DNS Resolver

DNS Resolver

Admin path: **System > DNS Resolver** (`view_dns_resolver.cfm`, `inc/dns_resolver_action.cfm`, `inc/generate_unbound_forward_conf.cfm`, `inc/generate_unbound_local_conf.cfm`).

Hermes ships its own **recursive caching DNS resolver** — a stock `hermes_unbound` container fronted by an admin UI that lets an operator toggle recursive vs. forwarding mode, manage upstream forwarders, add local-zone overrides for split-horizon hostnames, inspect cache statistics, and run ad-hoc lookups. Every other Hermes container points its `dns:` at `hermes_unbound` (`${IPV4SUBNET}.117`) rather than the host's resolver, so RBL/DNSBL lookups, MX resolution, ARC verification, Postfix recipient validation, and OIDC discovery all flow through this single resolver.

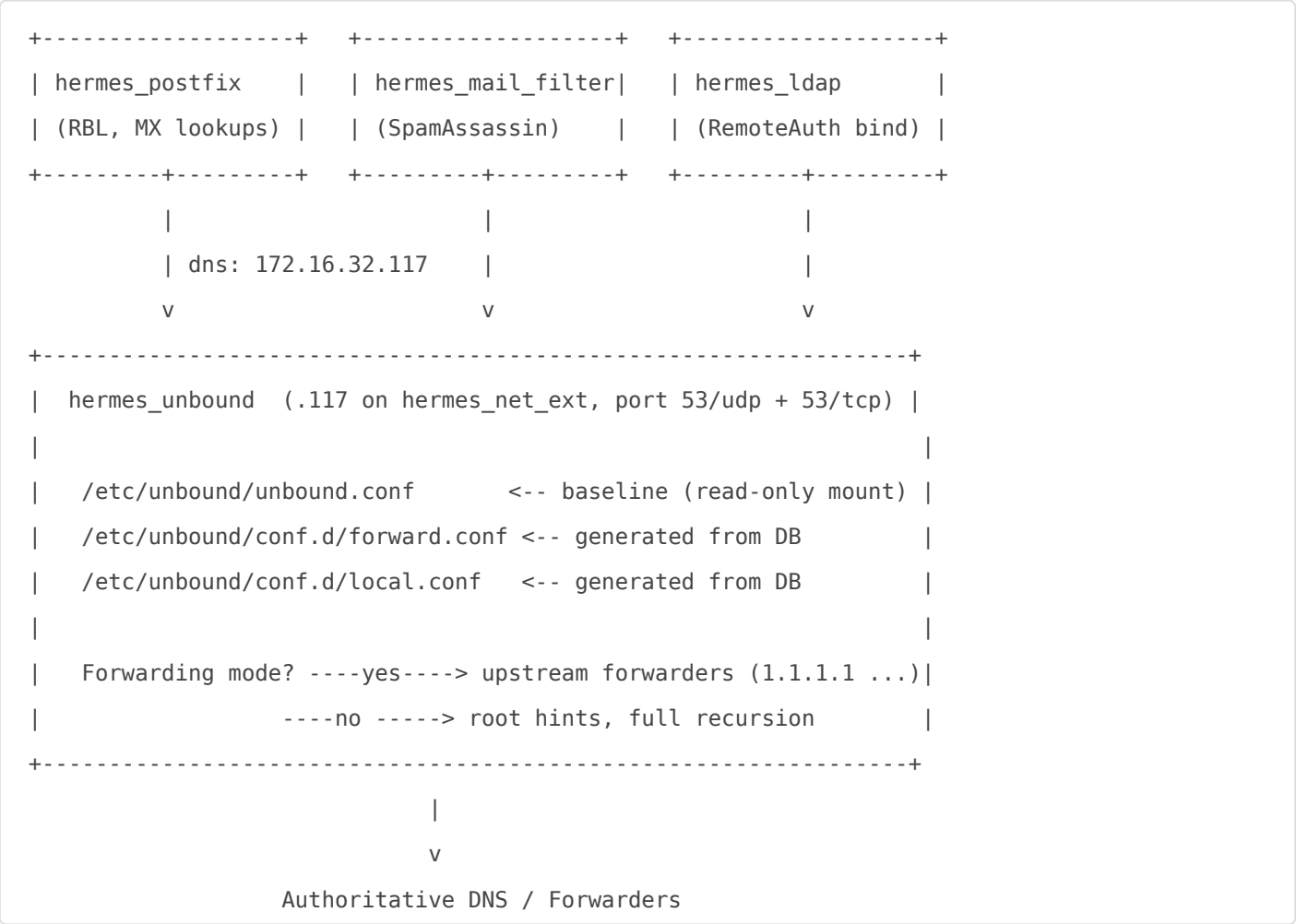
Why Hermes runs its own resolver

A mail gateway has DNS requirements that a stock host resolver does not meet:

Requirement	Why a shared resolver fails	What Unbound gives Hermes
RBL/DNSBL queries from a low-volume IP	Public resolvers (Cloudflare, Google, Quad9) issue thousands of queries per second on behalf of many tenants. RBL providers throttle or refuse responses to those shared IPs.	Recursive mode queries the authoritative servers directly from the gateway's own IP — well under any per-source rate limit.
Deterministic resolution path for DKIM / DMARC / ARC	A flaky host resolver causes intermittent `TEMPFAIL` on DNS-dependent auth	Unbound's cache survives container restarts of the consumers, and its TTLs are tuned for mail traffic
Split-horizon DNS (internal AD hostnames)	The host's `/etc/resolv.conf` typically points at public DNS — internal-only names fail	The Local DNS Overrides table writes `local-data` entries that Unbound returns for any container that asks
DNSSEC validation across the stack	Trust depends on every container running its own validator (rarely the case)	Unbound validates once; consumers get verified answers automatically

The container itself is custom-built (Hermes-published image at `ghcr.io/deeztek/hermes-unbound`) but the configuration is plain Unbound — there is no Hermes-specific patching at the daemon level.

How DNS flows through the stack



Every container declares `dns: ${IPV4SUBNET}.117` in `docker-compose.yml` so its `/etc/resolv.conf` points at the Unbound container regardless of the host's resolver configuration. The host itself is unaffected.

Configuration storage

Forwarding mode and the forwarders/local-records lists live in three places:

Setting	Storage	Notes
Forwarding mode	`parameters2.module = 'unbound', parameter = 'forwarding.enabled'`	`yes` or `no`
Upstream forwarders	`dns_forwarders` table	`server`, `port`, `tls`, `enabled`, `sort_order`; seeded with Cloudflare (1.1.1.1 / 1.0.0.1) + Google (8.8.8.8 / 8.8.4.4)

Setting	Storage	Notes
Local DNS overrides	`dns_local_records` table	`hostname`, `record_type` (A/AAAA/CNAME/MX/TXT/PTR), `value`, `enabled`, `description`; UNIQUE on `(hostname, record_type)`

The baseline `unbound.conf` (cache sizes, DNSSEC trust anchor, num-threads, access-control for the Docker subnets) ships as a read-only mount and is not editable from this page. To change those, edit `config/unbound/unbound.conf` directly and restart the container.

Recursive vs. forwarding mode

The default is **Recursive** and the in-page callout pushes hard against flipping it. The reasoning is operational, not philosophical:

> **Forwarding through public resolvers will cause RBL/DNSBL lookup failures.** When queries are forwarded through Cloudflare / Google / Quad9, your blocklist lookups originate from their shared IP addresses. RBL providers throttle or block these IPs because thousands of other customers are making the same queries from the same resolvers. With recursive resolution, queries come from your server's own IP, keeping you well under per-source rate limits.

Forwarding is still useful in a few specific cases:

- Egress-restricted networks where outbound port 53 to arbitrary authoritative servers is blocked but a known forwarder is allowed
- Compliance requirements forcing all DNS through a logged corporate resolver
- DNS-over-TLS to a specific provider (set `tls = yes` and `port = 853` on each forwarder)

In any of those cases, configure forwarders that you control or that have a known per-customer SLA. Public flat-rate resolvers cause RBL breakage that surfaces days later as inflated spam scores.

The four cards on the page

1. DNS Resolver Status

Shows container state (`running`, `exited`, or `error`) via `docker inspect --format='{{.State.Status}}|{{.State.StartedAt}}'`, computes the uptime in days/hours/minutes (the

StartedAt timestamp is UTC; the page converts before diffing — earlier versions had a tz-drift bug, see commit `644d56b1`), and exposes a **Restart Unbound** button.

> **Restarts are mail-safe.** Restarting `hermes_unbound` typically takes 1–3 seconds. During that window, consumer containers fall back to retry; Postfix, Amavis, and Dovecot all tolerate a brief DNS outage without losing mail. Plan restarts freely; you do not need an outage window.

2. DNS Forwarding

Two sub-controls. The **DNS Resolution Mode** select (recursive vs. forwarding) writes `parameters2.unbound.forwarding.enabled` and regenerates `forward.conf`. The **Upstream Forwarders** table is the working set used when forwarding is enabled — fields are Server IP, Port (default `853` for DoT, `53` for plain), TLS (yes/no), and per-row enable/disable + delete.

The two-step "edit then Apply" model is deliberate: adding, deleting, or toggling a forwarder marks the change pending (the page banner shifts to amber) but does **not** restart Unbound. Click **Apply & Restart Unbound** to regenerate `forward.conf` and bounce the container in one shot. This lets an admin batch a multi-row change without triggering multiple restarts.

3. Local DNS Overrides

A static-entries table that becomes `local-data` lines in `/etc/unbound/conf.d/local.conf`. The same two-step edit-then-Apply model applies.

This is the single most operationally important card on the page. Two canonical use cases:

Scenario	What to add
LDAP RemoteAuth against an internal AD DC (`dc01.corp.example.com`) that is not publicly resolvable	`dc01.corp.example.com` → `10.0.0.10` (A record). See LDAP RemoteAuth § DNS resolution prerequisite.
Split-horizon: the Console Address resolves externally but you want internal containers to skip the public lookup	`console.example.com` → `192.168.1.10` (A record)

The generator groups records by their second-level zone and emits a single `local-zone: "<zone>." transparent` declaration before the `local-data` lines — `transparent` means Unbound resolves the configured hostnames locally but **forwards everything else in the same zone** upstream as normal. This is the right choice for split-horizon: an override for `dc01.example.com` does not break public lookups for `www.example.com` against the same zone.

> **Operational consequence.** A misconfigured override can shadow a public hostname. Hermes resolves what you write — if you point `mail.example.com` at the wrong internal IP, every container that asks for that name will get the wrong answer. Test with the **DNS Lookup Test** card (below) before relying on the entry in production.

4. DNSSEC, Cache Statistics, DNS Lookup Test

Three read-only utility cards.

Card	What it shows / does
DNSSEC	Parses the live `unbound.conf` inside the container; reports Enabled / Disabled based on `auto-trust-anchor-file` / `trust-anchor-file` / `module-config: validator` presence. **Test DNSSEC** runs `drill -D example.com` and dumps the response. DNSSEC is enabled in the shipped baseline; this card is informational.
Cache Statistics	Runs `unbound-control stats_noreset` and parses `total.num.queries`, `cachehits`, `cachemiss`, `prefetch`, plus RRset/message cache counts and average recursion time. Useful for diagnosing cold-cache latency after a restart. **Flush Cache** clears the entire cache (`unbound-control flush_zone .`) — typically used after a downstream DNS record change that you don't want to wait for the TTL on.
DNS Lookup Test	Runs `drill @127.0.0.1 <TYPE> <name>` inside the container. Supports A / AAAA / MX / TXT / NS / SOA / PTR. Input is validated to `[a-zA-Z0-9.\-]+` before being passed to the shell. This is the right tool to verify a local override actually took effect.

Apply flow

A single Save / Apply click runs roughly this:

```
1. Validate input (IP octets in range, port 1-65535, hostname charset, ...)
2. UPDATE or INSERT INTO parameters2 / dns_forwarders / dns_local_records
3. cfinclude generate_unbound_forward_conf.cfm   (or _local_conf.cfm)
        - Read the table back
        - Render the conf into chr(10)-newline plain text
```

```
        - fileWrite("/etc/unbound/conf.d/forward.conf", ..., "utf-8")
4. cfexecute /usr/local/bin/docker container restart hermes_unbound
        (30s timeout; typically returns in 1-3s)
5. cflocation back to view_dns_resolver.cfm with session.m set
```

The generated `conf.d/*.conf` files are written via Lucee `fileWrite` into the `hermes_commandbox`-side bind-mount of `config/unbound/conf.d/` — the same directory `hermes_unbound` reads on restart. There is no `docker cp` step; both containers see the same files because they share the bind mount (commit `06acd4e1` switched away from the legacy `docker cp` pattern).

Cache TTL behavior

The baseline `unbound.conf` sets:

Knob	Value	Why
`cache-min-ttl: 300`	5 minutes	Floor — protects against authoritative servers that publish ultra-short TTLs
`cache-max-ttl: 86400`	24 hours	Ceiling
`cache-max-negative-ttl: 900`	15 minutes	Floor on NXDOMAIN cacheing — important for DNSBL hits, which produce intentional NXDOMAINs
`prefetch: yes`	—	Refreshes hot records before TTL expiry so cache misses are rare
`qname-minimisation: yes`	—	Privacy + reduces authoritative-server query volume

After a record change you depend on (e.g., updating an MX record at the registrar), use **Flush Cache** to skip the wait.

Failure semantics

What breaks	What happens
`dns_forwarders.server` validation fails (non-IPv4, octet > 255, port out of range)	`session.m = 11`, redirect, no DB write. Error text in the alert.
`dns_local_records.hostname` empty or invalid record type	Same — `session.m = 11` with specific error text.
`fileWrite` to `conf.d/` fails	`session.m = 10`, error surfaces. The container is **not** restarted; the previous `.conf` stays live.

What breaks	What happens
Container restart times out (30s)	`session.m = 10`. The restart was issued but did not complete in band; check `docker ps` and `docker logs hermes_unbound` manually.
`unbound-control` not available	Cache Statistics card shows "not available" message; the daemon itself is unaffected.
`drill` returns SERVFAIL for a DNSSEC test	Surfaced in the test output pane; usually means the test domain has misconfigured DNSSEC, not that Unbound is broken.
Local override shadows a public name	No error — Unbound returns the override. Use the DNS Lookup Test card to verify what consumers will actually see.

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_dns_resolver.cfm`	`hermes_commandbox`	The page
`config/hermes/var/www/html/admin/2/inc/dns_resolver_action.cfm`	`hermes_commandbox`	Save handlers (`save_forwarding`, `add_forwarder`, `add_local_record`, `restart_unbound`, `flush_cache`, etc.)
`config/hermes/var/www/html/admin/2/inc/generate_unbound_forward_conf.cfm`	`hermes_commandbox`	Renders `forward.conf` from DB and restarts the container
`config/hermes/var/www/html/admin/2/inc/generate_unbound_local_conf.cfm`	`hermes_commandbox`	Renders `local.conf` from DB and restarts the container
`config/unbound/unbound.conf`	`hermes_unbound` (read-only mount)	Baseline daemon config — cache sizes, DNSSEC, access-control
`config/unbound/conf.d/forward.conf`	`hermes_unbound` (read-write mount, regen target)	Generated forwarders
`config/unbound/conf.d/local.conf`	`hermes_unbound` (read-write mount, regen target)	Generated local overrides
`dns_forwarders`, `dns_local_records` tables	`hermes_db_server` (`hermes` DB)	Source of truth for the regen
`parameters2.unbound.forwarding.enabled`	`hermes_db_server` (`hermes` DB)	Recursive vs. forwarding mode
`${IPV4SUBNET}.117`	Docker network `hermes_net_ext`	Fixed Unbound IP that every other container's `dns:` declaration points at

Related

- LDAP RemoteAuth § DNS resolution prerequisite — the canonical case for adding a Local DNS Override (internal AD DC hostname)
- Console Settings — if the Console Address is an internal-only FQDN, this page's overrides decide whether other containers can reach it
- Server Setup — the mail-side hostname; RBL accuracy depends on the resolver's egress IP, which is the host's egress IP regardless of where Unbound is running
- Scheduled Tasks — the Ofelia jobs (RBL refresh, DMARC report fetch, fangfrisch malware-feed sync) that depend on this resolver
- Storage Topology — `hermes_unbound` is stateless; its mounts live in the Config tier (`config/unbound/`)

Revision #14
Created 2026-05-31 12:51:57 UTC by Dino Edwards
Updated 2026-06-13 12:30:01 UTC by Dino Edwards