Introducing MTA Hooks

Proposal for a modern, standardized MTA filtering protocol

IETF 123 Madrid - Stalwart Labs LLC

Brief introduction to Milter

What is Milter?

Legacy protocol for integrating filters with MTA.

- Milter (Mail Filter) was introduced by Sendmail in the early 2000s.
- Allows external programs to inspect and modify mail during the SMTP transaction.
- Widely adopted by MTAs like Sendmail and Postfix.
- Supports message manipulation, header rewriting, message rejection, etc.

Milter in the real world

Used widely, but inconsistently — and without formal standardization.

- Still actively used today in many production environments.
- De facto standard for integrating filters like Rspamd, ClamAV, SpamAssassin, etc.
- Mostly used **over the network**, not just via loopback.
- But… it’s not a formal standard, and **not portable across MTAs.**

Problems with Milter

Obscure protocol and poor developer experience.

- Opaque & poorly documented: no formal spec, only C API.
- Binary protocol: Undocumented and difficult to implement in modern languages.
- Poor tooling: libmilter is fragile and tightly coupled to Sendmail.
- Lack of interoperability: MTAs differ in macro support and behavior
- No native state-sharing: Filters can’t pass data between each other.
- Not supported by many modern MTAs.

But... do we really need standardization?

Why a standard matters

Why formalizing a filtering protocol matters today.

- Milter is already used as a de facto network protocol:
  - Tools like Rspamd, ClamAV, and SpamAssassin integrate via milter over the network.
  - MTAs treat it as a plug-and-play interface — even across hosts and containers.
  - Yet it’s completely undocumented at the protocol level.
- Lack of a standard blocks innovation:
  - Developers must reverse-engineer libmilter to figure out protocol.
  - No clean, consistent way to build filters that work across multiple MTAs.
  - Every integration becomes a custom, fragile solution.
- A common protocol would open the door...
  - Easier development of filtering, policy, and security tools.
  - Encourages modular, language-agnostic architectures.
  - Reduces the burden on both MTA developers and filter authors.

Why not just use Webhooks or Scripts?

Existing alternatives don’t provide sufficient control.

- Webhooks are passive: They can't instruct the MTA to act (accept/reject, etc.).
- Local scripts are limiting: Don't work in distributed/cloud-native environments.
- Users want more flexibility, portability, and integration possibilities.
- Existing solutions often lead to vendor lock-in and reduced choice.

MTA Hooks

A modern and interoperable mail filtering interface

What is MTA Hooks?

A **modern, protocol-based interface** for MTA filtering, designed to be:

- Language-agnostic
- Transport-agnostic
- Simple to implement
- Extensible and flexible

Why MTA Hooks was developed

Real user pain points drove the need for a better solution.

- Direct response to user feedback.
- Available options all had serious limitations:
  - Milter was too difficult to implement (binary, undocumented, C-centric).
  - Webhooks were limited to passive observability — no way to control the SMTP session.
  - Local scripts didn’t work for users running distributed or containerized setups.
- Users wanted:
  - A simple, language-agnostic, and network-safe protocol.
  - Clear documentation and easy integration.
  - Multi-stage scanning with shared state across SMTP phases.
- MTA Hooks was created to fill this gap — and it’s already running in production.

How MTA Hooks works

Stage-based HTTP interactions between MTA and scanner.

- MTA sends **HTTP POST requests** at various SMTP stages.
- Payload is JSON (or optionally other formats like MessagePack).
- Scanner replies with an action: accept, reject, modify, etc..
- MTA acts based on the scanner’s response.

Lifecycle Example

Each SMTP phase has a hook, with shared state across stages.

- **CONNECT** → Scanner evaluates client info.
- **MAIL FROM / RCPT TO** → Scanner checks envelope.
- **DATA** → Scanner evaluates content.
- Each step has its own hook, with shared context.
- Scanner can attach state across stages.

Why standardize MTA Hooks?

Unlock innovation and interoperability across the ecosystem.

- Encourages **interoperability** between MTAs and third-party filters.
- Simplifies development: no need to reverse-engineer C libraries.
- Creates a shared, open ecosystem for filtering, observability, and policy tools.
- Gives MTA and filter developers a modern, extensible alternative to milter.
- Helps formalize what is currently being used in an ad-hoc way.
- We’re already seeing interest from scanner developers and mailbox providers.
- Some MTA hooks plugins are starting to emerge on GitHub.

Open Questions for Standardization

As we consider moving toward a standard, a few areas need community input:

- Raw vs. Parsed message content:
  - Scanners like Rspamd need access to raw message bytes (e.g., to detect malformed MIME).
  - Others (e.g., Fastmail) prefer structured MIME representations.
  - Can/should both be supported?
- Encoding message bodies in JSON:
  - Escaping large blobs of text is inefficient and messy.
  - Base64? Or define a clean way to handle body content outside the JSON structure?
- Support for binary protocol alternatives:
  - Should we allow alternatives like MessagePack, CBOR, or gRPC?
  - Could use content negotiation or capability flags
- Extensibility:
  - How should we define extension points for future capabilities?
  - Custom actions? Metadata tagging? Plugin behaviors?
- Security & Authentication:
  - How should MTAs authenticate scanners?
  - Should we recommend mTLS? Tokens? Other mechanisms?

What We’re Asking

Looking for interest, feedback, and next steps.

- Input from the community on the need for a modern protocol.
- Help shape the **scope**: filtering, observability, policy control.
- Ideally: charter a discussion around standardizing MTA Hooks.
- If there's interest, we can work toward a formal I-D and WG adoption.
- Latest specification available at https://stalw.art/docs/api/mta-hooks/overview.

EOF

Questions? Feedback?

https://stalw.art

hello@stalw.art