HIN GATEWAY – TOP-LEVEL SYSTEM OVERVIEW

Purpose, Architecture, and Responsibilities

1. What is the new HIN Mail Gateway?

The new HIN Mail Gateway is a **secure, policy-driven email processing and relay platform**.

It receives emails for a customer-owned domain, processes them according to configurable security and delivery policies (such as S/MIME signing or sealing), and reliably delivers them to destination mail servers.

Core value proposition:

- Centralized and controllable email security behavior
- Policy-driven processing instead of hardcoded logic
- Strong cryptographic identity and certificate handling
- Designed for compliance-sensitive and enterprise environments

2. High-Level Mail Flow

This section explains the **roles and responsibilities in the mail flow**, assuming the reader already knows to what the new HIN Mail Gateway is used for.

The goal is to understand **who makes which decision, where control is enforced, and why components are separated**.

The Core Idea

The new HIN Mail Gateway deliberately splits mail handling into **three distinct responsibilities**:

1. **Edge handling** – accepting and delivering mail to the outside world
2. **Decision and processing** – applying policies and cryptographic actions
3. **Re-delivery** – ensuring compliant outbound delivery

This separation makes the system predictable, auditable, and policy-driven.

Mail Flow by Responsibility

1. **Postfix Relay: External Interface**
    - Owns communication with the external mail ecosystem

- Speaks SMTP to the outside world
- Is responsible for *accepting* and *delivering* mail, but not for business decisions

2. **MXEngine: Decision and Processing Core**
    - Receives every email via an enforced internal handoff
    - Evaluates delivery and security policies
    - Applies transformations such as signing, encryption, or sealing
    - Decides how an email is allowed to leave the system

3. **Postfix Relay: Controlled Re-Entry**
    - Receives processed emails back from MXEngine
    - Delivers them to destination mail servers
    - Operates without additional content modification to avoid loops

Why the Flow Is Designed This Way

- **No bypass:** MXEngine cannot be skipped
- **Clear trust boundaries:**
    - Postfix handles transport
    - MXEngine handles intent and policy
- **Operational safety:**
    - Mail delivery retries are handled by Postfix
    - Processing failures are isolated in MXEngine
- **Policy transparency:**
    - All decisions happen in one place

Simplified Mental Model

Transport In → Decision & Processing → Transport Out

Or in concrete terms:

External SMTP → Postfix → MXEngine → Postfix → External SMTP

This model allows stakeholders to reason about behavior, failures, and responsibilities without understanding SMTP internals.

3. System Architecture Overview

3.1 Application Layer (Business Logic)

- **MXEngine**
  - Core email processing engine
  - Applies policies and executes delivery strategies
  - Handles SMTP and HTTP ingestion
- **Policy Service**
  - Stores and serves Rego (OPA) policies
  - Acts as the decision engine for MXEngine
- **S/MIME Keys Client**
  - Generates and manages cryptographic keys
  - Creates CSRs for certificate issuance
- **ID Agent**
  - Handles identity-related workflows
  - Integrates with issuer and verifier services
  - Manages message metadata and callbacks

3.2 Infrastructure Layer (Platform Services)

- **PostgreSQL**
  - Separate databases per service
  - Persistent system state
- **Vault**
  - Central secrets management
  - Stores private keys, credentials, and sensitive configuration
- **MinIO (S3-compatible storage)**
  - Stores messages, attachments, and archives
- **Postfix Relay**
  - SMTP gateway to and from the external mail ecosystem
  - DNS-driven routing (MX, SPF)

3.3 Observability Layer

- **Promtail**
  - Collects application logs from containers
  - Ships logs to centralized logging (Loki)
- **Node Exporter**
  - Provides host-level metrics

    - Exposes application version information
- **Version Collector**
    - Periodically queries application /liveness endpoints
    - Publishes running versions as metrics

4. Policy Model

The new Mail Gateway uses **OPA / Rego policies stored in the database** to control behavior.

Example: Delivery Strategy Policy

- Default strategy:
    - tunnel → smime → seal
- Exception:
    - If the email subject contains specific keywords (e.g. [public], [open])
    - The strategy changes to:
        - tunnel → smime → smtp

Key concept: Business behavior is data-driven and can be changed without redeploying services.

5. Configuration Model

5.1 Customer Configuration

Customer-specific intent is defined in a dedicated configuration file.

This includes:

- Customer identity and deployment name
- Mail domain ownership
- Cryptographic certificate identity
- Public-facing service addresses
- Optional environment-specific overrides

The customer configuration represents **who the system acts on behalf of**.

5.2 Runtime Configuration

From the customer configuration, a runtime environment is generated automatically.

This includes:

- Environment variables
- Secrets and credentials
- Service versions
- Mail relay behavior
- Logging destinations

Key principle: Customer intent is transformed into a deterministic and reproducible runtime setup.

6. Installation and Operations Lifecycle

Day-0: Installation

Performed once per deployment:

- Dependency checks and setup
- Environment generation
- Vault initialization and unsealing
- Policy loading
- S/MIME key and CSR generation
- Backup scheduling

Day-1: Operations

- Services can be started and stopped safely
- Secrets are automatically reloaded
- Health checks and metrics are continuously available
- Logs are centrally collected

Controlled Reset

A destructive reset is available for testing or redeployment:

- Explicit confirmation required
- All data and secrets are deleted

7. Security Model (High-Level)

- Centralized secrets management

- Separation of secrets per service
- Policy-controlled email behavior
- DNS-based sender validation

8. Observability and Operations

- Prometheus-compatible metrics per service
- Host-level monitoring
- Centralized structured logging
- Clear visibility into running versions and health state

9. Responsibility Overview

- Product and Policy teams define mail behavior
- Platform and Operations ensure availability
- Security owns secrets and cryptographic integrity
- Customers own DNS and mail reputation

10. Key Takeaways

- Stargate is a platform, not a simple mail relay
- Email behavior is externalized into policies
- Identity and trust are first-class concepts
- Operations, monitoring, and backup are built in
- The system is designed for controlled, enterprise-grade environments