RBL Configuration

RBL Configuration

Admin path: **Content Checks > RBL Configuration** (`view_rbl_configuration.cfm`, `inc/get_rbl_configuration.cfm`, `inc/rbl_add_entry.cfm`, `inc/rbl_edit_entry.cfm`, `inc/rbl_delete_entry.cfm`, `inc/rbl_test_entry.cfm`, `inc/generate_postfix_configuration.cfm`).

This page manages the **DNSBL** (block) and **DNSWL** (allow) lists that Postfix's `postscreen` daemon consults before a connection is even handed off to `smtpd`. Each enabled entry contributes a **weighted score** for the connecting IP; when the running total crosses the threshold set on Perimeter Checks, postscreen rejects the connection with `550 5.7.1`. Allow-list entries subtract from that score and can rescue a sender that one or two block lists flag.

The list is row-per-entry data — add, edit, delete, and live-test operations all happen on this page. The numerical threshold those weights are compared against is a single integer on the Perimeter Checks page (`postscreen_dnsbl_threshold`, default `3`).

How postscreen scoring works

```
Inbound TCP -> postscreen :25
                  |
                  v
        For each enabled DNSBL site:
          dig <reversed-client-ip>.<rbl-zone>
          if A record returned (and matches optional =127.x.x.x filter):
            add (or subtract) the entry's weight
                  |
                  v
        Sum >= postscreen_dnsbl_threshold ?
          yes -> reject 550 5.7.1
          no  -> pass to smtpd for the rest of the perimeter checks
```

The decision is made against **a single connecting IP** in a single postscreen session. Postscreen does this in parallel across every enabled zone and waits up to a few seconds for responses.

Block vs. Allow

Type	Stored weight	DNS contribution	Typical use
Block List (DNSBL)	Positive integer (`+1` … `+8` typical)	Adds to the score on hit	`zen.spamhaus.org`, `bl.spamcop.net`, `b.barracudacentral.org`
Allow List (DNSWL)	Negative integer (`-2` … `-8` typical)	Subtracts from the score on hit	`list.dnswl.org`, `wl.mailspike.net`, `hostkarma.junkemailfilter.com=127.0.0.1`

The UI presents two radio buttons (Block List / Allow List) and a positive weight; the save handler signs the weight automatically (positive for block, negative for allow) and stores both the signed integer in the `weight` column and a string representation in the `parameter` column (`<host>*<weight>` for block, `<host>*-<abs(weight)>` for allow).

Return-code filtering

Many DNSBL providers publish **different return codes** for different sub-lists inside a single zone. Spamhaus ZEN is the canonical example: `127.0.0.2` for SBL, `127.0.0.3` for the CSS sub-list, `127.0.0.4`-`7` for XBL, `127.0.0.10`-`11` for PBL. Postfix lets you match a subset of those codes with the `<hostname>=127.x.x.x` syntax (and `=127.0.0.[N..M]` / `=127.0.0.[N;M;O]` for ranges and unions). This lets an admin assign a different weight to each sub-list:

```
zen.spamhaus.org=127.0.0.2        weight 3   (SBL — moderate confidence)
zen.spamhaus.org=127.0.0.3        weight 4   (CSS)
zen.spamhaus.org=127.0.0.[4..7]   weight 6   (XBL — exploit list)
zen.spamhaus.org=127.0.0.[10;11]  weight 8   (PBL — policy list)
```

The shipped baseline includes exactly this kind of staged Spamhaus configuration plus per-code weights for several other providers; see the **RBL Entries** table after a fresh install.

The two cards on the page

1. Add RBL Entry

Four inputs: hostname (with optional `=127.x.x.x` filter), type (Block / Allow), positive weight, and submit. The hostname is validated by stripping any `=...` suffix and running the bare host through `IsValid("email", "test@" & hostPart)` — a permissive syntactic check that accepts valid DNS labels and rejects empty strings, whitespace, and obvious garbage.

Duplicates are blocked via a `LIKE '%<host>%'` lookup on the `parameters` table before insert; the page surfaces a "Duplicate Entry" warning if a row already contains the hostname (including existing entries with different `=127.x.x.x` filters — be aware that the substring check will treat `zen.spamhaus.org=127.0.0.2` and `zen.spamhaus.org=127.0.0.3` as duplicates of each other, so add sub-list variants by editing the existing row's filter rather than inserting a second).

On success: `INSERT` into `parameters` under the `postscreen_dnsbl_sites` parent, immediately call `generate_postfix_configuration.cfm`, redirect with `session.m = 1` (green "Entry Added" alert). The full RBL list takes effect on the next inbound connection.

2. RBL Entries (DataTable)

Searchable, sortable, paginated table with bulk-delete checkboxes, per-row Test / Edit / Delete buttons, and a **Test All** action.

Column	Source
Hostname	`parameter` column with the trailing `*<weight>` stripped for display
Type	Derived from sign of `weight` — positive = Block, negative = Allow
Weight	`Abs(weight)`
Status	Live AJAX result of the per-row DNS test (see below); starts as "Not Tested"
Actions	Test (vial icon), Edit, Delete

The DataTable is wrapped in a `<form>` whose submit target is the **bulk delete** handler; per-row Delete and Edit use separate hidden forms outside the DataTable so they don't collide with the bulk form.

The live RBL test

The vial-icon button on each row triggers `view_rbl_configuration.cfm?action=test_entry&id=<id>` — an AJAX-only branch that runs **before** any HTML output and returns JSON. The handler performs a **two-stage DNS probe** from inside the same container Postfix uses for its real DNSBL queries:

Stage	Query	Pass criterion

1. Test-data lookup	`dig +short A 2.0.0.127.<zone>` (the IP `127.0.0.2` reversed, prefixed onto the zone — the universal DNSBL "test record")	Response starts with `12` (i.e. a `127.x.x.x` answer) → zone is actively publishing data
2. SOA fallback	`dig +short SOA <zone>`	Non-empty response → zone infrastructure exists even if the test record was not returned

Both `dig` invocations run via `docker exec hermes_postfix_dkim dig +short +time=3 +tries=1 ...` inside a `cfthread` with a 10-second join timeout. This matters for two reasons:

1. **Same resolver as Postfix.** The CommandBox JVM's DNS resolver cannot reliably reach DNSBL zones; querying from the postfix container guarantees the test sees what the live mail flow sees.
2. **Same source IP as Postfix.** Many DNSBL providers throttle or refuse responses to public-resolver IPs (Cloudflare, Google, Quad9). The test must originate from the same egress IP as the real queries to give a meaningful result. This is the central reason Hermes ships its own DNS Resolver; if that resolver is flipped to forwarding mode through a public provider, both the live tests and real DNSBL traffic will degrade.

Result encoding:

JSON `status`	Badge	Meaning
`ok` (stage 1 hit)	Green "Zone Active" with the returned IP in the tooltip	Zone is publishing test data and reachable
`ok` (stage 2 hit)	Green "Zone Active" with "Zone active (SOA)" tooltip	Zone infrastructure exists; test record not returned (common — many providers block data-center IPs from test queries)
`error`	Red "Error"	No DNS response, NXDOMAIN, or NS delegation only with no SOA
`timeout`	Red "Unreachable"	The 10-second thread join expired

> **Green only confirms zone infrastructure — not that the list is actively publishing data.** Many DNSBL providers (Barracuda is the common example) block data-center IP ranges from running live data queries. A stage-2-only green from such a provider is the **expected** healthy result, not a problem — the live mail-flow queries are coming from the same blocked IP, so they will also miss, and the provider in that case isn't actually contributing to scoring.

Why dead RBLs are dangerous in both directions

The in-page callout flags this explicitly:

- A **dead Block List** that starts returning wildcard `127.0.0.2` matches for every IP will inflate the postscreen score for every connection — potentially blocking all inbound mail. Spamhaus's domain seizure in 2013 and the SORBS hand-off in 2024 are both examples of zones that briefly entered this state.
- A **dead Allow List** that starts wildcard-matching will subtract from every score, letting spam through that would otherwise be blocked. DNSWL has had brief outages with similar effects.

The live tests catch zones that are flat-out unreachable; they cannot catch zones that are actively publishing wrong answers. The operational mitigation is to keep the weight on any single entry small enough that one misbehaving zone cannot single-handedly cross the threshold — the shipped weights are set with this in mind (per-zone weights of `2`-`8` against a threshold of `3` means at least two corroborating hits are required for a block).

Edit and delete

The Edit modal preserves the same Block / Allow toggle + positive weight UX as Add; on save it rewrites both the `parameter` string and the signed `weight` integer. Single-row delete uses a confirm prompt + hidden `<form>` POST; bulk delete posts a comma-separated list of `parameters.id` values from the wrapping DataTable form. All three (add, edit, delete) call `generate_postfix_configuration.cfm` inline and reload Postfix in the same request.

Save flow

```
1. (Add / Edit / Delete) Validate input, INSERT / UPDATE / DELETE
   on the `parameters` table under postscreen_dnsbl_sites parent
2. cfinclude generate_postfix_configuration.cfm
     - SELECT all enabled children of every enabled parent,
       including the full ordered list of postscreen_dnsbl_sites
     - Render a temp postconf -e script + `postfix reload`
     - docker exec hermes_postfix_dkim /bin/bash <script>
     - UPDATE parameters SET applied=1 WHERE applied=2
```

```
3. session.m = 1 / 2 / 5 (Added / Deleted / Updated)
   On failure -> session.m = 4
```

The `parameters` rows for DNSBL sites

Column	Value (block-list example)	Value (allow-list example)
`parameter`	`zen.spamhaus.org=127.0.0.[4..7]*6`	`list.dnswl.org=127.0.[0..255].3*-8`
`parent_name`	`postscreen_dnsbl_sites`	`postscreen_dnsbl_sites`
`weight`	`6` (positive integer)	`-8` (negative integer)
`child`	`1` (it's a child of the directive parent row)	`1`
`order1`	Sequence within the directive (auto-incremented on Add)	Same
`enabled`	`1` to include in the live `postscreen_dnsbl_sites` value	`1`
`applied`	`1` once Postfix has been reloaded against this row, `2` while pending	Same

The generator joins the children into a single comma-separated value for the `postscreen_dnsbl_sites` directive — the live Postfix configuration ends up as one long line of `<zone>=<filter>*<weight>` tokens.

Failure semantics

Failure	Behavior
Empty hostname on Add	`session.m = 10`, redirect, no DB write
Invalid hostname syntax (Add or Edit)	`session.m = 11`, redirect, no DB write
Duplicate hostname (Add)	`session.m = 12`, redirect, no DB write
`generate_postfix_configuration.cfm` throws	`session.m = 4`, red "Configuration Error" alert
`dig` inside `hermes_postfix_dkim` times out (test only)	JSON `{"status":"timeout"}` → red "Unreachable" badge; live mail flow is unaffected
`hermes_postfix_dkim` not running (test only)	JSON `{"status":"error"}` → red "Error" badge

Files and containers touched

Path	Owner	Role
`config/hermes/var/www/html/admin/2/view_rbl_configuration.cfm`	`hermes_commandbox`	The page (with the early `action=test_entry` AJAX intercept)
`config/hermes/var/www/html/admin/2/inc/get_rbl_configuration.cfm`	`hermes_commandbox`	Loads the `postscreen_dnsbl_sites` parent ID + all active children
`config/hermes/var/www/html/admin/2/inc/rbl_add_entry.cfm`	`hermes_commandbox`	Validate, INSERT, regen + reload
`config/hermes/var/www/html/admin/2/inc/rbl_edit_entry.cfm`	`hermes_commandbox`	Validate, UPDATE, regen + reload
`config/hermes/var/www/html/admin/2/inc/rbl_delete_entry.cfm`	`hermes_commandbox`	DELETE (single or bulk), regen + reload
`config/hermes/var/www/html/admin/2/inc/rbl_test_entry.cfm`	`hermes_commandbox`	Two-stage DNS probe via `docker exec` into the postfix container
`config/hermes/var/www/html/admin/2/inc/generate_postfix_configuration.cfm`	`hermes_commandbox`	Rebuilds `main.cf` from `parameters` and reloads Postfix
`parameters` table (rows under parent `postscreen_dnsbl_sites`)	`hermes_db_server` (`hermes` DB)	Source of truth
`hermes_postfix_dkim` container	—	Runs `dig` for the live tests and `postscreen` for the real DNSBL traffic
`hermes_unbound` container	—	The recursive resolver every `dig` (test) and every `postscreen` (live) query flows through

Future work

A scheduled RBL health checker that runs the per-entry test on a timer and emails the admin when a zone goes dark — including auto-disable of consistently-failing entries — is planned (tracked on the GitHub issue tracker). Until that ships, the **Test All** button on this page is the manual equivalent; it triggers every per-row test in parallel and refreshes the Status column in place.

Related

- Perimeter Checks — postscreen knobs and the **DNSBL Threshold** the weights here are compared against
- Network Block/Allow — the `postscreen_access.cidr` table that runs **before** any DNSBL lookup; an entry there can short-circuit an IP and skip RBL scoring entirely

- Sender/Recipient Rules — per-address override applied later in the pipeline
- Anti-Spam Settings — message-content scoring that runs after a connection clears the perimeter
- DNS Resolver — `hermes_unbound` serves every DNSBL query; recursive vs. forwarding mode is the single biggest knob that affects whether DNSBL lookups succeed at all
- Relay Networks — local trusted networks where `permit_mynetworks` rescues a connection before postscreen scoring applies
- ARC Settings — content-time chain validation; unrelated to perimeter scoring but a sibling Content Checks page

---

Revision #48
Created 2026-05-31 12:52:30 UTC by Dino Edwards
Updated 2026-06-20 13:33:17 UTC by Dino Edwards