SUCCESS`, current tag on `NOUPDATE`
3	released	`yyyy-mm-dd` from `published_at`
4	filename	*empty* (was tarball name on legacy server)
5	release_notes_url	GitHub `html_url` for the release
6	release_notes_file	*empty* (was per-release HTML file on legacy server)
7	mysqlroot	*empty* (was installer credential on legacy server)
8	dev	`daily_update_check` value from `system_settings`

Email notification

The notification is **once per release** — re-runs of the check against the same latest tag do not re-send (the job re-detects `UPDATEFOUND` every day, but the email path is gated on the cached comparison; if the dashboard cell already reads `UPDATEFOUND`, the admin is already informed). The email is sent through `hermes_postfix_dkim` on port 10026 (the post-content-filter re-injection port

that auto-DKIM-signs), so the message is signed under the gateway's own DKIM key like any other system mail.

The message includes a GitHub link and, when `console.host` is set, a hint to open the admin console where the dashboard prompt is waiting.

Toggling the daily check

The `daily_update_check` row in `system_settings` is wired through to the cache file (field 8 above), but the Ofelia schedule itself is the actual on/off switch — to stop the daily check, remove or comment the `[job-exec "hermes-update-check"]` block in `config/ofelia/config.ini` and restart `hermes_ofelia`. The `system_settings` toggle is a legacy UI surface from the pre-Ofelia era; the modern path is the Ofelia config.

Status values shown on the dashboard

The dashboard's **Hermes Update** cell (System Info card, last column) is the operator-visible side of this whole pipeline. See also System Status § System Info card.

Cache status	Cell text	What it means
`SUCCESS`	`UPDATE BUILD vYYMMDD FOUND` (link → release-notes modal)	New release available. Click for GitHub release notes; act via the orchestrator below.
`NOUPDATE`	`LATEST VERSION`	Local `build_no` matches `tag_name` on GitHub.
`UPDATE CHECK UNAVAILABLE`	`UPDATE CHECK UNAVAILABLE`	GitHub API call failed (rate limit, offline, DNS). Check `hermes_update_check` log on `hermes_commandbox`.
(cache file missing)	`UPDATE CHECK PENDING`	First-ever render before the 04:30 job has run. Wait one cycle or invoke manually (below).

Running the update

Today (CLI)

The page is currently a notice that delegates to the docs. To actually apply an update, SSH to the Docker host and run the orchestrator:

```
cd /opt/hermes-seg-docker-gl
./scripts/system_update_docker.sh                # apply latest
./scripts/system_update_docker.sh v260601        # apply a specific tag
./scripts/system_update_docker.sh --dry-run      # show what would run, change nothing
./scripts/system_update_docker.sh --skip-git     # containers + artifacts only
./scripts/system_update_docker.sh --skip-compose # git + artifacts only
./scripts/system_update_docker.sh -y             # don't prompt for confirmation
```

The orchestrator walks five phases. For the full breakdown of each phase — preflight, code pull, container update, per-release artifact application, finalize, and the persistent post-upgrade hook — see Release and Update Methodology § The update orchestrator. For the categories of artifact the orchestrator applies (baseline vs per-release vs persistent hook), see § Artifact taxonomy.

A condensed version of what the orchestrator does:

Phase	What it does	Idempotent?
Preflight	Refuses to run if working tree dirty, `hermes_db_server` down, or target older than current	Trivially
1 — Pull new code	`git fetch --tags` + `git checkout <tag>`	Yes
2 — Update containers	`docker compose pull` + `docker compose up -d`	Yes; only restarts services whose image or config changed
3 — Apply per-release artifacts	Walks `updates/v*/` directories newer than current `build_no`, applies `sql/` → `cfml/` → `scripts/` in order; each release's `schema_updates.sql` advances `build_no` at its end	Yes (every artifact must be idempotent — see methodology doc)
4 — Finalize	Restarts `hermes_commandbox`; logs reminders for `occ upgrade` (if `NCVERSION` bumped) and `*.HERMES` template re-render	Yes
5 — Post-upgrade hook	`curl http://localhost:8888/schedule/post_upgrade.cfm` — runs any persistent migrations gated by the `migrations` table	Yes (per-block gated)

Output is teed to a timestamped log under `install-logs/`: `install-logs/hermes_update_YYYYMMDD_HHMMSS.log`. If anything fails, the orchestrator aborts (`set -e`); inspect the log, fix the underlying issue, and re-run. Idempotency makes mid-upgrade resume safe — a failed Phase 3 picks up at the same release on the next run and re-applies its full artifact set; `IF NOT EXISTS` and `INSERT IGNORE` guards turn the second pass into a no-op.

Tomorrow (in-page button)

The page is positioned to grow a "Check Now" button (force-runs the daily check ahead of schedule) and a "Run Update" button (invokes the orchestrator via a CFML wrapper). Neither is wired today; the infrastructure they would call is already in place.

Track this in #221.

Forcing a manual check

If you cannot wait for the 04:30 UTC schedule (e.g., a release just shipped and you want the dashboard to update now), invoke the same endpoint Ofelia does:

```
docker exec hermes_commandbox curl --silent
http://localhost:8888/schedule/check_for_update.cfm
```

The response is the literal string `OK` and the cache file is rewritten in place. The dashboard picks it up on the next page load.

The same invocation is what Ofelia would have run at 04:30 — there is no difference between manual and scheduled execution.

The version stamp

What the orchestrator and the dashboard both compare against is the **`build_no`** row in `system_settings`:

Setting	Value	Set by
`version_no`	`Docker`	Baseline (`hermes_install.sql`) on fresh install; never changes in the Docker era

Setting	Value	Set by
`build_no`	`vYYMMDD`	Baseline at install; advanced by each release's `updates/v<DATE>/sql/schema_updates.sql` at its very end

A successful Phase 3 ends with `build_no` matching the target tag. If after an orchestrator run those two disagree, something in Phase 3 silently no-op'd a stamp-advance — inspect the log. See Release and Update Methodology § The release-cut procedure for the exact `UPDATE system_settings ...` block every release's `schema_updates.sql` ends with.

Skipping releases

The orchestrator handles release-skipping natively. Upgrading from `v260119` straight to `v260801` (skipping a hypothetical intermediate `v260601`) walks **both** release directories in order during Phase 3 — `v260601/` first, then `v260801/`. `build_no` advances after each release's `sql/` step, so the in-between cursor advancement is safe.

> **Operational consequence.** Releases are designed to be applied in chronological order; skipping is supported (and tested) but is not the optimized path. If you upgrade rarely, expect Phase 3 to take proportionally longer the further behind you are.

Failure semantics

What breaks	What happens
GitHub Releases API unreachable	`UPDATE CHECK UNAVAILABLE` in dashboard cell; cached value is overwritten with the unavailable marker. Logged to `hermes_update_check`.
GitHub Releases API rate-limited (HTTP 403 or 429)	Same as unreachable — anonymous polling is subject to GitHub's 60 req/hr per-IP limit. The daily schedule keeps usage trivial; the only way to hit the limit is repeated manual invocations.
`/releases/latest` returns 404 (no qualifying release on the repo)	Treated as `NOUPDATE`, not an error — the repo simply hasn't shipped its first qualifying release yet.
`published_at` in API response fails `ParseDateTime`	Falls back to the raw ISO string in the cache file — non-fatal.

What breaks	What happens
`cfmail` notification fails	Logged to `hermes_update_check`; cache file write proceeds (notification is best-effort).
Cache file cannot be written (`/opt/hermes/updates/` not writable)	Logged; the dashboard falls through to `UPDATE CHECK PENDING`.
Orchestrator Phase 1 fails (tag not pushed, dirty tree)	Aborts before touching containers or DB. Working tree is unchanged.
Orchestrator Phase 2 fails (image pull error, registry unreachable)	Aborts; previous containers keep running with their existing images. Re-run after fixing the registry / network issue.
Orchestrator Phase 3 fails on a SQL artifact	Aborts; `build_no` reflects whatever the last successful release's stamp set it to. Re-run picks up at the failed release; idempotency guards re-apply the partial work safely.
Orchestrator Phase 5 fails	Logged as a warning, **not** treated as fatal — the orchestrator exits 0. Run `post_upgrade.cfm` manually after fixing the underlying issue: `docker exec hermes_commandbox curl --silent http://localhost:8888/schedule/post_upgrade.cfm`

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_system_updates.cfm`	`hermes_commandbox`	The admin page (notice + future Run Update wiring)
`config/hermes/var/www/html/admin/2/inc/check_system_update.cfm`	`hermes_commandbox`	Reads the cache file for the dashboard cell
`config/hermes/var/www/html/schedule/check_for_update.cfm`	`hermes_commandbox`	Daily poll target
`config/ofelia/config.ini` (`hermes-update-check` job)	`hermes_ofelia`	Schedules the daily poll
`scripts/system_update_docker.sh`	host shell	The update orchestrator
`scripts/install_hermes_docker.sh --apply-schema`	host shell	Legacy pre-orchestrator schema-apply path; superseded by the orchestrator but still functional for emergency manual use
`/opt/hermes/updates/check_system_update.txt`	`hermes_commandbox`	Cache file; format above
`install-logs/hermes_update_<timestamp>.log`	host filesystem	Orchestrator output, teed live

Path	Owner	Role
`system_settings.build_no` / `system_settings.version_no`	`hermes_db_server` (`hermes` DB)	The version stamp the orchestrator and the dashboard both read
`migrations` table	`hermes_db_server` (`hermes` DB)	Tracks which Phase 5 migration blocks have run; see Methodology § Phase 5
`updates/v*/sql/schema_updates.sql`	repo working tree	Per-release SQL deltas; one of three artifact categories
`updates/v*/cfml/*.cfm`	repo working tree	Per-release CFML migrations (encryption / file IO / API calls)
`updates/v*/scripts/*.sh`	repo working tree	Per-release host-shell one-shots
`config/hermes/var/www/html/schedule/post_upgrade.cfm`	`hermes_commandbox`	The persistent cross-release migration hook

Related

- Release and Update Methodology — **the canonical reference** for everything covered on this page. Read it before adding a schema change, a one-shot migration, a service config edit, or cutting a release tag.
- System Status — the dashboard that surfaces the **Hermes Update** cell this page's daily check populates
- System Settings — `admin_email` (target of the update-found notification email), `postmaster` (sender), and the legacy `daily_update_check` toggle
- Scheduled Tasks — the admin surface over the Ofelia config that schedules the daily check
- System Logs — where `hermes_update_check` log entries surface for debugging failed polls
- Storage Topology — the four storage tiers an upgrade touches (Config tier is where `git checkout` runs; Data tier holds `/opt/hermes/updates/`)

Revision #8
Created 2026-05-31 12:52:07 UTC by Dino Edwards
Updated 2026-05-31 14:01:10 UTC by Dino Edwards