RoverAds Per-Tenant Webhook – Engineering Spec (HIPAA)

Owner: RoverAds microservice (Python)
Dev: Usama
Goal: Ingest lead data from external systems (e.g., Wix, Typeform, Meta Lead Ads) via per-tenant webhook endpoints; guarantee reliability, security, and HIPAA compliance.

---

1) Summary

- Provide a **per-tenant** webhook endpoint that accepts form/lead submissions from third parties.
- Treat the webhook as **ingestion only**: verify → normalize → enqueue → **ACK fast**. All heavy work happens asynchronously.
- Enforce **HIPAA** controls (transport, storage, audit, least privilege, logging hygiene).

Non-goals: UI for partners, ETL/reporting, or direct writes into PMS.

---

2) High-Level Flow

Edge → API Gateway → Ingestion Service (FastAPI or Flask on ECS/Fargate or Lambda) → Idempotency Check → **SQS (Standard) with DLQ** → Temporal workflow(s) → DB & downstream services.

- Response to provider: **202 Accepted** within ~100ms.
- Preserve **raw request body** for signature verification.

---

3) Endpoint Design (Per-Tenant)

Path pattern:

POST /v1/webhooks/{tenant_slug}/{provider}

Examples:

- `/v1/webhooks/calm-dental/wix`
- `/v1/webhooks/metro-dental/typeform`

Required headers (modes):

- `Content-Type: application/json`
- **Mode A (HMAC signed)** (preferred):
  - `X-Rover-Timestamp: <unix_ms>`
  - `X-Rover-Signature: <hex sha256 HMAC of (timestamp + "." + rawBody) using tenant+provider secret>`
  - `X-Provider-Event-Id: <string>` (if available)
- **Mode B (Bearer token)** (fallback for providers like Wix Automations that can't compute HMAC):
  - `Authorization: Bearer <tenant_webhook_token>`
  - Optional `X-Provider-Event-Id`

Query token (discouraged; last resort): `?token=<tenant_webhook_token>`

Response codes:

- `202` on accept (even if duplicate)
- `401/403` on auth/signature failure
- `415` unsupported media type

- `429` if tenant is throttled
- `500` unexpected errors (should be rare; prefer to swallow and 202 if safe)

---

4) Security & HIPAA Controls

- **Transport**: HTTPS/TLS 1.2+ (API Gateway managed certs). No plain HTTP.
- **Signature/Auth**:
    - Prefer **HMAC-SHA256** (Mode A) per tenant+provider secret in **AWS Secrets Manager**.
    - For sources lacking compute (e.g., Wix Automations), use **Bearer token** in header; rotate regularly.
    - **Timestamp skew** check ±5 minutes; reject stale requests.
    - **IP allowlisting** if provider publishes egress IPs (optional; maintainable via WAF).
- **Idempotency**: Store `(tenant, provider, event_id)` or `(tenant, provider, sha256(canonical_payload))` in **Redis** (with TTL 7 days) or Postgres unique index; drop duplicates.
- **PHI Minimization**: Only collect necessary fields (name, contact, consent, message). **Do not log PHI**. Redact sensitive values in logs.
- **Encryption at rest**: KMS CMKs for SQS, RDS, Secrets Manager, EBS.
- **Access control**: Least-privilege IAM for ingest → SQS only. Separate roles for workers.
- **Audit**: Write an **audit record** (non-PHI) per request: tenant, provider, receipt time, size, signature mode, enqueue message-id, correlation-id.
- **BAAs**: Ensure covered services are in AWS BAA scope.

---

5) Provider Compatibility Notes

- **Wix Automations (Forms → Send HTTP Request)**:
    - Use **Mode B** (Bearer) since Wix can't compute HMAC natively.
    - Configure Automation to POST JSON to our endpoint and add header: `Authorization: Bearer <tenant_webhook_token>`.
    - Include a stable identifier if Wix provides one (e.g., `submissionId`); else we compute `event_id` from a canonical hash.
- **Typeform/Jotform/HubSpot**: Prefer **Mode A** if they support signatures; else Bearer.
- **Meta Lead Ads**: Verify using their signature headers; map to Mode A equivalent and store provider receipt ID.

---

6) Normalization & Internal Envelope

Internal envelope (event):


```
{
  "event_id": "<provider GUID or computed hash>",
  "tenant_id": "calm-dental",
  "provider": "wix|typeform|meta|…",
  "event_type": "lead.submitted",
  "occurred_at": "2025-08-27T15:30:00Z",
  "source_ids": {"form_id": "abc123", "page_url": "…"},
  "payload_v": 1,
  "payload": {
    "lead": {
      "first_name": "Jane",
      "last_name": "Doe",
      "email": "jane@example.com",
      "phone": "+12025550123",
      "message": "Interested in veneers",
      "consent": {"marketing": true, "terms": true},
      "utm": {"source": "wix", "medium": "form", "campaign": "veneers"},
      "submitted_at": "2025-08-27T15:30:00Z",
      "ip": "203.0.113.10"
    }
  }
```

}

Versioning: bump `payload_v` on breaking changes; workers must handle multiple versions.

---

7) Ingestion Service – Behavior

1. Accept request, read **raw body** (required for HMAC check).
2. Validate **auth/signature** (Mode A/B). Reject on fail.
3. Parse JSON → validate against **pydantic** schemas.
4. Build **event envelope** (above). Generate `event_id` if provider lacks one: `sha256(tenant|provider|form_id|submitted_at|normalized_fields)`.
5. **Idempotency check**: if seen, short-circuit to `202`.
6. Publish to **SQS standard queue** (attributes: tenant_id, provider, event_type, event_id). Use a **per-tenant queue** or a shared queue with `MessageGroupId=tenant_id` if FIFO needed.
7. Return `202` with `X-Rover-Correlation-Id` header.

---

8) Queues & Workflows

- Default: **SQS Standard + DLQ**. Retention 4–7 days. DLQ redrive policy.
- Consumer triggers **Temporal** workflow `LeadIngestWorkflow` with input `(tenant_id, event_id)` and retrieves full event from the message body.
- Workflow steps: dedupe at DB level → enrich (geo/UTM parsing) → persist to Postgres → notify Rover Engagement/Zoe → optional fan-out to EventBridge.
- Retry policy: exponential backoff (max 7 attempts), terminal to DLQ.

---

9) Observability

- **Metrics** (per tenant & global): requests/sec, 2xx/4xx/5xx rates, signature failures, avg latency, SQS `ApproximateAgeOfOldestMessage`, DLQ size, processor success/failure.
- **Tracing**: add `correlation_id` = `uuidv4()` for each ingress; propagate through logs and SQS message attributes.
- **Logs**: JSON structured, no PHI. Redact `email/phone` in logs (store in DB only).

---

10) Error & Throttling Policy

- Unknown tenant/provider → `404` (or `403` to avoid enumeration).
- Signature/auth fail → `401/403`.
- Rate limits per tenant (API GW or WAF). On exceed → `429`.
- Validation errors: accept if minimally valid? Prefer `202` + push to **quarantine queue** for manual review.

---

11) Onboarding Checklist (Per Tenant)

1. Create `tenant_slug` and **Secrets Manager** entries:
    - `WEBHOOK_SECRET_{tenant}_{provider}` for HMAC (if used)
    - `WEBHOOK_TOKEN_{tenant}` for Bearer mode
2. Generate endpoint URL and token; store in **Rover Admin**.

3. Configure provider (e.g., Wix Automation) with URL + header `Authorization: Bearer <token>`.

4. Test send via provider sandbox; verify 202 and SQS message.

5. Set CloudWatch alarms: 5xx rate, queue age, DLQ count.

6. Document in runbook; schedule **token rotation** (e.g., 90 days).

---

12) Code Skeleton (FastAPI)

```
from fastapi import FastAPI, Request, Header, HTTPException
from starlette.responses import JSONResponse
import hmac, hashlib, os, json, time, uuid

app = FastAPI()

async def verify_hmac(raw: bytes, ts: str, sig: str, secret: str):
    if abs(int(time.time()*1000) - int(ts)) > 5*60*1000:
        raise HTTPException(status_code=401, detail="stale timestamp")
    expected = hmac.new(secret.encode(), (ts + "." + raw.decode()).encode(),
hashlib.sha256).hexdigest()
    if not hmac.compare_digest(expected, sig):
        raise HTTPException(status_code=403, detail="bad signature")

@app.post("/v1/webhooks/{tenant}/{provider}")
async def ingest(tenant: str, provider: str, request: Request,
                 x_rover_timestamp: str | None = Header(default=None),
                 x_rover_signature: str | None = Header(default=None),
                 authorization: str | None = Header(default=None),
                 x_provider_event_id: str | None = Header(default=None)):
    raw = await request.body()
    corr = str(uuid.uuid4())

    # Load secrets (cache in memory with TTL)
    mode = os.getenv("AUTH_MODE", "auto") # auto = prefer HMAC if headers present
    secret = os.getenv(f"WEBHOOK_SECRET_{tenant}_{provider}")
    token  = os.getenv(f"WEBHOOK_TOKEN_{tenant}")

    # Auth
    if x_rover_signature and x_rover_timestamp and secret:
```

```
        await verify_hmac(raw, x_rover_timestamp, x_rover_signature, secret)
    elif authorization and token and authorization == f"Bearer {token}":
        pass
    else:
        raise HTTPException(status_code=401, detail="unauthorized")

    try:
        body = json.loads(raw)
    except Exception:
        raise HTTPException(status_code=400, detail="invalid json")

    # Build envelope
    event_id = x_provider_event_id or hashlib.sha256((tenant+provider+json.dumps(body,
sort_keys=True)).encode()).hexdigest()
    envelope = {
        "event_id": event_id,
        "tenant_id": tenant,
        "provider": provider,
        "event_type": "lead.submitted",
        "occurred_at": body.get("submitted_at") or body.get("created_at"),
        "source_ids": {"form_id": body.get("form_id"), "page_url": body.get("page_url")},
        "payload_v": 1,
        "payload": {"lead": body.get("lead") or body}
    }

    # Idempotency check (pseudo)
    # if seen(event_id, tenant, provider): return JSONResponse(status_code=202, content={"ok":
True, "correlation_id": corr})

    # Enqueue to SQS (pseudo)
    # sqs.send_message(QueueUrl=..., MessageBody=json.dumps(envelope),
MessageAttributes={...})

    return JSONResponse(status_code=202, content={"ok": True, "correlation_id": corr})
```

---

13) Example cURL (Wix → RoverAds using Bearer)

```
curl -X POST "https://api.roverdent.com/v1/webhooks/calm-dental/wix" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <TENANT_WEBHOOK_TOKEN>" \
  -d '{
    "form_id": "contact-01",
```

```
  "submitted_at": "2025-08-27T15:30:00Z",
  "lead": {
    "first_name": "Jane",
    "last_name": "Doe",
    "email": "jane@example.com",
    "phone": "+12025550123",
    "message": "Interested in veneers",
    "utm": {"source":"wix","medium":"form","campaign":"veneers"}
  },
  "page_url": "https://calmdental.com/veneers"
}'
```

---

14) IaC (Terraform) – Core Resources (outline)

- `aws_apigatewayv2_api` + integrations → ECS service or Lambda
- `aws_sqs_queue` (per tenant or shared) + `aws_sqs_queue` DLQ
- `aws_secretsmanager_secret` per tenant/provider
- `aws_cloudwatch_metric_alarm` for 5xx, queue age, DLQ count
- `aws_wafv2_web_acl` (rate limiting, IP allowlist optional)

---

15) QA & Load Testing

- Unit tests for auth modes, schema validation, idempotency.
- Replay tests from DLQ.
- Load test to 1000 RPS bursts; confirm p95 < 100ms for ingress.
- Chaos test: drop SQS permissions → ensure graceful 500 alarms; autoscale restore.

---

16) Runbook

- **Rotate tokens**: update Secrets Manager; notify tenant; allow overlap window where both old & new tokens are accepted.
- **DLQ replay**: admin UI selects messages → re-enqueue.
- **Backfill**: import CSV/JSON via internal tool, push into same SQS path for uniform processing.

---

17) Acceptance Criteria

- Per-tenant endpoint operational in **staging** with Wix sample payload.
- 202 ACK within 150ms at p95 for 200 RPS.
- All messages appear in SQS with correct attributes; duplicates deduped.
- Signature/Bearer auth enforced; logs show **no PHI**.
- Alarms in place; runbook documented; secrets rotated successfully in staging.