ARC Settings

ARC Settings

Admin path: **Content Checks > ARC Settings** (`view_arc_settings.cfm`)

What ARC does

ARC (Authenticated Received Chain, RFC 8617) preserves authentication results across forwarding gateways. Each gateway that handles a message can add a sealed record of the authentication state it observed, so a downstream verifier can trust the cumulative chain even when an intermediate gateway modifies the message body (adding disclaimers, banners, forwarding annotations, etc.) — body modification would otherwise invalidate the original sender's DKIM signature and lose DMARC alignment.

Hermes participates in ARC at two roles:

1. **As an originating sealer** for mail submitted by authenticated Hermes users to external recipients — Hermes is the first hop in the chain (`i=1; cv=none`).
2. **As a forwarding sealer** for inbound mail being relayed to a downstream MX (relay-mode domains) — Hermes adds a seal at `i=N+1` referencing the upstream chain.

Container and milter placement

Component	Detail
Container	`hermes_openarc` (separate service, IPv4 `.114`)
Listen	`inet:8893`
Source	flowerysong/OpenARC v1.3.0, built from release tarball
Milter chain	`master.cf` `:10026` only (post-amavis re-injection, after OpenDKIM signer at `:8892`)

Component	Detail
NOT in	`main.cf` default `smtpd_milters` — sealing at `:25` over the pre-modification body would produce an invalid seal once body_milter and CipherMail change the bytes

Modes

Mode	Effect
`s` (sign only)	Adds Hermes's seal but does not validate upstream chains
`v` (verify only)	Records inbound chain validity in `Authentication-Results` headers; does not add a seal
`sv` (sign + verify)	The gateway default; validates upstream then seals over the final body

The ARC Settings page slider auto-syncs Mode between `sv` (enabled) and `v` (disabled). The master `arc_signing_enabled` flag controls whether the daemon adds anything at all — when disabled, OpenARC operates in pass-through mode (every peer in `PeerList`, no headers added).

Single signing identity per gateway

Unlike DKIM (which uses per-sender-domain keys), ARC uses a single signing identity per gateway — Gmail seals everything with `d=google.com`, Microsoft 365 with `d=outlook.com`, and Hermes with whatever domain you generate the key for. Pick a domain you control (typically your own organization's primary domain). The selector follows the same DNS publication pattern as DKIM: `<selector>._domainkey.<domain>` with value `v=DKIM1; k=rsa; p=<public-key>`.

Hermes is the auth boundary — what `cv=fail` means and doesn't mean

Hermes is the authoritative auth / security boundary for every domain it serves. Inbound DKIM, SPF, DMARC, ARC verify, spam, virus checks all happen at Hermes. Body modifications (External Sender Banner, disclaimer, signature insertion, encryption) also happen at Hermes. Customer downstream mail servers are expected to be configured to trust Hermes implicitly: allowlist Hermes

by IP / hostname, accept forwarded mail without re-running upstream auth checks. This matches how Mimecast, Proofpoint, and Barracuda customers deploy those products — the SEG IS the trust boundary.

When Hermes modifies a message body (banner, disclaimer, etc.), any cryptographic signature whose body hash was computed over the original bytes will no longer body-validate against the current bytes. This affects:

1. The original sender's `DKIM-Signature` body hash
2. The upstream `ARC-Message-Signature` body hash for each prior `i=`

Hermes's own outbound seal at `i=N+1` is mathematically valid (it is computed over the modified body), but the `cv=` field on that seal must honestly report whether the upstream chain passed when Hermes received the message AND remains body-valid in the message it is about to send. Once Hermes modifies the body, the upstream `bh=` no longer matches the current body, so `cv=fail` is the correct (and only defensible) value.

This is by design. A correctly-configured customer downstream MX allowlists Hermes and does not re-check auth on Hermes-forwarded mail; the `cv=fail` and broken DKIM signals never gate delivery. If a customer reports forwarded mail being rejected by their downstream MX due to ARC / DKIM / DMARC failure, the fix is to allowlist Hermes on their MX, not to silence Hermes.

Removing Hermes's seal does not help: the verifier walks the chain back to `i=1` and recomputes each prior body hash against the current body independently of our seal. Stripping the entire upstream chain would require Hermes to rewrite the `From:` header (mailing-list style) to maintain DMARC alignment with a domain Hermes controls — this is a significant UX cost that all major SEG vendors (Mimecast, Proofpoint, Barracuda) have chosen not to pay.

Default Hermes behavior

Scenario	Behavior
Inbound mail with NO upstream ARC chain → any local recipient	Banner injects; Hermes seals at `i=1; cv=none`; chain is clean
Inbound mail with upstream ARC → local mailbox recipient	Banner injects; Hermes seals at `i=N+1; cv=fail`; message ends at Hermes (no downstream chain to protect — `cv=fail` is just bytes in the user's inbox)
Inbound mail with upstream ARC → relay-mode recipient	Banner injects; Hermes seals at `i=N+1; cv=fail`; downstream MX (which should be allowlisting Hermes) accepts and delivers regardless
Outbound from local Hermes user → external	Hermes is the first sealer; `i=1; cv=none`; clean chain to downstream

There is no toggle, no conditional skip, no per-domain override. Hermes always behaves the same way and reports the chain state honestly. Customer-side trust configuration is the responsibility of

the customer's MX administrator.

When a Trusted ARC Sealer configuration helps

Trusted ARC Sealer configuration on the customer side is useful in **cross-org scenarios** that aren't direct relay-to-customer-MX — for example, when a Hermes-served domain is part of a chain that forwards through other gateways, or when Hermes is forwarding to a third-party tenant the customer doesn't control. See the Trusted ARC Sealers — M365 guide for the M365 PowerShell configuration. For the standard Hermes-as-relay-MX-to-customer-mail-server case, IP allowlisting on the customer's MX is simpler and sufficient.

When to ask receivers to trust Hermes as a sealer

For customers running strict downstream verifiers (Microsoft 365 tenants that DMARC-enforce, Gmail Workspace receivers that escalate on `arc=fail`, etc.), the chain-integrity limitation can cause relay-out delivery issues even on benign inbound that happens to come through an upstream sealer. The standard industry remedy is for the receiver to add Hermes to its **Trusted ARC Sealers** list.

For Microsoft 365 customers, follow the Trusted ARC Sealers — M365 guide which covers the PowerShell command, identity requirements, and verification steps.

Key management workflow

1. Click **Add ARC Key** in the *Gateway ARC Signing Identity* card
2. Enter the signing domain (must validate as `bob@<domain>`) and selector (DNS-safe label, e.g. `arc1`)
3. Choose key size (RSA 1024 or 2048)
4. Hermes generates the key pair in `/opt/hermes/arc/keys/`
5. Copy the public key TXT record and publish at `<selector>._domainkey.<domain>` in your authoritative DNS
6. Verify DNS propagation, then click the slider to enable signing

Without an active key, Mode is forced to `v` (verify only) regardless of the saved Mode setting.

Troubleshooting

Symptom	Likely cause
Gmail "Show original" shows `arc=fail (signature failed)` on outbound from a local Hermes user	DNS for selector not published, propagated incorrectly, or wrong key
Downstream MX rejects forwarded mail from M365 sender with `arc=fail`	Expected when upstream ARC + body modification meet on relay-out; either ensure the conditional banner skip is active (`/etc/hermes/body_milter/relay_domains` is populated) or ask the receiver to configure Hermes as a Trusted ARC Sealer
OpenARC fails to start with `key data is not secure`	The signing key file ownership is not `openarc:openarc` or permissions are too loose; check the entrypoint chown step
ARC headers absent from outbound entirely	`arc_signing_enabled = 0` (master off), or no enabled key exists for the configured `arc_mode`

Related

- Email flow — full pipeline diagram including ARC placement
- DKIM Settings — outbound signing (separate from ARC)
- Trusted ARC Sealers — M365 — receiver-side trust configuration

---

Revision #8
Created 2026-05-31 12:52:21 UTC by Dino Edwards
Updated 2026-05-31 14:01:17 UTC by Dino Edwards