

# Opendm Email Delivery Architecture & Implementation Guide

## 1. What the System Is

Opendm is a SaaS platform that provides:

- A public profile page ([opendm.io/username](https://opendm.io/username))
- A contact form where visitors can:
  - Choose a subject
  - Write a message
  - Enter their own email address
- A forwarding mechanism that sends the message to a **user-defined destination email**

Opendm is **not an email provider or inbox**. It is a **message relay system** that:

- Receives messages via web forms
- Forwards them as emails to users
- Preserves the visitor's identity so recipients can reply directly

Because email forwarding is the core function of the product, email delivery must be treated as **critical infrastructure**, not a convenience feature.

---

## 2. Why Email Must Be Handled Centrally

### 2.1 Reliability

- Depends on hosting server MTAs (Postfix / Sendmail)
- Shared IP reputation issues
- Silent delivery failures

### 2.2 Deliverability

- Weak or missing SPF/DKIM alignment
- High spam placement rates
- Frequent rejections by Gmail and Outlook

## 2.3 Control & Observability

- Limited header control
- No structured logging
- No bounce or complaint feedback

## 2.4 Scalability

- Each new user increases outbound email volume
- Hosting-based mail systems do not scale predictably

**Conclusion:** Email must be sent through a dedicated, authenticated email delivery service using APIs.

---

## 3. Correct Email Architecture for Opendm

### 3.1 High-Level Flow

1. Visitor submits message on [opendm.io/username](https://opendm.io/username)
  2. Opendm backend validates and stores the message
  3. Opendm sends an outbound email via an email delivery API
  4. Email is delivered to the user's chosen destination inbox
  5. User clicks **Reply** and responds directly to the visitor
- 

## 4. Critical Email Header Rules (From vs Reply-To)

### 4.1 Why Visitor Email Must NOT Be the From Address

Modern email systems enforce strict authentication rules:

#### SPF (Sender Policy Framework)

- Checks whether the sending server is authorized to send email on behalf of the **From domain**
- Opendm cannot be authorized to send email as [gmail.com](https://gmail.com), [yahoo.com](https://yahoo.com), etc.

#### DKIM (DomainKeys Identified Mail)

- Requires cryptographic signing by the **From domain**
- Opendm cannot DKIM-sign messages for third-party domains

## DMARC

- Tells receiving servers what to do if SPF/DKIM fail
- Most major providers enforce rejection or spam placement

**Result:** If Opendm sets the visitor's email as the From address, messages will fail authentication and be rejected or flagged as spam.

This behavior is enforced by Gmail, Outlook, Yahoo, and SES itself.

---

## 4.2 Correct Header Strategy

To ensure both deliverability and correct reply behavior:

- **From:** `noreply@opendm.io` (or another authenticated Opendm domain address)
- **To:** User's chosen destination email
- **Reply-To:** Visitor's email address

Email clients follow this rule:

- If `Reply-To` exists → replies go there
- If not → replies go to `From`

This guarantees:

- SPF / DKIM / DMARC pass
- No email spoofing
- Reply goes directly to the visitor

This is the **industry-standard approach** used by contact forms, support systems, and SaaS platforms.

---

## 5. Why Amazon SES Is the Best Choice

### 5.1 Amazon SES Overview

Amazon Simple Email Service (SES) is a transactional email delivery service designed for:

- High deliverability
- API-based sending
- Large-scale systems

**SES is not a marketing platform and does not add unnecessary features.**

---

## 5.2 Cost Comparison

Provider	Approx. Cost	Notes
Amazon SES	~\$0.10 / 1,000 emails	No monthly minimum
SendGrid	Higher	Costs increase quickly
Mailgun	Medium	Tier-based pricing
Postmark	High	Premium pricing

Amazon SES offers the **lowest cost per email** while maintaining strong deliverability.

---

## 5.3 SES vs SendGrid

Feature	Amazon SES	SendGrid
Cost	Very low	Medium–High
API-first	Yes	Yes
Templates / Marketing	No	Yes
Vendor lock-in	Low	Medium
Scaling	Excellent	Excellent

**Decision rationale:**

- Opendiff sends simple transactional messages
  - Branding and marketing tools are unnecessary
  - SES provides maximum control at minimal cost
-

## 6. Security & Abuse Prevention

Recommended safeguards:

- Rate limiting per profile
- CAPTCHA or bot detection
- Message size limits
- Email format validation
- Domain-based spam filtering

These controls protect both Opendm and its users.

---

## 7. Developer Implementation Guide (Amazon SES)

This section defines **what must be implemented**, not the actual code.

---

### 7.1 SES Setup

**Goal:** Enable authenticated email sending

Tasks:

- Create AWS account
- Enable Amazon SES in chosen region
- Verify sending domain
- Add SPF & DKIM DNS records
- Request production (sandbox removal)

#### AI Code Generator Prompt

"Generate a step-by-step technical guide for setting up Amazon SES, including domain verification, SPF, DKIM, and moving out of sandbox mode. Assume the reader is a backend developer."

---

## 7.2 Backend Email Send Logic

**Goal:** Send transactional email via SES API

Requirements:

- Use SES API (not SMTP)
- Dynamic recipient email
- Dynamic subject and body
- **Reply-To** set to visitor email
- Store SES message ID

**AI Code Generator Prompt**

"Generate backend code that sends a transactional email using Amazon SES API with dynamic To, Subject, Body, and Reply-To headers. Exclude frontend code."

---

## 7.3 Validation & Sanitization

**Goal:** Prevent abuse and injection

Requirements:

- Email format validation
- Content escaping
- Length limits

**AI Code Generator Prompt**

"Generate server-side validation and sanitization logic for a contact form that forwards messages using Amazon SES. Focus on security and correctness."

---

## 7.4 Logging, Monitoring & Feedback

**Goal:** Track delivery and failures

Requirements:

- Log outbound email metadata
- Capture SES message IDs
- Process bounce and complaint events

## **AI Code Generator Prompt**

"Generate a design for logging, monitoring, and handling bounce/complaint events for Amazon SES using AWS services."

---

## **8. Failure Modes & Handling Strategy**

This section documents expected failure scenarios and how the system must behave. These are normal conditions in email delivery and must be handled explicitly.

---

### **8.1 Hard Bounces (Permanent Failures)**

#### **Examples:**

- Recipient email address does not exist
- Recipient domain does not exist

#### **SES Behavior:**

- SES returns a hard bounce event

#### **System Response:**

- Mark the destination email as invalid
- Stop sending further messages to that address
- Notify the Opendifm user (optional, configurable)

Hard bounces must be treated as permanent failures.

---

### **8.2 Soft Bounces (Temporary Failures)**

#### **Examples:**

- Recipient inbox is full
- Temporary mail server issue

#### **SES Behavior:**

- SES retries delivery automatically

#### **System Response:**

- Log the event
- Allow SES retries
- No user-facing error unless repeated

Soft bounces should not immediately disable delivery.

---

### **8.3 Spam Complaints**

#### **Definition:**

- Recipient marks the email as spam

#### **SES Behavior:**

- SES emits a complaint event
- Complaint affects sending reputation

#### **System Response:**

- Log the complaint
- Flag the recipient address
- Optionally suspend forwarding for that user

Repeated complaints must trigger review or throttling.

---

### **8.4 SES API Errors**

#### **Examples:**

- Rate limit exceeded
- Temporary AWS outage
- Invalid parameters

#### **System Response:**

- Retry with exponential backoff for transient errors
  - Fail fast for validation errors
  - Alert engineering if error rate exceeds threshold
-

## **8.5 Form Abuse & Spam Submissions**

### **Examples:**

- Automated spam bots
- High-volume abuse targeting a user

### **System Response:**

- Enforce rate limits per profile and IP
  - Apply CAPTCHA or bot detection
  - Temporarily disable the affected profile if abuse persists
- 

## **8.6 Reply Handling Limitations**

### **Clarification:**

- Opendifm does not process replies
- Replies go directly from the user to the visitor

### **Implication:**

- Opendifm does not store reply content
- Delivery responsibility ends after initial send

This is an intentional design decision.

---

## **8.7 Reputation Protection**

To protect sending reputation:

- Never spoof From addresses
- Immediately suppress hard-bounced addresses
- Monitor complaint rates
- Throttle sending if abnormal patterns are detected

Failure to enforce these rules risks SES account suspension.

---

## 9. Summary

- Opendm is a message relay SaaS, not an email provider
- Visitor email must NEVER be used as the From address
- **Reply-To** preserves correct reply behavior
- Amazon SES provides the best cost-to-reliability ratio
- API-based sending ensures scalability, security, and control

This architecture follows modern email standards and is suitable for long-term scaling.