Messaging Engine Documentation

PMM Messaging Engine — Technical Documentation

Comprehensive Architecture, Pipeline, and Prompt Reference

---

1. Executive Summary

The PMM Messaging Engine converts product documentation into scored, quality-tested messaging assets through 5 specialized AI pipelines. It follows an "outside-in" philosophy — starting with real practitioner pain from developer communities rather than vendor features.

Key Capabilities

- **5 distinct generation pipelines** — Standard, Outside-In (signature), Adversarial, Multi-Perspective, and Straight Through
- **Sequential DAG architecture** — each pipeline step feeds the next
- **Shared refinement loop** — score → deslop → refine (up to 3x) with plateau detection
- **Multi-model AI strategy** — Gemini 3 Pro/Flash (production) with test profile fallback
- **5-dimension quality scoring** with scorer health tracking and degraded mode
- **8 asset types** from battlecards to narratives
- **Voice profile system** with per-voice quality thresholds

Tech Stack

Layer	Technology
Backend	Hono (REST API), TypeScript
Frontend	Vite + React + Tailwind CSS
Database	SQLite + Drizzle ORM (14 tables)
AI Models	Gemini 3 Pro/Flash (prod), Gemini 2.5 Flash (test), Claude (opt-in)

---

2. System Architecture

File Structure

```
src/
├── api/
│   ├── generate.ts             # ~186 lines — route definitions +
re-exports only
│   ├── validation.ts           # Zod schemas for all endpoints
│   ├── index.ts                # Hono app + route mounting
│   ├── middleware/
│   │   ├── auth.ts             # JWT authentication (hard error)
│   │   └── rate-limit.ts       # Per-IP HTTP rate limiting
│   ├── admin/                  # Admin CRUD (settings, voices,
documents)
│   └── workspace/              # Session management, chat, actions
├── config.ts                   # Model profiles, env config
├── types/index.ts              # Shared interfaces
(ScoringThresholds, PipelineStep, etc.)
```

```
├── services/
│   ├── pipeline/
│   │   ├── orchestrator.ts      # Job lifecycle, dispatch,
generateAndScore, refinementLoop, storeVariant
│   │   ├── prompts.ts           # All prompt builders, templates,
banned words
│   │   ├── evidence.ts          # EvidenceBundle,
community/competitive research
│   │   └── pipelines/
│   │       ├── standard.ts      # PoV-first with deep extraction
│   │       ├── outside-in.ts    # Community pain-first (signature)
│   │       ├── adversarial.ts   # Attack/defend generation
│   │       ├── multi-perspective.ts  # 3 angles + synthesis
│   │       └── straight-through.ts   # Score-only, no generation
│   ├── quality/
│   │   ├── score-content.ts     # 5 parallel scorers + health
tracking
│   │   ├── slop-detector.ts     # AI cliché detection
│   │   ├── vendor-speak.ts      # Marketing jargon scorer
│   │   ├── authenticity.ts      # Authenticity scorer
│   │   ├── specificity.ts       # Specificity scorer
│   │   ├── persona-critic.ts    # Persona fit scorer
│   │   └── grounding-validator.ts # Strips fabricated quotes
│   ├── ai/clients.ts            # Gemini + Claude API wrappers
│   ├── product/insights.ts      # Insight extraction + deep PoV
│   └── research/deep-research.ts # Gemini Deep Research agent
├── db/                          # Schema, connection, seed data
└── utils/                       # Logger, hash, retry
```

Security

Layer	Mechanism
Authentication	JWT — hard error on invalid/missing token
Validation	Zod schemas on all request bodies
Rate Limiting	Per-IP, per-path (Flash: 60/min, Pro: 15/min)
Input Limits	productDocs: 500K chars, prompt: 10K chars

---

3. Model Profile System

Controlled by `MODEL_PROFILE` env var. Every model call logs the actual model name and emits it via pipeline step events for UI visibility.

Task	Production	Test
flash	gemini-3-flash-preview	gemini-2.5-flash
pro	gemini-3-pro-preview	gemini-2.5-flash
deepResearch	deep-research-pro-preview-12-2025	gemini-2.5-flash
generation	gemini-3-pro-preview	gemini-2.5-flash
scoring	gemini-3-flash-preview	gemini-2.5-flash
deslop	gemini-3-pro-preview	gemini-2.5-flash

Test profile uses a single cheap model for all tasks — ideal for development and CI.

---

4. Quality Scoring System

5 independent scorers run in parallel via `Promise.all`:

Scorer	Scale	Direction	What It Measures
Slop	0-10	Lower is better	AI clichés, filler phrases

Vendor Speak	0-10	Lower is better	Marketing jargon, buzzwords
Authenticity	0-10	Higher is better	Genuine practitioner voice
Specificity	0-10	Higher is better	Concrete details vs vague claims
Persona	0-10	Higher is better	Fit with target persona

Scorer Health & Degraded Mode

Each scoring run tracks health: `{ succeeded: N, failed: [...], total: 5 }`.

If a scorer throws, it returns a neutral fallback score of 5 and the failure is logged. The system continues in degraded mode rather than failing the entire job. Health is visible in the pipeline step UI.

Quality Gates (Per-Voice Thresholds)

Gate	Default	Direction
slopMax	5	Reject if above
vendorSpeakMax	5	Reject if above
authenticityMin	6	Reject if below
specificityMin	6	Reject if below
personaMin	6	Reject if below

---

5. Pipelines

All pipelines follow a sequential DAG pattern. Each step feeds the next — no branching.

5.1 Standard Pipeline

PoV-first — extract a deep product thesis before generating.

```
Deep PoV Extraction (Pro)
  → Insight Extraction (Flash)
  → Community Research (Deep Research)
  → Competitive Research (Deep Research)
  → For each assetType × voice:
      → Generate with PoV-first prompt (Pro)
      → Score → Refinement Loop (up to 3x) → Store
```

5.2 Outside-In Pipeline (Signature)

Community pain-first — start with real practitioner frustrations.

```
Insight Extraction (Flash)
  → Community Deep Research (Deep Research)
  → Competitive Research (Deep Research)
  → For each assetType × voice:
      → Generate with pain-first prompt (Pro)
      → Score → Refinement Loop (up to 3x) → Store
```

5.3 Adversarial Pipeline

Attack/defend — generate objections, then craft messaging that survives them.

```
Insight Extraction (Flash)
  → Research (Deep Research)
  → For each assetType × voice:
      → Generate attack (hostile buyer objections)
      → Generate defense (messaging addressing attacks)
```

```
        → Score defense → Refinement Loop (up to 3x) → Store
```

5.4 Multi-Perspective Pipeline

3 angles + synthesis — practitioner, buyer, executive perspectives merged.

```
Insight Extraction (Flash)
  → Research (Deep Research)
  → For each assetType × voice:
      → Generate Practitioner angle
      → Generate Buyer angle
      → Generate Executive angle
      → Synthesize into unified content
      → Score → Refinement Loop (up to 3x) → Store
```

5.5 Straight Through Pipeline

Score-only — evaluate existing content without any generation.

```
Insight Extraction (Flash)
  → For each assetType × voice:
      → Score existing content (5 scorers)
      → Store (original content preserved as-is)
```

Key differences: - NO generation — content is never rewritten - NO refinement loop — no deslop, no iteration - NO research phase - Requires `existingMessaging` — fails if not provided

Use case: Benchmarking existing marketing content.

---

6. Shared Pipeline Components

Refinement Loop

1. Score with 5 scorers
2. If quality gates pass → done
3. Deslop to remove AI clichés
4. Build refinement prompt with score feedback
5. Regenerate
6. Rescore — stop if plateau detected
7. Repeat up to 3 times

Evidence Bundle

Community and competitive research packaged as `EvidenceBundle`: - `communityPain` — real developer frustrations - `competitorWeaknesses` — positioning gaps - `practitionerQuotes` — attributed quotes with source URLs - `evidenceLevel` — `strong` | `moderate` | `weak`

---

7. API Reference

Core Endpoints

Method	Path	Auth	Description
POST	`/api/generate`	JWT	Start generation job
GET	`/api/voices`	JWT	List voice profiles
POST	`/api/extract`	JWT	Extract document content
POST	`/api/auth/login`	—	Get JWT token
POST	`/api/auth/signup`	—	Create account

Generate Request Schema

```json
{
  "productDocs": "string (required, max 500K)",
          "existingMessaging": "string (optional, max 500K)",
          "prompt": "string (optional, max 10K)",
          "voiceProfileIds": ["string (min 1)"],
          "assetTypes": ["battlecard | talk_track | launch_messaging |
social_hook | one_pager | email_copy | messaging_template |
narrative"],
          "model": "string (optional)",
          "pipeline": "standard | outside-in | adversarial | multi-
perspective | straight-through"
        }
```

Asset Types

Type	Label
battlecard	Battlecard
talk_track	Talk Track
launch_messaging	Launch Messaging
social_hook	Social Hook
one_pager	One-Pager
email_copy	Email Copy
messaging_template	Messaging Template
narrative	Narrative