01 — Overview Integrator Tenant EMBAN-AI

Audience: Tim IT di sisi tenant CS yang punya backend sendiri dan ingin mengintegrasikannya dengan agent EMBAN-AI. **Bahasa**: Indonesia. **Status dokumen**: aktif.

---

1. Siapa Anda di doc ini

Anda adalah developer di sisi **tenant** yang punya backend sendiri (sistem booking, CRM, ticketing internal, inventory, dsb) dan ingin agent CS EMBAN-AI berinteraksi dengan backend itu lewat API.

Contoh skenario: - Tim IT "Rent-a-Car X" yang sudah subscribe plan `cs` EMBAN-AI dan mau agent CS bisa booking mobil lewat API mereka sendiri. - Tim IT toko ritel yang mau agent CS bisa cek stok produk + menerbitkan tiket eskalasi ke help-desk internal.

Tugas teknis Anda:

- **Host ticketing webhook** yang menerima notifikasi `ticket.created` dari EMBAN saat agent CS eskalasi ke human.
- **Membalas ticket** via `POST /api/v1/tickets/:id/reply`.
- **Host action endpoint** yang menerima outbound request dari EMBAN saat customer memicu custom action (booking, cek stok, dsb).
- **Mengirim callback** via `POST /api/v1/callbacks/:submission_id` untuk action mode asinkron.

Yang Anda pegang:

- 1 secret per tenant untuk **ticket webhook** (disepakati saat onboarding tenant CS).
- 1 secret per **custom action** (disepakati saat action didaftarkan).
- Akses ke endpoint **ticket reply** dan **callback**.

Yang **tidak** Anda pegang: provisioning tenant baru, lifecycle (plan change / suspend / activate / delete), dan billing/onboarding di platform — itu jalur Upstream, di luar cakupan doc ini.

---

2. Aktor & boundary

Selain Anda, ada tiga aktor lain yang berinteraksi dengan EMBAN. Memahami siapa-bicara-ke-siapa membantu debug saat flow lintas-aktor.

Aktor	Siapa	Bicara ke EMBAN lewat
Tenant IT (Anda)	Developer di sisi tenant CS	HTTP API callback/reply + webhook receiver, HMAC per-tenant/per-action
Tenant owner	End-user yang bayar plan	WhatsApp / Telegram / Email — **bukan API**
Customer (khusus role CS)	Pelanggan akhir dari tenant CS	WhatsApp / Telegram / Email, direlay lewat tenant
EMBAN engine	Pusat orchestration	menerima request dari Anda, memanggil endpoint Anda

Catatan penting: - Tenant owner **tidak pernah** bicara ke EMBAN lewat API. Sesi tenant hidup di channel messaging. - Customer juga tidak lewat API. Customer bicara ke channel tenant; EMBAN bertindak sebagai agent CS yang membalas (atau eskalasi via ticketing ke Anda).

---

3. Base URL & versioning

- **Host canonical**: `https://api.rentalai.id`. Semua endpoint di-prefix `/api/v1/...`.
- **Mode test/sandbox**: kalau dibutuhkan untuk integrasi awal, koordinasikan via channel onboarding. Saat ini production dan test share base URL yang sama; isolasi diatur per-tenant (mis. tenant test pakai akun terpisah dengan email khusus). Subdomain dedicated belum disediakan.
- Versi API saat ini: **v1**. Breaking change memicu versi baru (`/api/v2/...`); v1 tetap hidup minimal 6 bulan setelah sunset diumumkan.

> **Jangan hardcode host** di kode. Saat onboarding, Anda akan menerima `webhook_base_url` dari Upstream — pakai value itu, simpan di env. Alasan: infrastruktur di belakang `api.rentalai.id` bisa berkembang (multi-region/routing), kode yang hardcode akan broken saat migration terjadi.

---

4. Model autentikasi (ringkasan)

Anda akan memegang **dua jenis secret** (jumlah bisa lebih banyak kalau Anda register multiple custom actions):

Secret	Skup	Dipakai di
`ticket_webhook_secret` (per-tenant)	tenant-scoped, 1 per tenant	outbound `ticket.created` webhook + inbound `/api/v1/tickets/:id/reply`
`action.secret` (per-action, per-tenant)	action-scoped, N per tenant	outbound action endpoint + inbound `/api/v1/callbacks/:submission_id`

Semua autentikasi memakai pola **HMAC-SHA256**. Ada **dua skema** yang berbeda:

1. **Ts-prefixed scheme** (default — dipakai oleh ticket webhook + callback + reply). Window 5 menit.
2. **Body-only scheme** (Action Engine outbound saja — engine kontrol retry sendiri, no replay risk).

Detail skema + contoh sign lintas bahasa ada di **#2 Authentication**.

---

5. Standard response envelope

Semua response JSON mengikuti skema berikut.

Sukses:

```
{
  "ok": true,
  "tenant_id": "01J...",
  "...": "field-domain-specific"
}
```

Error:

```
{
  "ok": false,
  "error_code": "schema_invalid",
  "message": "Human-readable reason (EN)",
  "request_id": "req_01J...",
  "details": { "...": "opsional, berisi breakdown validasi" }
}
```

`error_code` adalah string stabil yang aman di-switch di kode Anda. `message` boleh berubah. `request_id` selalu ada — sertakan di ticket support kalau ada masalah.

Daftar lengkap `error_code` di **#8 Error codes**.

---

6. Rate limit & idempotency (ringkasan)

- **Ticket reply + callback** memakai kuota standar (default 60 req/menit, `API_RATE_LIMIT_STANDARD`).

- Semua endpoint *write* menerima header `Idempotency-Key` (UUID/ULID pilihan caller). Permintaan ulang dengan key yang sama dalam 24 jam mengembalikan response cached, tidak membuat resource duplikat.
- Retry dari sisi caller: **eksponensial** (2s, 4s, 8s, 16s) maksimal 4 kali, lalu gagal permanen.

Detail di **#8 Error codes** (§4 Rate limits + §5 Idempotency).

---

7. Hello world — cek health

Endpoint `GET /api/v1/health` adalah **public**, tanpa HMAC. Pakai untuk smoke-test koneksi sebelum Anda mulai sign request.

curl

```
curl -sS https://api.rentalai.id/api/v1/health
```

Node.js (fetch)

```
const res = await fetch("https://api.rentalai.id/api/v1/health");
const body = await res.json();
console.log(body); // { ok: true, status: "healthy", version: "1.0.0", ... }
```

Python (requests)

```
import requests

resp = requests.get("https://api.rentalai.id/api/v1/health", timeout=5)
resp.raise_for_status()
print(resp.json())  # {'ok': True, 'status': 'healthy', ...}
```

Kalau 3 bahasa di atas jalan, lanjut ke #2 untuk skema HMAC.

---

8. Urutan membaca dokumen

Saran urutan: **#1 → #2 → #5 → #12 (untuk read-only integrasi) → #3 → #6 → #4 → #7 (skim) → #8 → #9 → #11**.

Daftar lengkap:

#	File	Ringkasan
1	`01-overview.md`	File ini.
2	`02-authentication.md`	Skema HMAC (ts-prefixed + body-only) + contoh sign lintas bahasa.
3	`03-callbacks-api.md`	Callback async action (Anda → EMBAN).
4	`04-ticketing-api.md`	`ticket.created` webhook (EMBAN → Anda) + endpoint reply (Anda → EMBAN).
5	`05-action-engine.md`	Outbound request EMBAN → endpoint action Anda.
6	`06-file-delivery.md`	Tenant → customer file delivery via `files[]` di callback.
7	`07-working-hours.md`	Awareness — apa yang terjadi saat customer masuk di luar jam kerja.
8	`08-error-codes.md`	Konsolidasi `error_code` + rate limit + idempotency detail.
9	`09-examples.md`	Flow end-to-end yang merangkai endpoint-endpoint.
10	`10-glossary.md`	Istilah: tenant, agent, role, plan, channel, customer, ticket, action, submission, dsb.
11	`11-reference-implementation.md`	Cross-link ke kode receiver nyata di repo `emban-ai` (confidential).
12	`12-template-library.md`	**Template library** — 3 action template generic (`business_metric` , `business_list` , `business_search`) untuk read-only query data backend. Satu endpoint per template, internal dispatch berdasarkan field. Cocok kalau owner butuh expose metric/list/search dari ERP/ CRM/POS tenant tanpa custom-define action per-query.

9. Kontak & dukungan

- Bug report & pertanyaan teknis: channel komunikasi yang disepakati pada onboarding (biasanya Slack shared channel atau email ke tim EMBAN).
- Security issue (suspected leak HMAC secret, anomaly): segera rotate via prosedur di **#2 §10**, lalu escalate.
- SLA, jadwal maintenance, dan runbook insiden: di luar cakupan doc publik — kontrak terpisah.

02 — Authentication (HMAC)

Audience: Tenant IT. **Prasyarat**: `01-overview.md`.

Semua endpoint write yang Anda akan call (`POST /api/v1/tickets/:id/reply` , `POST /api/v1/callbacks/:submission_id`) DAN semua endpoint Anda yang EMBAN call (ticket webhook, action endpoint) diautentikasi pakai **HMAC-SHA256 signature**. Tidak ada OAuth, tidak ada bearer token. Pola ini mengikuti konvensi Stripe/Slack — sederhana, stateless, dan tahan terhadap replay selama jam NTP-sinkron.

---

1. Dua jenis secret

Anda akan memegang dua kategori secret. Masing-masing hanya dipakai untuk satu golongan endpoint.

Nama logis	Scope	Dari mana didapat	Dipakai di
Ticket webhook secret	per-tenant	disepakati saat tenant CS onboard; EMBAN menyimpan nama env var di `tenants.ticket_webhook_secret_env`	outbound `ticket.created` webhook + inbound `POST /api/v1/tickets/:id/reply`
Action callback secret	per-action, per-tenant	dikonfigurasi saat Anda mendaftarkan custom action (`endpoint.auth.secret_env`)	outbound action endpoint + inbound `POST /api/v1/callbacks/:submission_id`

> Catatan implementasi: EMBAN menyimpan hanya **nama env var** di DB — secret aslinya hidup di `process.env` di runtime EMBAN. Anda cukup tahu nilai secret-nya (yang dipakai untuk sign), tidak perlu tahu nama env var di sisi EMBAN.

---

2. Dua skema tanda tangan

Berbeda dengan banyak API yang pakai satu skema HMAC saja, EMBAN pakai **dua skema** tergantung kategori endpoint. Penting: pilih skema yang benar untuk endpoint yang Anda implement, kalau salah signature 100% tidak match.

2.1 Ts-prefixed scheme (default)

Dipakai oleh: ticket webhook (#4), ticket reply, callback async action (#3), file delivery (#6).

Canonical signing input:

```
<timestamp_ms> "." <raw_request_body>
```

- `timestamp_ms` — current time sebagai **Unix millis** (string angka desimal, bukan float, bukan detik).
- `raw_request_body` — **string body persis seperti yang dikirim di wire**. Kalau Content-Type `application/json`, artinya exact bytes dari `JSON.stringify(...)` yang akan dikirim. **Jangan whitespace-tweak, jangan re-stringify.**
- Untuk request tanpa body, `raw_request_body` = string kosong (`""`).

Signature:

```
signature = "sha256=" + hex( HMAC_SHA256(secret, signing_input) )
```

`hex(...)` = lowercase hex, **tanpa pemisah**, tanpa uppercase. Prefix literal `sha256=` wajib.

2.2 Body-only scheme (Action Engine outbound)

Dipakai oleh: outbound action endpoint (#5 §3) — saja.

Canonical signing input:

```
<raw_request_body>
```

(Tidak ada timestamp prefix.)

Signature dihitung dengan rumus yang sama:

```
signature = "sha256=" + hex( HMAC_SHA256(secret, raw_request_body) )
```

> **Kenapa skip timestamp?** Engine kontrol retry sendiri dengan `Idempotency-Key` (= `submission_id`). Tidak ada risk replay dari pihak ketiga karena tenant set secret-nya di env, EMBAN yang outbound. Untuk inbound (callback Anda balik ke EMBAN), tetap pakai ts-prefixed (§2.1).

---

3. Header wajib

Header yang Anda kirim saat outbound (callback / reply ke EMBAN):

Header	Nilai	Contoh
`X-EMBAN-Timestamp`	timestamp dalam Unix millis (string) — **ts-prefixed scheme saja**	`1745145600123`
`X-EMBAN-Signature`	`sha256=<hex-digest>`	`sha256=9f3a1c...`
`Content-Type`	`application/json`	wajib untuk POST berbody
`Idempotency-Key`	opsional tapi **sangat direkomendasikan** untuk semua POST	ULID / UUID / key unik lain, < 128 char

Header yang EMBAN kirim saat outbound (ticket webhook / action endpoint Anda):

Header	Nilai	Catatan
`X-EMBAN-Signature`	`sha256=<hex-digest>`	selalu ada
`X-EMBAN-Timestamp`	Unix millis	hanya pada ts-prefixed scheme (ticket webhook)
`X-EMBAN-Tenant-Id`	tenant ULID terkait	identifikasi tenant
`X-EMBAN-Submission-Id`	ULID	hanya untuk Action Engine outbound
`X-EMBAN-Schema-Version`	mis. `"1.0"`	hanya untuk Action Engine, sesuai `schema_version` action def
`Idempotency-Key`	ULID (= `ticket.id` / `submission_id`)	pakai untuk dedup di sisi Anda

---

4. Toleransi skew & replay

Berlaku hanya untuk ts-prefixed scheme:

- Default window: **5 menit** (`API_HMAC_TIMESTAMP_SKEW_MS=300000`).
- Jika `|now - X-EMBAN-Timestamp| > window` → 401 `timestamp_skew` .
- Implikasi: **jam server Anda wajib tersinkron NTP**. Drift > 5 menit = semua request ditolak.
- Tidak ada nonce/anti-replay kedua. Window timestamp sudah cukup kecil untuk mencegah replay; idempotency-key mencegah efek samping duplikat di layer aplikasi.

Body-only scheme tidak melakukan skew check — engine kontrol retry sendiri.

---

5. Kode contoh — Node.js

5.1 Sign outbound (callback / reply ke EMBAN — ts-prefixed)

```
import { createHmac, randomUUID } from "node:crypto";

function signRequest(body, secret) {
  const ts = Date.now();
  const raw = typeof body === "string" ? body : JSON.stringify(body);
  const sig =
    "sha256=" +
    createHmac("sha256", secret).update(`${ts}.${raw}`).digest("hex");
  return { ts, raw, sig };
}

async function replyToTicket(baseUrl, ticketId, secret, payload) {
  const { ts, raw, sig } = signRequest(payload, secret);
  const res = await fetch(`${baseUrl}/api/v1/tickets/${ticketId}/reply`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-EMBAN-Timestamp": String(ts),
      "X-EMBAN-Signature": sig,
      "Idempotency-Key": randomUUID(),
    },
    body: raw,                    // kirim string yang SAMA dengan yang di-sign
  });
  return res.json();
}
```

Pitfall: jangan lakukan `body: payload` (object) — framework fetch/axios akan re-stringify-kan secara internal dengan urutan key yang berbeda, membuat signature tidak cocok. Sign dulu, kirim string yang sama persis.

5.2 Verify inbound (ticket webhook / action endpoint dari EMBAN — ts-prefixed)

```
import express from "express";
import { createHmac, timingSafeEqual } from "node:crypto";

const SECRET = process.env.EMBAN_TICKET_WEBHOOK_SECRET;
const SKEW_MS = 5 * 60 * 1000;

app.use("/emban/tickets", express.raw({ type: "application/json" }));

app.post("/emban/tickets", (req, res) => {
  const ts = Number(req.header("X-EMBAN-Timestamp"));
  const sig = req.header("X-EMBAN-Signature") ?? "";
  if (!Number.isFinite(ts) || Math.abs(Date.now() - ts) > SKEW_MS) {
    return res.status(401).send("timestamp skew");
  }
  const raw = req.body.toString("utf8");
  const expected =
    "sha256=" +
    createHmac("sha256", SECRET).update(`${ts}.${raw}`).digest("hex");
  const a = Buffer.from(sig);
  const b = Buffer.from(expected);
  if (a.length !== b.length || !timingSafeEqual(a, b)) {
    return res.status(401).send("bad signature");
  }
  // ... process payload
});
```

5.3 Verify inbound body-only (action endpoint outbound dari EMBAN)

Lebih sederhana — tidak ada timestamp / skew check.

```
app.post("/emban/action", (req, res) => {
  const sig = req.header("X-EMBAN-Signature") ?? "";
  const raw = req.body.toString("utf8");
  const expected =
    "sha256=" +
    createHmac("sha256", SECRET).update(raw).digest("hex");
  const a = Buffer.from(sig);
  const b = Buffer.from(expected);
  if (a.length !== b.length || !timingSafeEqual(a, b)) {
    return res.status(401).send("bad signature");
  }
  // ... process action payload
});
```

6. Kode contoh — Python

6.1 Sign outbound (ts-prefixed)

```
import hmac, hashlib, json, time, uuid, requests

def sign_request(body, secret: str):
    ts = int(time.time() * 1000)
    raw = body if isinstance(body, str) else json.dumps(body, separators=(",", ":"))
    sig = "sha256=" + hmac.new(
        secret.encode(),
        f"{ts}.{raw}".encode(),
        hashlib.sha256,
    ).hexdigest()
    return ts, raw, sig

def reply_to_ticket(base_url: str, ticket_id: str, secret: str, payload: dict):
    ts, raw, sig = sign_request(payload, secret)
    resp = requests.post(
        f"{base_url}/api/v1/tickets/{ticket_id}/reply",
        data=raw,                # kirim string, bukan json=
        headers={
            "Content-Type": "application/json",
            "X-EMBAN-Timestamp": str(ts),
            "X-EMBAN-Signature": sig,
            "Idempotency-Key": str(uuid.uuid4()),
        },
        timeout=10,
    )
    return resp.json()
```

Pitfall: `requests.post(url, json=payload)` me-reserialize dengan whitespace default `", "` — signature tidak akan cocok. Pakai `data=raw` dengan string yang SAMA dengan yang di-sign.

6.2 Verify inbound (Flask — ts-prefixed)

```
import hmac, hashlib, time, json, os
from flask import Flask, request, abort

SECRET = os.environ["EMBAN_TICKET_WEBHOOK_SECRET"].encode()
SKEW_MS = 5 * 60 * 1000

app = Flask(__name__)

@app.post("/emban/tickets")
def handle_ticket():
    ts_hdr = request.headers.get("X-EMBAN-Timestamp", "")
    sig_hdr = request.headers.get("X-EMBAN-Signature", "")
    try:
        ts = int(ts_hdr)
    except ValueError:
        abort(401)
    if abs(int(time.time() * 1000) - ts) > SKEW_MS:
        abort(401)

    raw = request.get_data()                                    # bytes, bukan parsed JSON
    expected = "sha256=" + hmac.new(
        SECRET, f"{ts}.".encode() + raw, hashlib.sha256
    ).hexdigest()
    if not hmac.compare_digest(sig_hdr, expected):
        abort(401)

    payload = json.loads(raw)
    # ... process
    return "ok", 200
```

6.3 Verify inbound body-only (action endpoint)

```
@app.post("/emban/action")
def handle_action():
    sig_hdr = request.headers.get("X-EMBAN-Signature", "")
    raw = request.get_data()
    expected = "sha256=" + hmac.new(
        ACTION_SECRET, raw, hashlib.sha256
    ).hexdigest()
    if not hmac.compare_digest(sig_hdr, expected):
        abort(401)
    # ... process
    return "ok", 200
```

---

7. Kode contoh — curl + openssl (reproduksi manual)

Untuk debugging — generate signature secara manual lalu compare dengan output kode Anda.

7.1 Ts-prefixed (callback ke EMBAN)

```
#!/usr/bin/env bash
SECRET="your-action-callback-secret"
BASE="https://api.rentalai.id"
SUBMISSION_ID="01J1ZXK7ABCDE..."
BODY_FILE="/tmp/body.json"

cat > "$BODY_FILE" <<'JSON'
{"status":"success","template_key":"booking_confirmed","data":{"booking_id":"BK-2026-0412"}}
JSON

TS=$(date +%s%3N)
BODY=$(cat "$BODY_FILE")
SIG="sha256=$(printf '%s.%s' "$TS" "$BODY" | openssl dgst -sha256 -hmac "$SECRET" -hex | awk '{print $2

curl -sS -X POST "$BASE/api/v1/callbacks/$SUBMISSION_ID" \
  -H "Content-Type: application/json" \
  -H "X-EMBAN-Timestamp: $TS" \
  -H "X-EMBAN-Signature: $SIG" \
  -H "Idempotency-Key: $SUBMISSION_ID" \
  --data-binary "@$BODY_FILE"
```

Pitfall: `-d` vs `--data-binary`. `-d` akan strip newline/whitespace; gunakan `--data-binary` supaya body di-kirim byte-for-byte sama dengan yang di-hash.

7.2 Body-only (verify inbound dari EMBAN — manual reproduksi)

Skip TS, just hash body:

```
SIG_EXPECTED="sha256=$(cat "$BODY_FILE" | openssl dgst -sha256 -hmac "$SECRET" -hex | awk '{print $2}')
echo "Expected: $SIG_EXPECTED"
echo "Got:      $(grep X-EMBAN-Signature received-headers.txt)"
```

---

8. Daftar error autentikasi

HTTP	`error_code`	Artinya	Aksi
401	`missing_auth_headers`	salah satu dari `X-EMBAN-Signature` / `X-EMBAN-Timestamp` absen (timestamp untuk ts-prefixed; signature selalu)	pastikan header sesuai skema endpoint
401	`invalid_timestamp`	`X-EMBAN-Timestamp` bukan angka	kirim Unix millis sebagai string
401	`timestamp_skew`	drift > 5 menit	sync NTP; cek response body untuk nilai drift
401	`unknown_signer`	EMBAN tidak tahu secret untuk request ini	biasanya tenant_id salah, atau tenant belum set `ticket_webhook_secret_env` di sisi mereka
401	`invalid_signature`	signature mismatch	cek: (a) body bytes identik dengan yang di-sign, (b) secret benar, (c) hex lowercase, (d) prefix `sha256=` ada, (e) skema benar (ts-prefixed vs body-only)

9. Tips debugging signature mismatch

1. **Print-then-send**: log `raw_request_body` yang di-sign sebelum dikirim, log lagi setelah dikirim. Harus identik.
2. **Encoding**: raw body harus UTF-8. Non-ASCII (aksara Arab, emoji) aman selama encoding konsisten di kedua sisi.
3. **Headers dipakai apa?** EMBAN hanya men-sign `timestamp.body` (atau `body` saja untuk body-only) — **tidak** mensign header lain. Jadi perbedaan header tidak mempengaruhi signature.
4. **Window**: jika `timestamp_skew`, response menyertakan nilai drift ms; pakai itu untuk mengukur seberapa jauh clock Anda mundur/maju.
5. **Salah skema**: paling sering. Action Engine outbound = body-only. Semua yang lain = ts-prefixed. Cek ulang di doc endpoint mana.
6. **Test vektor**: ulang dengan body `{"ping":1}` + timestamp tetap + secret tetap, bandingkan digest dengan implementasi lain (mis. CyberChef / openssl) untuk memastikan library HMAC Anda benar.

Contoh test vektor (ts-prefixed):

```
secret    = "test-secret"
timestamp = 1745145600123
body      = {"ping":1}
input     = 1745145600123.{"ping":1}
sha256    = 6a2f8e0b0f0e3b2f7e0b7a4e8e4f0c9e2d1b8a5f3c6e9d2a4b7e0c1d3f5a8b9c   (contoh)
header    = X-EMBAN-Signature: sha256=6a2f8e0b...8b9c
```

> Angka di atas **bukan** nilai nyata — regenerate dengan HMAC sendiri untuk verifikasi.

---

10. Rotasi secret

Ticket / action secret (per-tenant): - Koordinasi per tenant. Secret diubah di file env EMBAN + sisi Anda dalam satu operasi. - Pakai nama env var baru (`TICKET_WEBHOOK_SECRET_TENANT_X_V2`) dan update `tenants.ticket_webhook_secret_env` via migration/admin tool — ini menjamin tidak ada "ambiguous" secret.

Indikator leak: - 401 `invalid_signature` burst dari IP yang tidak dikenal → audit log, anggap secret compromised, rotate segera. - Secret jangan sekali-kali di-commit ke git. Kalau terlanjur, rotate dulu baru rewrite history.

Cadence yang biasa dipakai: 90 hari untuk webhook secrets, lebih sering kalau Anda punya regulatory requirement.

---

11. Ringkasan checklist

- [ ] Punya 1 ticket secret per tenant CS (kalau menerima forwarded ticket).
- [ ] Punya 1 action secret per custom action (kalau host action endpoint).
- [ ] NTP-sync di semua host yang sign/verify (untuk ts-prefixed scheme).
- [ ] Body byte-for-byte identik dengan yang di-sign (di kedua arah: verify inbound + sign outbound).
- [ ] Header `X-EMBAN-Signature` pakai prefix `sha256=` + lowercase hex.
- [ ] Pilih skema yang benar per endpoint (ts-prefixed vs body-only) — lihat doc endpoint masing-masing.
- [ ] Library HMAC-SHA256 standar (crypto/hashlib/openssl) — jangan re-implement.

Lanjut: **#5 Action engine** (lalu #3 Callbacks / #6 File delivery / #4 Ticketing).

03 — Callbacks API (async action)

Audience: Tenant IT. **Prasyarat**: `01-overview.md`, `02-authentication.md`, dan paham konsep **custom action** (didefinisikan oleh tenant CS di konfigurasi EMBAN — lihat **#5 Action engine**).

Endpoint ini dipanggil Tenant IT **setelah** EMBAN memicu action asinkron ke backend Tenant IT. Alurnya searah: customer → EMBAN → Tenant IT → EMBAN → customer.

---

1. Model & terminologi singkat

Istilah	Artinya
Action	Spesifikasi kustom milik tenant yang mendefinisikan endpoint Tenant IT + field-field yang harus dikumpulkan dari customer + template jawaban. Disimpan sebagai row di `tenant_actions`.
Action definition	JSON schema action (lihat `src/services/action-schema.ts`), mencakup `endpoint.url`, `endpoint.auth.secret_env`, dan `response.mode` (`sync` atau `async`).
Submission	Satu instansi eksekusi action. Row di `action_submissions` dengan `id` ULID. Status lifecycle: `collecting` → `submitted` → `awaiting_callback` → `callback_delivered`.
Sync mode	EMBAN menunggu response HTTP Tenant IT sampai `timeout_ms`, render langsung ke customer. **Tidak ada callback.**
Async mode	EMBAN balas `immediate_ack` ke customer, lalu menunggu Tenant IT POST ke callback endpoint untuk melanjutkan. **Mode inilah yang membutuhkan endpoint ini.**

Jika action Anda pakai `response.mode: "sync"`, dokumen ini tidak relevan — cukup reply body HTTP di tempat (lihat **#5 Action engine**).

---

2. Endpoint

```
POST https://api.rentalai.id/api/v1/callbacks/:submission_id
Content-Type: application/json
X-EMBAN-Timestamp: <unix-millis>
X-EMBAN-Signature: sha256=<hex>
Idempotency-Key: <ulid | uuid>            # opsional tapi direkomendasikan
```

- `submission_id` diberikan ke Tenant IT di **outbound request pertama** (`POST <action.endpoint.url>`) dalam body JSON + header `X-EMBAN-Submission-Id` (lihat **#5**). Simpan itu, pakai di URL callback.

- **Secret untuk HMAC** = value dari env var yang namanya disebut di `action.endpoint.auth.secret_env`. Disepakati saat tenant mendaftarkan action.
- **Window status**: EMBAN hanya menerima callback saat submission masih berstatus `awaiting_callback`. Status lain → 409 `invalid_submission_state`.

---

3. Request body

Skema Zod ada di `src/api/schemas/callback.ts`. Ringkasan field:

Field	Tipe	Wajib	Catatan
`status`	`"success"` | `"error"`	default `"success"`	Menentukan template mana yang dipilih otomatis.
`template_key`	string	optional	Override eksplisit — nama key di `response.error_templates`. Bekerja untuk success maupun error outcomes.
`data`	object	default `{}`	Payload yang diinterpolasi ke template (mustache-lite). Key bebas, nested paths OK.
`error_code`	string	optional, direkomendasikan saat `status="error"`	Dicari di `response.error_templates[<code>]`. Jatuh ke `.default` jika tidak ada.
`error_message`	string	optional	Tersedia sebagai `{{error_message}}` di template.
`files`	array ≤ 10	optional	Attachment untuk customer. Detail di §6.

Contoh minimal (success):

```
{
  "status": "success",
  "template_key": "booking_confirmed",
  "data": {
    "booking_id": "BK-2026-0412",
    "car_model": "Toyota Avanza",
    "pickup_date": "22 April 2026",
    "price_idr": "850.000"
  }
}
```

Contoh error:

```
{
  "status": "error",
  "error_code": "out_of_stock",
  "error_message": "Unit sudah disewa tanggal tersebut",
  "data": { "alternative_date": "24 April 2026" }
}
```

---

4. Pemilihan template (fallback chain)

EMBAN memilih template dalam urutan berikut — yang pertama cocok dipakai:

1. `template_key` eksplisit → `response.error_templates[template_key]`
2. Jika `status="success"` → `response.success_template`, lalu `error_templates.default`
3. Jika `status="error"` → `error_templates[error_code]`, lalu `error_templates.default`
4. **Fallback terakhir** (no template matched): EMBAN menyusun pesan generik dari `data` / `error_message`. Tidak silent-drop.

Rekomendasi: tenant **selalu** menyediakan `error_templates.default` supaya fallback generik jarang dipakai.

---

5. Template rendering (mustache-lite)

EMBAN memakai parser mustache-lite sederhana (lihat `renderMustache` di `src/api/routes/callbacks.ts`):

- `{{field}}` → `data.field`
- `{{obj.nested.key}}` → `data.obj.nested.key`
- `{{error_code}}`, `{{error_message}}` tersedia top-level
- Value `null` / `undefined` → empty string (tidak error, tidak "undefined" literal)
- Non-string value → `JSON.stringify(value)`
- **Tidak ada** loop `{{#each}}`, conditional `{{#if}}`, partial `{{>}}`, atau filter Handlebars — sengaja sederhana. Kalau perlu logika, lakukan di sisi Tenant IT sebelum POST.

Contoh action definition yang cocok dengan request di §3:

```
{
  "response": {
    "mode": "async",
    "immediate_ack": {
      "id": "Booking sedang diproses, mohon tunggu..."
    },
    "error_templates": {
      "booking_confirmed": {
        "id": "Booking {{booking_id}} untuk {{car_model}} pada {{pickup_date}} sudah dikonfirmasi. Tota
      },
      "out_of_stock": {
        "id": "Maaf, {{error_message}}. Tanggal terdekat tersedia: {{alternative_date}}."
      },
      "default": {
        "id": "Sistem: {{error_message}}"
      }
    }
  }
}
```

6. File attachment (Phase 5.5)

Field `files[]` memungkinkan Tenant IT melampirkan file (invoice PDF, foto unit, dll) yang akan dikirim ke customer di channel aslinya.

Schema per-file:

```
{
  "url": "https://files.tenant.example/invoice/BK-2026-0412.pdf",
  "filename": "invoice.pdf",
  "mime_type": "application/pdf",
  "expected_size_bytes": 524288
}
```

Aturan sisi EMBAN:

Aturan	Nilai default	Sumber config
Protokol	**HTTPS wajib**	hardcoded
Hostname	harus cocok dengan **allowlist per-tenant** (`tenants.file_download_allowlist`)	dikonfigurasi tenant saat onboarding
Max size per file	10 MB	`FILE_DELIVERY_MAX_BYTES`
Max files per callback	10	schema
MIME allowlist	PDF, PNG, JPG, MP3, MP4, MP3, docx, xlsx, zip (lihat `file-download.ts`)	hardcoded
Content sniff	magic bytes dicocokkan dengan `mime_type` — mismatch → rejected	
TTL di disk EMBAN	24 jam (lalu auto-delete)	`FILE_DELIVERY_TTL_HOURS`
Connect timeout	10 detik	`FILE_DELIVERY_CONNECT_TIMEOUT_MS`
Read timeout	30 detik	`FILE_DELIVERY_READ_TIMEOUT_MS`

Partial success: text message (dari §5) selalu di-deliver dulu. Setiap file diproses independen — beberapa sukses, beberapa gagal, reply body membawa detail per file:

```
{
  "ok": true,
  "delivered": true,
  "customer_channel": "whatsapp",
  "submission_id": "01J...",
  "files": {
    "delivered": 1,
    "rejected": 1,
    "failed": 0,
    "details": [
      { "url": "...", "status": "delivered" },
      { "url": "...", "status": "rejected", "reason": "mime_not_allowed" }
    ]
  }
}
```

Status nilai: `delivered` , `rejected` (validation sisi EMBAN — hostname/MIME/size), `fetch_failed` (network/4xx/5xx), `delivery_failed` (channel adapter error).

7. Response EMBAN

200 OK — delivered

```
{
  "ok": true,
  "delivered": true,
  "customer_channel": "whatsapp",
  "submission_id": "01J1ZXK7..."
}
```

200 OK — customer deleted / blocked

```
{
  "ok": true,
  "delivered": false,
  "reason": "customer_deleted"
}
```

Submission tetap ditandai `callback_delivered` (dengan `error_message=customer_deleted` di DB). Tenant IT tidak perlu retry — customer memang sudah hilang dari sistem.

4xx/5xx — daftar error

HTTP	`error_code`	Artinya	Aksi Tenant IT
400	`schema_invalid`	body tidak lulus Zod validation	cek `details`, perbaiki payload
401	`missing_auth_headers` / `invalid_signature` / `timestamp_skew` / `unknown_signer`	HMAC gagal	lihat `02-authentication.md`
404	`submission_not_found`	`submission_id` tidak ada	cek typo / expired
409	`invalid_submission_state`	submission bukan `awaiting_callback` (mis. sudah delivered, atau timeout)	**jangan retry** — state tidak akan balik
409	`idempotency_key_conflict`	key yang sama dipakai untuk endpoint berbeda	ganti key
500	`action_missing` / `action_definition_corrupt` / `no_customer_linked` / `no_agent_for_tenant`	state rusak di sisi EMBAN	escalate ke tim EMBAN dengan `request_id`
503	`delivery_unavailable` / `delivery_failed`	channel adapter belum siap atau error saat kirim	retry dengan backoff eksponensial; kalau konsisten → escalate

8. Contoh lengkap — Node.js

```
import { createHmac, randomUUID } from "node:crypto";

const BASE = "https://api.rentalai.id";
const SECRET = process.env.EMBAN_ACTION_SECRET;        // dari env yang SAMA dengan action.endpoint.aut

async function sendCallback(submissionId, body) {
  const raw = JSON.stringify(body);
  const ts = Date.now();
  const sig =
    "sha256=" +
    createHmac("sha256", SECRET).update(`${ts}.${raw}`).digest("hex");

  const res = await fetch(`${BASE}/api/v1/callbacks/${submissionId}`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-EMBAN-Timestamp": String(ts),
      "X-EMBAN-Signature": sig,
      "Idempotency-Key": randomUUID(),
    },
    body: raw,
  });
  return { status: res.status, body: await res.json() };
}

// contoh pemakaian
await sendCallback("01J1ZXK7ABCDE", {
  status: "success",
  template_key: "booking_confirmed",
  data: {
    booking_id: "BK-2026-0412",
    car_model: "Toyota Avanza",
    pickup_date: "22 April 2026",
    price_idr: "850.000",
  },
  files: [
    {
      url: "https://files.tenant.example/invoice/BK-2026-0412.pdf",
      filename: "invoice-BK-2026-0412.pdf",
      mime_type: "application/pdf",
    },
  ],
});
```

9. Contoh lengkap — Python

```python
import hmac, hashlib, json, time, uuid, os, requests

BASE = "https://api.rentalai.id"
SECRET = os.environ["EMBAN_ACTION_SECRET"].encode()

def send_callback(submission_id: str, body: dict):
    raw = json.dumps(body, separators=(",", ":"))
    ts = int(time.time() * 1000)
    sig = "sha256=" + hmac.new(
        SECRET, f"{ts}.{raw}".encode(), hashlib.sha256
    ).hexdigest()

    resp = requests.post(
        f"{BASE}/api/v1/callbacks/{submission_id}",
        data=raw,
        headers={
            "Content-Type": "application/json",
            "X-EMBAN-Timestamp": str(ts),
            "X-EMBAN-Signature": sig,
            "Idempotency-Key": str(uuid.uuid4()),
        },
        timeout=15,
    )
    return resp.status_code, resp.json()

# contoh pemakaian
code, body = send_callback(
    "01J1ZXK7ABCDE",
    {
        "status": "error",
        "error_code": "out_of_stock",
        "error_message": "Unit sudah disewa tanggal tersebut",
        "data": {"alternative_date": "24 April 2026"},
    },
)
print(code, body)
```

10. Retry, idempotency, dan timing

- **Retry**: ulang maks 4× dengan backoff 2s/4s/8s/16s untuk error transient (5xx, network). **Jangan retry** 4xx kecuali `schema_invalid` yang sudah diperbaiki.
- **Idempotency-Key**: WAJIB kalau Anda mungkin retry. EMBAN cache response selama 24 jam — retry dengan key yang sama mengembalikan response cached tanpa double-deliver.
- **Callback timeout**: tenant bisa set `response.callback_timeout_ms` di action definition. Kalau Tenant IT tidak callback dalam window itu, EMBAN akan mengirim pesan `callback_timeout_message` ke customer dan menandai submission timeout. Setelah itu callback late → 409 `invalid_submission_state`.

- **Ordering**: tidak ada jaminan FIFO antar submission berbeda. Setiap submission ditangani independen.

---

11. Checklist pre-go-live

- [ ] Action definition di-register (via flow onboarding tenant) dengan `endpoint.auth.secret_env` valid + secret sudah ada di env EMBAN.
- [ ] Outbound handler di sisi Tenant IT menyimpan `submission_id` dari request pertama EMBAN.
- [ ] Callback sender pakai HMAC dengan secret yang sama.
- [ ] Semua `error_templates` punya entry `default`.
- [ ] File URL di HTTPS + hostname ada di `tenants.file_download_allowlist`.
- [ ] Retry + idempotency-key diimplementasikan.
- [ ] Smoke-test di staging: success path, error path, timeout path, file delivery success + rejected.

Lanjut: **#4 Ticketing API** — endpoint balasan dari sistem ticketing internal Tenant IT (CS role "forward to human").

04 — Ticketing API (CS role escalation)

Audience: Tenant IT. **Prasyarat**: `01-overview.md`, `02-authentication.md`, paham bahwa role **CS** adalah varian tenant yang melayani customer akhir (pelanggan dari tenant).

Ticketing adalah jalur "forward to human" — ketika agent CS tidak mampu / tidak boleh menjawab otomatis, ia membuat ticket dan meneruskannya ke sistem ticketing **milik Tenant IT**. Petugas manusia di sisi Tenant IT membalas, dan EMBAN merelay balasan itu ke customer di channel asli (WhatsApp/Telegram/Email).

Berbeda dari callback (#3) yang memanggil endpoint kustom Tenant IT **per action**, ticketing memakai **satu endpoint tetap per tenant** (`tenants.ticket_webhook_url`).

---

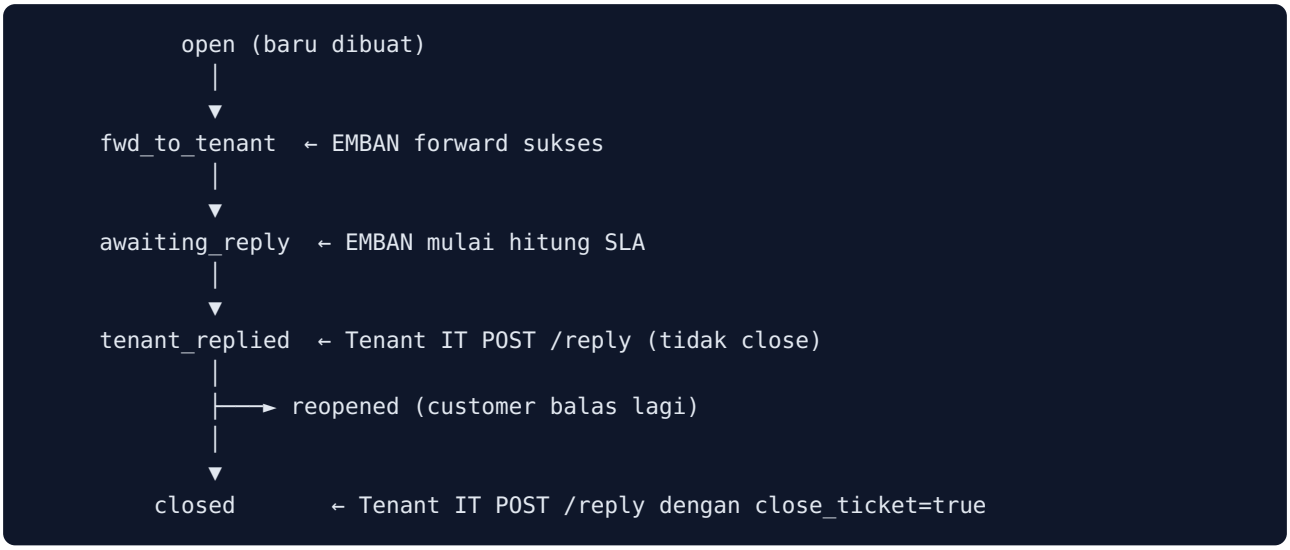
1. Model

Istilah	Artinya
Ticket	Row di tabel `tickets`. Dibuat oleh agent CS saat eskalasi. Punya `ticket_number` customer-facing (format `{prefix}-{yyyymmdd}-{seq6}`, contoh `RAI-20260420-000007`).
Ticket message	Trail percakapan. `sender_type` ∈ `customer` / `agent` / `tenant_system`. Tenant IT reply → `tenant_system`.
Ticket webhook URL	Endpoint di sisi Tenant IT yang menerima notifikasi `ticket.created`. Dikonfigurasi per-tenant di `tenants.ticket_webhook_url`.
Ticket webhook secret	Pre-shared secret HMAC. Disimpan di env EMBAN (referensi via `tenants.ticket_webhook_secret_env`), Tenant IT tahu nilai mentahnya.

Kalau tenant tidak set `ticket_webhook_url`, forward di-skip silently — ticket tetap tercatat di EMBAN, tenant-owner bisa balas manual via Telegram.

---

2. Status lifecycle



Tenant IT **tidak mengatur** transisi `open → fwd_to_tenant → awaiting_reply` — itu engine sisi EMBAN. Tenant IT hanya menyebabkan transisi ke `tenant_replied` atau `closed` via endpoint reply.

---

3. Outbound — EMBAN → Tenant IT (`ticket.created`)

Saat agent CS meng-eskalasi, EMBAN POST ke `tenants.ticket_webhook_url`.

Headers

Header	Nilai
`Content-Type`	`application/json`
`X-EMBAN-Signature`	`sha256=<hex>` — signing input = `timestamp . raw_body`
`X-EMBAN-Timestamp`	Unix millis
`X-EMBAN-Tenant-Id`	tenant ID terkait
`Idempotency-Key`	= `ticket.id` (ULID ticket) — pakai ini untuk dedup di sisi Tenant IT

Body

```
{
  "event": "ticket.created",
  "ticket_id": "01J1ZXK7...",
  "ticket_number": "RAI-20260420-000007",
  "tenant_id": "01J...",
  "customer": {
    "channel": "whatsapp",
    "platform_id": "+628123456789",
    "display_name": "Budi Santoso"
  },
  "subject": "Tidak bisa login ke dashboard",
  "initial_message": "Halo min, saya sudah reset password tapi tetap error...",
  "priority": "normal",
  "sla_due_at": "2026-04-20T14:30:00.000Z",
  "reply_callback_url": "https://api.rentalai.id/api/v1/tickets/01J1ZXK7.../reply",
  "timestamp": "2026-04-20T10:30:00.000Z"
}
```

Yang harus dilakukan Tenant IT:

1. **Verifikasi HMAC** pakai secret yang disepakati (skema sama dengan doc #2 — `sha256=hex(HMAC(secret, ts + "." + rawBody))`).
2. **Deduplikasi** pakai `ticket_id` (atau `Idempotency-Key` yang nilainya sama) — EMBAN me-retry hingga 2× pada 5xx/timeout dengan backoff.
3. **Simpan ticket** di sistem internal Tenant IT, tampilkan ke petugas.
4. **Simpan** `reply_callback_url` — inilah endpoint yang harus dipanggil petugas untuk membalas.
5. **Respond 2xx cepat**. Timeout EMBAN = 10 detik per attempt. Body response tidak dipakai — cukup status code.

Retry & error

- EMBAN retry: max 2× setelah attempt pertama gagal (total 3 attempts), backoff 500ms → 1s.
- Kalau tetap gagal setelah retry, EMBAN log `ticket forward failed after retries` tapi ticket **tetap ada** di sisi EMBAN — tenant-owner bisa balas manual. Tidak ada dead-letter queue otomatis.
- Status response dari Tenant IT:
- `2xx` → `delivered: true`, status tiket `fwd_to_tenant`.
- `4xx` → **tidak** di-retry (client error permanen). Ticket tetap di sisi EMBAN, marked undelivered.
- `5xx` / network error / timeout → di-retry sesuai aturan di atas.

Contoh handler Node (Express)

```
import express from "express";
import { createHmac, timingSafeEqual } from "node:crypto";

const SECRET = process.env.EMBAN_TICKET_WEBHOOK_SECRET;
const SKEW_MS = 5 * 60 * 1000;

const app = express();

// capture raw body untuk HMAC
app.use("/emban/tickets", express.raw({ type: "application/json" }));

app.post("/emban/tickets", (req, res) => {
  const ts = Number(req.header("X-EMBAN-Timestamp"));
  const sig = req.header("X-EMBAN-Signature") ?? "";
  if (!Number.isFinite(ts) || Math.abs(Date.now() - ts) > SKEW_MS) {
    return res.status(401).send("timestamp skew");
  }
  const raw = req.body.toString("utf8");
  const expected =
    "sha256=" +
    createHmac("sha256", SECRET).update(`${ts}.${raw}`).digest("hex");
  const a = Buffer.from(sig);
  const b = Buffer.from(expected);
  if (a.length !== b.length || !timingSafeEqual(a, b)) {
    return res.status(401).send("bad signature");
  }

  const payload = JSON.parse(raw);
  // dedup by ticket_id
  if (await alreadyStored(payload.ticket_id)) {
    return res.status(200).send("dup");
  }
  await storeTicket(payload);
  return res.status(200).send("ok");
});
```

Contoh handler Python (Flask)

```
import hmac, hashlib, time, json
from flask import Flask, request, abort

SECRET = os.environ["EMBAN_TICKET_WEBHOOK_SECRET"].encode()
SKEW_MS = 5 * 60 * 1000

app = Flask(__name__)

@app.post("/emban/tickets")
def handle_ticket():
    ts_hdr = request.headers.get("X-EMBAN-Timestamp", "")
    sig_hdr = request.headers.get("X-EMBAN-Signature", "")
    try:
        ts = int(ts_hdr)
    except ValueError:
        abort(401)
    if abs(int(time.time() * 1000) - ts) > SKEW_MS:
        abort(401)

    raw = request.get_data()                                    # bytes, bukan parsed JSON
    expected = "sha256=" + hmac.new(
        SECRET, f"{ts}.".encode() + raw, hashlib.sha256
    ).hexdigest()
    if not hmac.compare_digest(sig_hdr, expected):
        abort(401)

    payload = json.loads(raw)
    if already_stored(payload["ticket_id"]):
        return "dup", 200
    store_ticket(payload)
    return "ok", 200
```

---

4. Inbound — Tenant IT → EMBAN (balas ticket)

Endpoint ini dipanggil saat petugas Tenant IT mengetik balasan di sistem internal mereka.

```
POST https://api.rentalai.id/api/v1/tickets/:id/reply
```

- `:id` = `ticket_id` ULID (dari event `ticket.created`). **Bukan** `ticket_number` customer-facing.
- Secret HMAC = **sama** dengan secret ticket webhook (simetris). Jangan bikin secret berbeda.

Body

```
{
  "message": "Halo Pak Budi, password sudah kami reset manual. Silakan coba login lagi dengan...",
  "close_ticket": false,
  "sender_label": "Andi dari Admin RAI"
}
```

Field	Tipe	Wajib	Catatan
`message`	string ≥ 1 char	ya	Teks yang akan direlay ke customer, di-prefix otomatis (lihat §4.2).
`close_ticket`	boolean	tidak, default `false`	`true` → status `closed`. Customer tidak bisa balas lagi tanpa ticket dibuka ulang.
`sender_label`	string ≤ 80 char	tidak	Label staff yang terlihat ke customer. Default `"[Admin kami membalas:]"`.

4.1 Response sukses

```
{
  "ok": true,
  "delivered": true,
  "ticket_id": "01J1ZXK7...",
  "customer_channel": "whatsapp",
  "new_status": "tenant_replied"
}
```

Saat customer sudah di-soft-delete:

```
{
  "ok": true,
  "delivered": false,
  "reason": "customer_deleted"
}
```

4.2 Format pesan ke customer

Customer melihat:

```
[Andi dari Admin RAI] Halo Pak Budi, password sudah kami reset manual. Silakan coba login lagi dengan...
```

Kalau `sender_label` tidak diisi:

```
[Admin kami membalas:] Halo Pak Budi, password sudah kami reset manual...
```

Prefix ini hardcoded — Tenant IT tidak bisa override lebih dalam. Kalau butuh disesuaikan, sampaikan ke tim EMBAN.

4.3 Error codes

HTTP	`error_code`	Artinya	Aksi
400	`schema_invalid`	body gagal validasi	cek `details`
401	`missing_auth_headers` / `invalid_signature` / `timestamp_skew` / `unknown_signer`	HMAC gagal	cek secret + body byte-for-byte
404	`ticket_not_found`	`:id` tidak ada di DB	cek ULID benar
409	`ticket_closed`	ticket sudah `closed`	buka ulang lewat prosedur terpisah (belum tersedia di API v1; kontak tim EMBAN)
500	`ticket_lost` / `no_agent_for_tenant`	state rusak	escalate dengan `request_id`
503	`delivery_unavailable` / `delivery_failed`	channel adapter tidak siap / error saat kirim	retry exponential; escalate jika konsisten

4.4 Contoh Node.js

```
import { createHmac, randomUUID } from "node:crypto";

const BASE = "https://api.rentalai.id";
const SECRET = process.env.EMBAN_TICKET_WEBHOOK_SECRET;

async function replyToTicket(ticketId, message, opts = {}) {
  const body = JSON.stringify({
    message,
    close_ticket: opts.close ?? false,
    sender_label: opts.senderLabel,
  });
  const ts = Date.now();
  const sig =
    "sha256=" +
    createHmac("sha256", SECRET).update(`${ts}.${body}`).digest("hex");

  const res = await fetch(`${BASE}/api/v1/tickets/${ticketId}/reply`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-EMBAN-Timestamp": String(ts),
      "X-EMBAN-Signature": sig,
      "Idempotency-Key": randomUUID(),
    },
    body,
  });
  return res.json();
}

// pemakaian
await replyToTicket("01J1ZXK7...", "Halo Pak Budi...", {
  senderLabel: "Andi dari Admin",
  close: true,
});
```

4.5 Contoh Python

```
import hmac, hashlib, json, time, uuid, os, requests

BASE = "https://api.rentalai.id"
SECRET = os.environ["EMBAN_TICKET_WEBHOOK_SECRET"].encode()

def reply_to_ticket(ticket_id: str, message: str, close: bool = False, sender_label: str | None = None):
    body = json.dumps({
        "message": message,
        "close_ticket": close,
        **({"sender_label": sender_label} if sender_label else {}),
    }, separators=(",", ":"))
    ts = int(time.time() * 1000)
    sig = "sha256=" + hmac.new(
        SECRET, f"{ts}.{body}".encode(), hashlib.sha256
    ).hexdigest()

    resp = requests.post(
        f"{BASE}/api/v1/tickets/{ticket_id}/reply",
        data=body,
        headers={
            "Content-Type": "application/json",
            "X-EMBAN-Timestamp": str(ts),
            "X-EMBAN-Signature": sig,
            "Idempotency-Key": str(uuid.uuid4()),
        },
        timeout=15,
    )
    return resp.status_code, resp.json()
```

---

5. Relationship dengan `subscribe_pr_activity` / channel kiri

Tenant IT **tidak** menerima event tiap kali customer membalas setelah ticket di-forward. Alur:

1. EMBAN forward `ticket.created` → Tenant IT.
2. Petugas Tenant IT balas via `POST /reply`.
3. Customer membalas lagi di channel asli → EMBAN menambahkan message ke ticket (status jadi `reopened` kalau sebelumnya `closed`, atau tetap `tenant_replied`).
4. Tapi **Tenant IT tidak di-notify** untuk pesan baru customer. Kalau butuh, tenant harus pull via tool lain, atau (roadmap) `ticket.message_appended` webhook — belum ada di v1.

Kalau flow Anda butuh multi-turn realtime, pertimbangkan: - Minta tenant close ticket setelah balasan final + informasikan customer untuk buka ticket baru kalau butuh. - Atau tunggu fitur `ticket.updated` event di v1.1.

---

6. Checklist pre-go-live

- [ ] Tenant IT sudah expose `POST <ticket_webhook_url>` yang verify HMAC + store ticket + respond < 10s.
- [ ] Secret yang sama dipakai untuk verify outbound ticket DAN sign inbound reply.
- [ ] Deduplikasi pakai `ticket_id` (EMBAN retry hingga 3× pada transient error).
- [ ] Petugas UI menampilkan `ticket_number` customer-facing + `subject` + `priority` + `sla_due_at`.
- [ ] Handler reply di Tenant IT men-set `sender_label` yang identifiable (nama petugas atau tim).
- [ ] Test end-to-end di staging: eskalasi → forward → reply → customer menerima relay dengan prefix.

Lanjut: **#7 Working hours** (skim — awareness saat terima ticket.created di luar jam kerja) atau **#8 Error codes** kalau sudah paham konfigurasi. Untuk lihat receiver implementasi nyata + smoke test: **#11 Reference implementation §2**.

05 — Action Engine

Audience: Tenant IT. **Prasyarat**: `01-overview.md`, `02-authentication.md`. Doc ini adalah **counterpart** `03-callbacks-api.md` — doc #3 menjelaskan *callback balik* setelah action fire; doc #5 ini menjelaskan *outbound request pertama* dari EMBAN ke endpoint Tenant IT.

Action adalah mekanisme yang membuat customer bisa memicu side-effect di sistem Tenant IT lewat percakapan natural. Contoh: customer ngetik "mau booking Toyota Avanza untuk besok", agent CS mengumpulkan field yang dibutuhkan lalu POST ke endpoint booking-manager Tenant IT.

---

1. Alur end-to-end

```
Customer (WhatsApp/TG/Email)
    │ "Mau booking mobil untuk besok"
    ▼
Agent CS (LLM) — collect fields conversationally
    │ nama, tanggal, model
    ▼
EMBAN action-engine
    │ validate → autofill → sign → POST
    ▼
Endpoint Tenant IT (YANG DIDEFINISIKAN DI action.endpoint.url)
    │ process, respond 2xx (+ optional body)
    ▼
 ├── SYNC mode:  response body di-render langsung ke customer
 └── ASYNC mode: customer dapat "sedang diproses" → Tenant IT callback
                 via POST /api/v1/callbacks/:submission_id (doc #3)
```

---

2. Action definition (ringkasan skema)

Action didaftarkan oleh **tenant** (bukan Tenant IT) via tool di percakapan dengan EMBAN. JSON schema lengkap: `src/services/action-schema.ts`. Yang perlu Tenant IT tahu hanya **endpoint + auth + response** blocks.

Contoh minimal:

```
{
  "name": "book_car",
  "description": "Create car rental booking",
  "schema_version": "1.0",
  "trigger_hints": ["booking", "sewa", "rental"],
  "fields": {
    "car_model": {
      "type": "enum_single",
      "required": true,
      "ask": { "id": "Mobil yang mau disewa?" },
      "options": [
        { "value": "avanza", "label": { "id": "Toyota Avanza" } },
        { "value": "innova", "label": { "id": "Toyota Innova" } }
      ]
    },
    "pickup_date": {
      "type": "date",
      "required": true,
      "ask": { "id": "Tanggal pengambilan? (YYYY-MM-DD)" }
    },
    "customer_phone": {
      "type": "phone",
      "autofill": "$sender.platform_id"
    }
  },
  "endpoint": {
    "method": "POST",
    "url": "https://backend.tenant.example/emban/book",
    "auth": {
      "type": "hmac_sha256",
      "secret_env": "EMBAN_BOOK_CAR_SECRET"
    },
    "timeout_ms": 8000,
    "retry": { "max": 3, "backoff": "exponential" }
  },
  "response": {
    "mode": "async",
    "immediate_ack": { "id": "Booking diproses, mohon tunggu..." },
    "success_template": { "id": "OK! Booking {{booking_id}} confirmed." },
    "error_templates": {
      "out_of_stock": { "id": "Maaf, {{car_model}} habis {{pickup_date}}." },
      "default": { "id": "Error: {{error_message}}" }
    },
    "callback_timeout_ms": 600000,
    "callback_timeout_message": { "id": "Maaf sistem lambat, admin akan follow-up." }
  },
  "rate_limit": {
    "per_customer_per_day": 5,
    "global_per_minute": 30
  }
}
```

3. Outbound request dari EMBAN

Saat customer selesai mengisi semua `required` field, EMBAN memanggil `endpoint.url` dengan payload standar.

Headers

Header	Nilai
`Content-Type`	`application/json`
`Idempotency-Key`	`submission_id` (ULID)
`X-EMBAN-Tenant-Id`	tenant ID
`X-EMBAN-Submission-Id`	sama dengan `submission_id`
`X-EMBAN-Schema-Version`	`"1.0"` (sesuai action def)
`X-EMBAN-Signature`	`sha256=<hex(HMAC(secret, body))>` — *hanya body*, **tanpa timestamp** (skema berbeda dari doc #2)
custom headers	apapun yang dikonfigurasi di `endpoint.headers`

Catatan penting: signing scheme untuk **outbound action request** hanya `HMAC(secret, body)`, tanpa timestamp prefix. Ini berbeda dari incoming HMAC di doc #2. Alasan: engine mengendalikan retry, tidak ada risk replay dari pihak ketiga (Tenant IT sendiri yang set secret-nya).

Body (canonical wire shape)

```
{
  "submission_id": "01J1ZXK7ABCDE",
  "action": "book_car",
  "version": 3,
  "data": {
    "car_model": "avanza",
    "pickup_date": "2026-04-22",
    "customer_phone": "+628123456789"
  },
  "callback_url": "https://api.rentalai.id/api/v1/callbacks/01J1ZXK7ABCDE",
  "customer": {
    "channel": "whatsapp",
    "platform_id": "+628123456789",
    "display_name": "Budi Santoso"
  },
  "tenant_id": "01J1TENANT...",
  "timestamp": "2026-04-20T10:30:00.000Z"
}
```

- `data` berisi semua field yang dikumpulkan + hasil autofill. Tenant IT pakai ini sebagai input bisnis.
- `callback_url` hanya dipakai kalau `response.mode = "async"`. Tenant IT simpan ini (atau rekonstruksi dari `submission_id`), pakai saat memanggil callback.
- `submission_id` = pakai ini untuk idempotency di sisi Tenant IT (retry dari EMBAN akan pakai ULID yang sama).

Contoh handler Tenant IT (Express)

```
import express from "express";
import { createHmac, timingSafeEqual } from "node:crypto";

const SECRET = process.env.EMBAN_BOOK_CAR_SECRET;
const app = express();
app.use("/emban/book", express.raw({ type: "application/json" }));

app.post("/emban/book", async (req, res) => {
  const raw = req.body.toString("utf8");
  const expected =
    "sha256=" + createHmac("sha256", SECRET).update(raw).digest("hex");
  const a = Buffer.from(req.header("X-EMBAN-Signature") ?? "");
  const b = Buffer.from(expected);
  if (a.length !== b.length || !timingSafeEqual(a, b)) {
    return res.status(401).json({ error_code: "bad_signature" });
  }

  const payload = JSON.parse(raw);
  if (await isDuplicateSubmission(payload.submission_id)) {
    return res.status(200).json(await getCachedResponse(payload.submission_id));
  }

  // business logic
  const result = await bookingService.create(payload.data);
  if (!result.ok) {
    return res.status(result.status).json({
      error_code: result.error_code,        // dipakai template_key di EMBAN
      error_message: result.reason,
    });
  }

  // SYNC mode: return data langsung, EMBAN render ke customer
  return res.status(200).json({
    booking_id: result.booking.id,
    pickup_date: result.booking.pickup_date,
  });

  // ASYNC mode: cukup 202/200 dengan ack, lalu nanti POST ke callback_url
  // return res.status(202).json({ queued: true });
});
```

4. Autofill variables

Engine mengisi field dengan `autofill` reference sebelum POST. Customer tidak ditanya. Variable yang tersedia:

Variable	Nilai
`$sender.platform_id`	nomor WA / Telegram user_id / email customer
`$sender.display_name`	nama customer (bisa null)
`$sender.channel`	`"whatsapp"` / `"telegram"` / `"email"`
`$tenant.id`	ULID tenant
`$tenant.name`	nama tenant
`$datetime.now`	ISO 8601 UTC, mis. `"2026-04-20T10:30:00.000Z"`
`$datetime.today`	`"2026-04-20"`
`$session.customer_id`	ULID customer di sisi EMBAN
`$session.ticket_id`	ULID ticket (kalau action dipicu dari context ticket)

Variabel yang tidak dikenal → empty string + warning log (bukan error).

5. Field types

Lihat `action-schema.ts` untuk detail. Ringkasan:

Type	Catatan
`string`	plain text; `validation.pattern` regex
`number`	integer/float; `validation.min/max`
`email`	regex basic
`phone`	E.164-ish (`+XXXXXXX`), 7-15 digit
`url`	harus parse sebagai URL
`date`	`YYYY-MM-DD`
`enum_single`	satu value dari `options[]`
`enum_multi`	array dari `options[]` , `min_selections` / `max_selections`
`boolean`	`true` / `false` atau `"ya"/"tidak"/"yes"/"no"/"1"/"0"`
`file`	auto-ingest dari inbound attachment; LLM tidak pass value. Wire: `{filename, mime_type, size_bytes, content_base64}` . Default max 5MB per file, override via `max_size_mb`

Field `sensitive: true` → di audit DB di-redact jadi `[redacted]` tapi tetap dikirim utuh di wire ke Tenant IT.

6. URL templating

`endpoint.url` boleh pakai `{{var}}` untuk path/query. Contoh:

```
"endpoint": {
  "method": "GET",
  "url": "https://backend.example/api/products?q={{query}}&limit=10"
}
```

Substitusi dilakukan sebelum fetch; value di-URL-encode. Variable resolve dari `data` (field + autofill).

---

7. Sync vs async mode

`response.mode: "sync"`

- EMBAN menunggu HTTP response sampai `timeout_ms`.
- Body response (harus JSON) diinterpolasi ke `success_template` / `error_templates` via `{{field}}` mustache-lite.
- Customer menerima hasil dalam satu turn.
- **Gunakan kalau**: operation < 5 detik, deterministic.

`response.mode: "async"`

- EMBAN kirim `immediate_ack` ke customer dan tunggu callback.
- Tenant IT eventually POST ke `callback_url` (lihat doc #3).
- Kalau Tenant IT tidak callback dalam `callback_timeout_ms` (default tidak ada timeout), EMBAN akan kirim `callback_timeout_message` dan mark submission timeout.
- **Gunakan kalau**: operation butuh manual review, dependency eksternal yang lambat, atau proses > 5 detik.

`response.mode: "structured"`

- Engine PASS raw JSON response body ke EMBAN consumer tool (bukan render template ke customer).
- `success_template` / `immediate_ack` / `error_templates` ditolak (warning) — consumer tool yang compose user-facing message.
- Backend WAJIB return JSON dengan shape yang spesifik per consumer tool. Engine validate at consumer level, bukan template level.
- **Gunakan kalau**: tenant-internal data lookup yang owner (bukan end-customer) butuh via agent CS/SA. Contoh kanonik: `refresh_inventory_from_backend` (Phase 7.3) + Template Library trio (`business_metric` / `business_list` / `business_search`).

- **Doc lanjut**: **#12 Template Library** untuk full reference impl Node.js + response contract per template.

---

8. Retry policy (outbound)

Dari sisi EMBAN → Tenant IT: - **5xx response atau network error**: retry sesuai `endpoint.retry.max` (default 3), backoff `exponential` (500ms → 1s → 2s) atau `linear` (1s → 2s → 3s). - **4xx response**: tidak di-retry (client error permanen). Body diparse sebagai `{error_code, error_message}` untuk template lookup. - **Timeout per attempt**: `endpoint.timeout_ms` (default 5000ms).

Tenant IT **wajib idempotent** pakai `submission_id` sebagai dedup key — EMBAN bisa retry request yang sebetulnya sudah berhasil di sisi Tenant IT (network drop sebelum response sampai).

---

9. Response contract

Success (HTTP 200-299)

Body JSON optional. Kalau ada, field-field-nya tersedia sebagai `{{field}}` di `success_template`. Top-level key reserved: `error_code`, `error_message` (abaikan saat success).

Error (HTTP ≥ 400)

Body **direkomendasikan** memuat:

```
{
  "error_code": "out_of_stock",
  "error_message": "Unit tidak tersedia tanggal tersebut"
}
```

- `error_code` → dipakai lookup di `error_templates[code]`. Fallback ke `error_templates.default`.
- `error_message` → tersedia sebagai `{{error_message}}` di template.
- Kalau body tidak JSON valid: engine pakai `error_code: "http_<status>"` + `error_message` = 200 char pertama response body.

---

10. Rate limiting

Action bisa dibatasi lewat `rate_limit` block. Terhitung di sisi EMBAN berdasarkan `action_submissions` row dalam rolling window.

Key	Scope
`per_customer_per_minute` / `_hour` / `_day`	per (customer, action) pair
`global_per_minute` / `_hour`	all customers per action

Hit → customer menerima render dari `error_templates.rate_limited` (atau `.default`), **tidak** ada outbound request ke Tenant IT. Audit: row di `action_throttle_events`.

---

11. Status submission (info untuk debug)

Kalau Tenant IT punya akses dashboard (out of scope v1), submission row memuat `status`:

`status`	Artinya
`pending`	persisted, HTTP request sedang berjalan
`success`	sync 2xx, customer sudah terima template
`awaiting_callback`	async 2xx, engine menunggu Tenant IT POST callback
`callback_delivered`	callback masuk, customer terima render
`error`	gagal di validation, rate limit, atau HTTP layer

---

12. Checklist go-live

- [ ] Tenant sudah register action + test via `dry_run_action` tool di percakapan.
- [ ] Endpoint Tenant IT verify HMAC dengan skema `HMAC(secret, body)` — **tanpa timestamp prefix**.
- [ ] Handler idempotent via `submission_id`.
- [ ] Error response pakai JSON `{error_code, error_message}` supaya template mapping jalan.
- [ ] Sync mode: semua business logic selesai < `timeout_ms`; kalau tidak, ubah jadi async.
- [ ] Async mode: `callback_url` disimpan, callback dikirim dalam `callback_timeout_ms`.
- [ ] Smoke test: success sync, error dengan `error_code`, async happy path (callback masuk), throttle hit.

Lanjut: **#3 Callbacks API** — untuk async action, bagaimana callback balik dari Tenant IT ke EMBAN. Untuk lihat sync action receiver implementasi nyata: **#11 Reference implementation §4** (`query_product_info` stub). Untuk pattern structured-mode + read-only data integrasi (count, list, search), lanjut **#12 Template Library**.

06 — File Delivery

Audience: Tenant IT. **Prasyarat**: `01-overview.md`, `02-authentication.md`, `03-callbacks-api.md`. Doc ini adalah **pendalaman** §6 doc #3 — fokus penuh ke pipeline download, security posture, dan recovery.

Fitur ini (Phase 5.5) memungkinkan Tenant IT mengirim file (invoice PDF, foto unit, dokumen kontrak, dll) ke customer lewat channel aslinya. Tenant IT **tidak** upload file ke EMBAN secara langsung — Tenant IT meng-host file di server sendiri (atau CDN), lalu beri URL-nya di field `files[]` saat POST callback. EMBAN yang akan men-download, validasi, dan meneruskan ke customer.

---

1. Kenapa download-link, bukan upload langsung?

Alternatif (tenant upload file ke EMBAN via multipart) punya tiga masalah:

- **Bandwidth EMBAN** ditanggung untuk ingress dari N tenant → storage sementara.
- **Autentikasi + size cap** harus di-enforce di edge HTTP layer, bukan hanya di engine.
- **Tenant biasanya sudah punya file hosting** (S3, CDN, internal file server). Menduplikasi upload ke EMBAN = double storage.

Download-link pattern: EMBAN fetch dari URL Tenant IT, validasi, forward. Tenant IT tidak perlu tahu detail channel (Telegram media upload, WA multimedia, SMTP attachment). EMBAN yang handle.

Security posture disebut internal **Opsi X** — **tanpa virus scanning**. Aman karena: HTTPS only + hostname allowlist + MIME allowlist + content sniff + size cap. Bukan air-gap level, tapi cukup untuk use case B2C dengan file yang Tenant IT sendiri sudah produce (bukan user-uploaded content).

---

2. Flow singkat

```
Tenant IT backend
    | POST callback /api/v1/callbacks/:submission_id
    | body.files = [{url, mime_type, filename, expected_size_bytes}]
    ▼
EMBAN — callbacks route
    | text message dulu di-deliver ke customer (§6 doc #3)
    | lalu untuk tiap file:
    ▼
EMBAN — file-delivery orchestrator
    | fetch allowlist tenant (fresh per-request)
    ▼
EMBAN — file-download (validasi 6 lapis, lihat §4)
    | tulis ke {FILE_DELIVERY_DIR}/{tenant_id}/{ulid}.{ext}
    ▼
Channel adapter
    | upload ke TG/WA / attach ke email
    ▼
Customer menerima file
    | (text message sudah duluan)
    ▼
Scheduler cleanup (hourly tick, 24h TTL)
    | delete file dari disk, set audit status=expired
```

3. Request shape (recap dari doc #3)

Callback body field `files[]` — max 10 entry per callback:

```
{
  "status": "success",
  "template_key": "booking_confirmed",
  "data": { "booking_id": "BK-2026-0412" },
  "files": [
    {
      "url": "https://files.tenant.example/invoice/BK-2026-0412.pdf",
      "filename": "invoice.pdf",
      "mime_type": "application/pdf",
      "expected_size_bytes": 524288
    }
  ]
}
```

Field	Tipe	Wajib	Catatan
`url`	string URL	ya	**HTTPS wajib**, hostname harus di allowlist
`mime_type`	string	ya	MIME deklarasi — harus di `MIME_ALLOWLIST`
`filename`	string ≤ 255 char	tidak	Nama file yang terlihat customer. Default: auto-generated `{ulid}.{ext}`
`expected_size_bytes`	integer ≥ 0	tidak	Pre-check sebelum fetch. Direkomendasikan biar file oversized di-reject cepat

---

4. Security posture — enam lapis validasi

Urutan pengecekan (first-fail short-circuits):

1. **Declared MIME** harus ada di `MIME_ALLOWLIST` (lihat §5).
2. **URL** parse valid.
3. **Protocol** = `https:` (HTTP ditolak).
4. **Hostname** (lowercase) exact-match di tenant allowlist (lihat §6). Tidak ada wildcard, tidak ada port bypass.
5. **Size cap** — deklarasi `expected_size_bytes` > 10 MB → reject. HEAD pre-check `Content-Length` > 10 MB → reject. **Streaming byte counter** — abort kalau body overrun (HEAD bisa berbohong).
6. **Content sniff** — `file-type` library baca magic bytes dari buffer, dibandingkan dengan declared MIME. Mismatch → reject. (Exception: `text/plain` dan `text/csv` tidak punya magic bytes, sniff di-skip.)

Kalau semua 6 lulus: file ditulis ke disk dengan ULID name + extension dari declared MIME.

---

5. MIME allowlist

Hardcoded di EMBAN (`src/services/file-download.ts`). Nilai default yang sudah approved:

MIME	Ext	Catatan
`application/pdf`	pdf	dokumen umum
`image/png`	png	
`image/jpeg`	jpg	
`image/webp`	webp	
`image/gif`	gif	
`application/msword`	doc	Word 97-2003
`application/vnd.openxmlformats-officedocument.wordprocessingml.document`	docx	Word modern (sniff jadi `application/zip`)
`application/vnd.ms-excel`	xls	Excel 97-2003
`application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`	xlsx	Excel modern (sniff jadi `application/zip`)
`text/csv`	csv	sniff di-skip
`text/plain`	txt	sniff di-skip
`application/zip`	zip	

Belum didukung: video (mp4, mov), audio (mp3, opus, wav), PowerPoint (ppt/pptx). Kalau butuh, ajukan ke tim EMBAN — perlu update allowlist + testing per channel.

---

6. Konfigurasi allowlist per-tenant

Field di DB: `tenants.file_download_allowlist` — JSON array of hostname string (lowercase, exact-match).

Contoh isi:

```
["files.tenant.example", "cdn.tenant.example", "s3.amazonaws.com"]
```

- **Empty / null → semua file di-reject** dengan `allowlist_empty`. Fitur opt-in per tenant.
- **Exact match only** — `files.tenant.example` **tidak** match `sub.files.tenant.example`. Daftar subdomain yang perlu di-allow.
- **Port** tidak di-match — `files.tenant.example:8443` = hostname `files.tenant.example`.
- Dibaca **fresh per-request** — tenant bisa tambah hostname lalu test di callback berikutnya, tanpa restart EMBAN.

Cara Tenant IT minta allowlist di-update: via onboarding tenant CS, atau escalate ke tim EMBAN (belum ada self-service API di v1).

---

7. Aturan size & timeout

Parameter	Default	Env	Catatan
Max size per file	10 MB	`FILE_DELIVERY_MAX_BYTES`	Streaming cap — abort kalau overrun
Max files per callback	10	schema callback	Hard-cap; kirim lebih dari 10 di split callback
Connect timeout (HEAD + GET)	10 detik	`FILE_DELIVERY_CONNECT_TIMEOUT_MS`	Per attempt
Read timeout (GET body)	30 detik	`FILE_DELIVERY_READ_TIMEOUT_MS`	Overall abort
TTL on EMBAN disk	24 jam	`FILE_DELIVERY_TTL_HOURS`	Cleanup scheduler GC
Cleanup tick	60 menit	`FILE_DELIVERY_CLEANUP_INTERVAL_MINUTES`	Hourly sweep

TTL 24 jam artinya: kalau customer tidak menerima file dalam 24 jam setelah callback sukses (mis. customer telegram-nya offline lama), file tidak bisa di-resend otomatis. Tenant IT harus re-POST callback baru. EMBAN tidak retry delivery.

8. Error codes & audit

Setiap attempt (sukses / gagal) menulis row ke `file_deliveries` table. Status value:

Status	Artinya
`downloaded`	Lolos 6 lapis validasi, tersimpan di disk, **belum** dikirim ke channel
`delivered`	Sudah sampai ke customer via channel adapter
`rejected`	Gagal di policy (URL / MIME / allowlist / size) — tidak akan retry
`fetch_failed`	Gagal di transport (HTTP error, timeout, write_failed) — juga tidak di-retry
`delivery_failed`	Download sukses tapi channel adapter throw saat forward
`expired`	Scheduler cleanup men-GC file setelah TTL

Reject reason detail yang tersimpan di kolom `error_code`:

error_code	Lapisan yang gagal
`mime_not_allowed`	MIME di luar allowlist
`bad_url`	URL tidak parseable
`not_https`	URL bukan `https://`
`host_not_allowed`	Hostname tidak ada di tenant allowlist
`allowlist_empty`	Tenant belum konfigurasi allowlist
`size_exceeds_cap`	Deklarasi / HEAD / streaming overrun > 10 MB
`head_failed`	HEAD request error (fallback ke GET, tidak fatal biasanya)
`fetch_failed`	GET 4xx/5xx atau network error
`timeout`	Abort karena connect/read timeout
`content_sniff_unknown`	Declared MIME butuh magic bytes tapi buffer tidak match pattern apapun
`content_sniff_mismatch`	Declared MIME ≠ sniffed MIME
`write_failed`	Disk full / permission error

Tenant IT **tidak** akses langsung tabel ini — kalau butuh audit, escalate ke tim EMBAN dengan `submission_id` atau `tenant_id` + window waktu.

---

9. Response callback dengan file report

EMBAN kembalikan per-file outcome di body callback response:

```
{
  "ok": true,
  "delivered": true,
  "customer_channel": "whatsapp",
  "submission_id": "01J1ZXK7...",
  "files": {
    "delivered": 2,
    "rejected": 1,
    "failed": 0,
    "details": [
      { "url": "https://files.tenant.example/a.pdf", "status": "delivered" },
      { "url": "https://files.tenant.example/b.png", "status": "delivered" },
      {
        "url": "http://bad.example/c.pdf",
        "status": "rejected",
        "error": "not_https"
      }
    ]
  }
}
```

Penting: **text message selalu di-deliver dulu**, tidak tergantung status file. Kalau semua file gagal, customer tetap terima text message dari template. Tenant IT bisa kirim ulang callback hanya dengan `files[]` yang gagal saja (gunakan template minimal atau `template_key` yang sesuai — misal template "lampiran menyusul").

---

10. Channel-specific behavior

Channel	Cara delivery	Batasan platform (eksternal)
WhatsApp	Upload via Baileys multimedia (image/document/audio/video API)	WA sendiri batasi ~16 MB untuk video, ~100 MB untuk document. Kami cap 10 MB, jauh di bawah.
Telegram	Upload via `sendDocument` / `sendPhoto` / `sendVideo` sesuai MIME	TG batasi 50 MB untuk bot upload. Kami cap 10 MB.
Email	Attachment di SMTP2Go (SMTP)	Provider biasanya batasi 25 MB total per email. Kami cap 10 MB per file × max 10 files = 100 MB total; potensi reject di SMTP layer kalau total attachment terlalu besar

Kalau ingin support payload > 10 MB di future, butuh update `FILE_DELIVERY_MAX_BYTES` + test ulang per channel (WA dan email akan jadi bottleneck lebih dulu daripada TG).

---

11. Rekomendasi hosting file dari sisi Tenant IT

- **Gunakan HTTPS** dengan sertifikat yang di-trust OS (Let's Encrypt, DigiCert, dll). Self-signed cert akan gagal di TLS verify.
- **Stable hostname** — hindari URL yang hostname-nya berubah (preview URL, ephemeral container URL). Setiap kali hostname baru, perlu update allowlist.
- **Pre-signed URL** (S3, R2, GCS) OK asal hostname-nya fixed (`s3.amazonaws.com` , bukan `unique-id.s3.amazonaws.com`). Kalau pakai per-object subdomain, allowlist tidak akan cover.
- **Serve dengan `Content-Length` header benar** — HEAD pre-check mengandalkan ini untuk early-reject oversized file, menghemat bandwidth.
- **Jangan pakai redirect** — EMBAN `fetch()` default mengikuti redirect, tapi validasi URL hanya cek URL asli. Re-konfirmasi hostname final kalau pakai shortener/redirector.
- **File TTL di sisi Tenant IT** — minimal 1 jam (buffer retry) + 24 jam untuk in-transit. Rekomendasi: URL valid minimal 2 jam.
- **Naming**: isi `filename` dengan nama yang rapi untuk customer. Kalau di-skip, customer terima nama `{ulid}.{ext}` — kurang user-friendly.

---

12. Checklist go-live

- [ ] Hostname file server sudah di `tenants.file_download_allowlist` (koordinasi via onboarding).
- [ ] Semua URL HTTPS + sertifikat valid.
- [ ] MIME type diisi benar (bukan `application/octet-stream` untuk PDF).
- [ ] `expected_size_bytes` diisi untuk early-reject oversized file.
- [ ] File hosting stabil + TTL ≥ 2 jam.
- [ ] Smoke test: 1 file sukses, 1 file size-overflow, 1 file MIME mismatch, 1 file hostname not allowlisted.
- [ ] Per-file report di response callback di-log untuk audit sendiri (supaya Tenant IT tahu file mana yang gagal, tanpa escalate ke tim EMBAN).

Lanjut: **#4 Ticketing API** — untuk flow "forward to human" kalau agent CS eskalasi.

07 — Working Hours (awareness)

Audience: Tenant IT. **Prasyarat**: `01-overview.md`.

Working hours adalah guardrail di sisi EMBAN — saat customer kirim pesan di luar jam operasional tenant, agent CS **tidak** memanggil LLM. Agent langsung reply dengan `closed_message` yang pre-set.

Anda **tidak** mengkonfigurasi working hours; itu di-set oleh tenant-owner via chat dengan agent EMBAN, atau oleh Upstream saat provisioning. Doc ini ada di sini supaya Anda **paham efeknya** saat receive `ticket.created` atau callback async — bukan untuk Anda set/baca.

---

1. Yang Anda perlu tahu

- **CS role only**. Guard hanya aktif saat agent menjawab customer akhir. Sesi tenant-owner via TG/WA tidak terpengaruh.
- **Berlaku di semua channel** (WA, TG, Email).
- **Callback dari Anda tidak di-gate**. Saat petugas tenant balas ticket pakai `POST /api/v1/tickets/:id/reply`, atau Anda kirim callback async action via `POST /api/v1/callbacks/:submission_id`, EMBAN tetap deliver ke customer kapan saja — termasuk di luar jam kerja. Karena itu reply untuk request yang sudah masuk; bukan inisiasi conversation baru.
- **Reminder/scheduler tidak di-gate**. EMBAN tetap kirim reminder di luar jam kerja kalau jadwalnya jatuh di sana.

---

2. Implikasi untuk receiver Anda

Saat handle `ticket.created` webhook (#4):

- Petugas human mungkin belum online kalau ticket masuk di luar jam kerja. Set expectation di UI internal Anda — mis. badge "after-hours arrival" supaya petugas pagi tahu mana yang prioritas.
- `sla_due_at` di payload sudah dihitung EMBAN termasuk pertimbangan jam kerja (kalau di-set tenant). Anda cukup respect nilai itu, tidak perlu re-compute.

Saat send callback async (#3):

- Tidak perlu delay di sisi Anda menunggu jam buka. Kirim langsung saat ready — engine EMBAN deliver ke customer kapan saja.

---

3. Apakah Anda bisa baca status "buka sekarang?"

Tidak ada endpoint API untuk baca working hours tenant. Pilihan kalau Anda butuh status itu untuk UI internal:

1. **Implement sendiri**. Format JSON shape working hours dijelaskan di §4 di bawah — Anda replicate logic timezone lookup di sisi Anda. Simple.
2. **Tunggu fitur baca-status di v1.x** (roadmap, belum ada di v1).

---

4. JSON shape (referensi opsional)

Kalau Anda perlu replicate logic, ini shape yang dipakai engine:

```
{
  "timezone": "Asia/Jakarta",
  "schedule": {
    "mon": [{ "open": "09:00", "close": "17:00" }],
    "tue": [{ "open": "09:00", "close": "17:00" }],
    "wed": [{ "open": "09:00", "close": "17:00" }],
    "thu": [{ "open": "09:00", "close": "17:00" }],
    "fri": [{ "open": "09:00", "close": "17:00" }],
    "sat": [{ "open": "09:00", "close": "13:00" }],
    "sun": []
  },
  "closed_message": "Halo, toko sedang tutup. Admin akan balas Senin pagi. Terima kasih!"
}
```

Field	Tipe	Catatan
`timezone`	IANA timezone string	Mis. `"Asia/Jakarta"`, bukan abbreviation seperti `WIB`.
`schedule[day]`	array of slots	`[]` = tutup seharian. Multi-slot untuk break siang.
`schedule[day][].open/close`	`"HH:MM"`	Inclusive-open, exclusive-close. Sentinel `"24:00"` valid untuk end-of-day.
`closed_message`	string	Yang customer terima saat tutup.

Catatan semantik (yang sering tricky kalau Anda implement sendiri):

- Multi-slot untuk lunch break: `json "mon": [ { "open": "09:00", "close": "12:00" }, { "open": "13:00", "close": "17:00" } ]` Customer chat 12:30 → terima `closed_message`.
- Cross-midnight (`22:00` - `02:00`) **tidak** didukung. Split jadi 2 slot di 2 hari.
- Holiday override / kalender tanggal merah belum ada di v1.

---

5. Behavior summary

Kondisi	Apa yang customer terima
Dalam jam kerja	LLM jawab normal. Anda mungkin terima `ticket.created` kalau eskalasi terjadi.
Di luar jam kerja	`closed_message` (canned response). LLM **tidak** dipanggil. **Tidak ada** `ticket.created` event.
Working hours tidak di-set tenant	"always open" — guard di-skip.

Implikasi penting: kalau Anda tidak terima ticket dari customer yang harusnya kirim, **bukan berarti webhook Anda broken** — bisa jadi customer kirim di luar jam kerja dan engine sudah balas dengan canned message tanpa LLM/eskalasi.

Lanjut: **#8 Error codes** untuk konsolidasi error_code yang mungkin Anda terima.

08 — Error codes

Audience: Tenant IT. **Prasyarat**: `01-overview.md`. Doc ini adalah referensi silang — konsolidasi semua `error_code` yang mungkin muncul, plus detail rate limit + idempotency.

1. Envelope

Semua error EMBAN membawa envelope standar:

```
{
  "ok": false,
  "error_code": "schema_invalid",
  "message": "Human-readable reason (EN)",
  "request_id": "req_01J...",
  "details": { "...": "opsional" }
}
```

- `error_code` — string stabil. Switch di kode Anda berdasarkan ini.
- `message` — bisa berubah antar versi. Untuk display ke operator, **bukan** untuk logic.
- `request_id` — selalu ada. Sertakan di ticket support.
- `details` — optional, isinya tergantung error. Untuk `schema_invalid` berisi breakdown validasi; untuk `rate_limit_exceeded` berisi `retry_after_ms`.

2. Error codes — by category

2.1 Authentication (HTTP 401)

error_code	Artinya	Aksi
`missing_auth_headers`	`X-EMBAN-Signature` atau `X-EMBAN-Timestamp` absen	kirim keduanya
`invalid_timestamp`	`X-EMBAN-Timestamp` bukan angka Unix millis	kirim string angka desimal
`timestamp_skew`	drift > 5 menit	NTP sync; `details.drift_ms` memberi clue
`unknown_signer`	EMBAN tidak punya secret untuk request ini	cek endpoint benar + tenant/action sudah ter-setup
`invalid_signature`	HMAC mismatch	cek (a) body byte-identical dengan yang di-sign, (b) secret benar, (c) hex lowercase, (d) prefix `sha256=`

Detail sign-flow di `02-authentication.md`.

2.2 Request validation (HTTP 400)

error_code	Artinya	Aksi
`schema_invalid`	Body gagal Zod validation	cek `details.fieldErrors` untuk per-field breakdown
`request_error`	Generic client error (400) — tidak sering terlihat	cek `message`

2.3 Resource not found (HTTP 404)

error_code	Artinya	Dipakai di
`tenant_not_found`	Tenant ULID tidak ada	`/api/v1/tenants/:id`, lifecycle endpoints
`submission_not_found`	Submission ULID tidak ada	`/api/v1/callbacks/:submission_id`
`ticket_not_found`	Ticket ULID tidak ada	`/api/v1/tickets/:id/reply`
`route_not_found`	Method + path tidak match route manapun	cek typo URL

2.4 Conflict (HTTP 409)

error_code	Artinya	Aksi
`idempotency_key_conflict`	Key yang sama dipakai di endpoint berbeda	generate key baru
`invalid_submission_state`	Submission bukan `awaiting_callback` — mungkin sudah delivered, timeout, atau error	**jangan retry**; state tidak kembali
`ticket_closed`	Ticket sudah closed, tidak bisa di-reply	buka ulang via prosedur terpisah (belum di API v1)

2.5 Rate limit (HTTP 429)

error_code	Artinya	Aksi
`rate_limit_exceeded`	Melebihi kuota sliding window	baca `Retry-After` header + `details.retry_after_ms`; backoff eksponensial

Detail kuota di §4.

2.6 Server error (HTTP 500)

error_code	Artinya
`internal_error`	Exception tak tertangkap di handler EMBAN
`action_missing`	Submission merujuk action yang sudah dihapus/corrupt di DB
`action_definition_corrupt`	JSON definition action tidak valid (manual DB tampering / migration bug)
`no_customer_linked`	Submission tanpa customer_id — state rusak
`no_agent_for_tenant`	Tenant tidak punya agent aktif untuk delivery
`secret_not_configured`	`endpoint.auth.secret_env` menunjuk env var yang tidak ada di `process.env` EMBAN
`ticket_lost`	Race condition — ticket hilang antara lookup dan handler

Semua 500-series: **jangan retry otomatis**. Escalate ke tim EMBAN dengan `request_id`.

2.7 Delivery / transient (HTTP 503)

error_code	Artinya	Aksi
`delivery_unavailable`	Channel adapter belum siap (biasanya saat boot / restart)	retry dengan backoff
`delivery_failed`	Channel adapter throw saat kirim (WA disconnect, TG rate limit, SMTP error)	retry dengan backoff; kalau konsisten → escalate

---

3. File delivery — reject reasons

Error ini **tidak** menghentikan callback (HTTP response tetap 200). Mereka muncul di `response.files.details[].error` (format: `"<code>: <message>"`).

`error_code` (di detail)	Lapisan	Perbaikan
`mime_not_allowed`	MIME declaration	pakai MIME dari allowlist (§5 doc #6)
`bad_url`	URL parse	URL tidak valid secara RFC
`not_https`	TLS gate	paksa HTTPS
`host_not_allowed`	Allowlist	minta tim EMBAN tambah hostname
`allowlist_empty`	Opt-in	tenant belum konfigurasi `file_download_allowlist`
`size_exceeds_cap`	Size gate	file > 10 MB (declared / HEAD / streaming)
`head_failed`	HEAD pre-check	tidak fatal — engine fallback ke GET streaming
`fetch_failed`	HTTP GET	4xx/5xx / network error / body missing
`timeout`	Connect atau read	cek server file response time
`content_sniff_unknown`	Magic bytes	declared MIME butuh binary magic, buffer kosong atau unrecognized
`content_sniff_mismatch`	Magic bytes	declared vs sniffed MIME berbeda (mis. declared PDF tapi konten PNG)
`write_failed`	Disk write	server full / permission issue — escalate ke tim EMBAN

4. Rate limits

4.1 Kuota

Kategori	Default	Env var	Endpoint
Standard	60 req/ menit global	`API_RATE_LIMIT_STANDARD`	`/api/v1/tickets/:id/reply`, `/api/v1/callbacks/:submission_id`

Window: **sliding 60 detik**. Keyed per-label (global, bukan per-tenant di v1).

4.2 Response saat hit

```
HTTP/1.1 429 Too Many Requests
Retry-After: 23
Content-Type: application/json

{
  "ok": false,
  "error_code": "rate_limit_exceeded",
  "message": "Too many requests (60/60 in 60s window).",
  "details": { "label": "standard", "retry_after_ms": 22500 },
  "request_id": "req_01J..."
}
```

- `Retry-After` header: integer detik (dibulatkan ke atas).
- `details.retry_after_ms`: lebih presisi, pakai ini untuk logic.

4.3 Retry strategy

Eksponensial, max 4× total attempt:

Attempt	Delay sebelum
1 (original)	0s
2	2s
3	4s
4	8s
5 (final)	16s

Kalau masih 429 setelah 4× retry → gagal permanen. Operator investigasi (traffic legit membludak vs bug loop).

Kalau `details.retry_after_ms > 16s`, **hormati nilai itu** — jangan retry sebelum window reset. Server melihat traffic masih hot.

4.4 Ukuran kuota vs throughput real

- **Standard 60/min** cocok untuk ~86.400 callback/ticket-reply/hari per cluster EMBAN. Tenant yang butuh burst lebih tinggi → escalate.

---

5. Idempotency

5.1 `Idempotency-Key` header

- Semua endpoint `POST`, `DELETE` menerima header ini.

- Value: string ≤ 128 char, rekomendasi UUID atau ULID.
- Cache: 24 jam.
- Scope: per-endpoint. Key yang sama dipakai di endpoint berbeda = 409 `idempotency_key_conflict`.

5.2 Behavior

Kondisi	Behavior
First request dengan key X	Handler jalan normal; response + status di-cache setelah success
Retry dengan key X (endpoint sama)	Return response cached, **no side-effect**
Key X dipakai di endpoint lain	409 `idempotency_key_conflict`
Tidak pakai header	Tidak ada caching, setiap request benar-benar di-process (risk: duplikat)
Key expired (> 24h)	Treated as first request — bisa dobel kalau request asli sukses

5.3 Rekomendasi penggunaan

- **Wajib** untuk retry network transient. Tanpa ini, retry di 5xx/timeout bisa create dua tenant / dua ticket reply / dua callback.
- **Generate fresh key per logical operation**. Jangan reuse key antar tenant berbeda. Jangan reuse key saat Anda memang ingin effect baru.
- **Kombinasi dengan natural key** (seperti `submission_id` di callback, `ticket_id` di ticket reply) memberi dua lapis proteksi — pakai keduanya untuk receiver yang robust.

5.4 Apa yang TIDAK di-cache

- Response 5xx (internal_error). Retry dengan key sama akan benar-benar re-run handler — supaya transient fault bisa pulih.
- Response 401/409/429. Kondisi error ini karena client state (bukan server bug), response cache di-skip.
- Handler yang tidak eksplisit panggil `writeCache` (harusnya semua handler POST panggil ini — kalau ada yang skip itu bug).

---

6. Debug workflow — error muncul, apa yang dicek?

1. **401 dari endpoint yang sebelumnya OK** → timestamp skew (NTP drift, untuk ts-prefixed scheme), atau secret di-rotate tanpa sync. Atau salah skema (lihat **#2 §2** ts-prefixed vs body-only).

2. **400 schema_invalid muncul tiba-tiba** → EMBAN mungkin update schema Zod; cek changelog rilis. Atau body Anda encoding-nya berubah (non-UTF8, BOM, dsb).
3. **429 burst** → traffic Anda lebih dari biasa; backoff + review apakah retry loop bug.
4. **500 internal_error konsisten** → kirim `request_id` + payload (redacted secrets) ke tim EMBAN.
5. **503 delivery_failed konsisten** → channel tenant kemungkinan disconnect (WA session expired, TG bot token revoked). Kontak tenant untuk re-pair.
6. **File delivery rejected semua** → cek `error_code` spesifik: - `allowlist_empty` → tenant belum di-onboard fitur file delivery. - `host_not_allowed` → hostname baru, perlu tambah allowlist. - `content_sniff_mismatch` → file Anda corrupt atau declared MIME salah.

---

7. Changelog stability guarantee

- `error_code`: stable across patch + minor release. Major version (v1 → v2) boleh rename.
- **HTTP status**: stable.
- `message`: string bisa berubah; tidak boleh dipakai untuk logic.
- `details` **shape**: semi-stable — field baru bisa ditambah; existing field tidak di-rename.

Kalau schema/error baru ditambahkan, akan diumumkan via changelog rilis. Subscribe ke kanal komunikasi yang disepakati saat onboarding.

Lanjut: **#9 Contoh End-to-End** untuk melihat semua pola ini dirangkai jadi flow lengkap.

09 — Contoh End-to-End

Audience: Tenant IT. **Prasyarat**: minimal `01-overview.md` dan `02-authentication.md`. Setiap contoh menandai doc yang spesifik dibutuhkan.

Doc ini merangkai endpoint-endpoint individual jadi flow lengkap. Kode contoh pakai Node.js untuk brevity — Python/curl setara ada di doc endpoint masing-masing.

Contoh 1 — Customer triggers custom async action (booking)

Audience: Tenant IT. **Doc terkait**: #3, #5.

Skenario

Tenant "Rent-a-Car Bagus" sudah register action `book_car` lewat chat dengan agent EMBAN. Endpoint backend Tenant IT di `https://backend.rentbagus.com/emban/book`.

Customer ngetik di WA: "Mau booking Avanza untuk hari Rabu." Agent CS mengumpulkan field yang kurang, lalu fire action.

Step 1 — EMBAN POST ke Tenant IT

EMBAN mengirim:

```
POST https://backend.rentbagus.com/emban/book
Content-Type: application/json
Idempotency-Key: 01J1ZXK7SUB...
X-EMBAN-Tenant-Id: 01J1TENANT...
X-EMBAN-Submission-Id: 01J1ZXK7SUB...
X-EMBAN-Schema-Version: 1.0
X-EMBAN-Signature: sha256=abc123...       (HMAC(secret, body) — TANPA timestamp)

{
  "submission_id": "01J1ZXK7SUB...",
  "action": "book_car",
  "version": 3,
  "data": {
    "car_model": "avanza",
    "pickup_date": "2026-04-22",
    "customer_phone": "+628123456789"
  },
  "callback_url": "https://api.rentalai.id/api/v1/callbacks/01J1ZXK7SUB...",
  "customer": {
    "channel": "whatsapp",
    "platform_id": "+628123456789",
    "display_name": "Budi Santoso"
  },
  "tenant_id": "01J1TENANT...",
  "timestamp": "2026-04-20T10:30:00.000Z"
}
```

Step 2 — Tenant IT handler

```
app.post("/emban/book", express.raw({ type: "application/json" }), async (req, res) => {
  const raw = req.body.toString("utf8");
  // Verify (HMAC(secret, body) — no timestamp prefix)
  const expected = "sha256=" +
    createHmac("sha256", process.env.EMBAN_BOOK_CAR_SECRET).update(raw).digest("hex");
  if (!timingSafeEqual(Buffer.from(req.header("X-EMBAN-Signature") ?? ""), Buffer.from(expected))) {
    return res.status(401).json({ error_code: "bad_signature" });
  }

  const payload = JSON.parse(raw);
  if (await bookingStore.exists(payload.submission_id)) {
    return res.status(200).json({ queued: true, duplicate: true });
  }

  // Async mode — queue dan return 202
  await bookingQueue.enqueue({
    submissionId: payload.submission_id,
    callbackUrl: payload.callback_url,
    data: payload.data,
    customer: payload.customer,
  });
  return res.status(202).json({ queued: true });
});
```

Step 3 — customer sementara terima `immediate_ack`

"Booking diproses, mohon tunggu..."

Step 4 — worker Tenant IT proses booking, lalu callback

```
async function processBookingJob(job) {
  const result = await createBookingInLegacySystem(job.data);

  const callbackBody = result.success
    ? {
        status: "success",
        template_key: "booking_confirmed",
        data: {
          booking_id: result.bookingId,
          car_model: job.data.car_model,
          pickup_date: job.data.pickup_date,
          price_idr: result.totalPrice,
        },
        files: result.invoicePdfUrl ? [{
          url: result.invoicePdfUrl,
          filename: `invoice-${result.bookingId}.pdf`,
          mime_type: "application/pdf",
        }] : [],
      }
    : {
        status: "error",
        error_code: "out_of_stock",
        error_message: "Unit sudah disewa tanggal tersebut",
        data: { alternative_date: result.nextAvailable },
      };

  const raw = JSON.stringify(callbackBody);
  const ts = Date.now();
  const sig = "sha256=" + createHmac("sha256",
    process.env.EMBAN_BOOK_CAR_SECRET).update(`${ts}.${raw}`).digest("hex");

  await fetch(job.callbackUrl, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-EMBAN-Timestamp": String(ts),
      "X-EMBAN-Signature": sig,
      "Idempotency-Key": job.submissionId,
    },
    body: raw,
  });
}
```

Step 5 — customer terima rendered template + file

```
[Agent EMBAN]
Booking BK-2026-0412 untuk Toyota Avanza pada 22 April 2026 sudah
dikonfirmasi. Total: Rp850.000.

[attachment: invoice-BK-2026-0412.pdf]
```

Contoh 2 — Ticket escalation + human reply

Audience: Tenant IT. **Doc terkait**: #4.

Skenario

Customer tanya hal teknis kompleks. Agent CS EMBAN tidak bisa jawab dengan SOP yang ada → escalate. Petugas di sistem ticketing Tenant IT balas, customer terima reply via WA.

Step 1 — EMBAN POST ticket.created ke Tenant IT

```
POST https://ticket.rentbagus.com/emban/webhook
Content-Type: application/json
X-EMBAN-Signature: sha256=...          (HMAC(secret, ts + "." + body))
X-EMBAN-Timestamp: 1745145600123
X-EMBAN-Tenant-Id: 01J1TENANT...
Idempotency-Key: 01J1TICKET001...

{
  "event": "ticket.created",
  "ticket_id": "01J1TICKET001...",
  "ticket_number": "RAI-20260420-000007",
  "tenant_id": "01J1TENANT...",
  "customer": {
    "channel": "whatsapp",
    "platform_id": "+628123456789",
    "display_name": "Budi Santoso"
  },
  "subject": "Tidak bisa login ke dashboard",
  "initial_message": "Halo min, saya sudah reset password tapi tetap error...",
  "priority": "normal",
  "sla_due_at": "2026-04-20T14:30:00.000Z",
  "reply_callback_url": "https://api.rentalai.id/api/v1/tickets/01J1TICKET001.../reply",
  "timestamp": "2026-04-20T10:30:00.000Z"
}
```

Step 2 — Tenant IT verify + simpan

Seperti handler di doc #4 §3.

Step 3 — petugas balas di UI Tenant IT

Petugas "Andi" ketik balasan di dashboard internal. Backend Tenant IT memanggil reply endpoint:

```
async function postReply(ticketId, message, senderName, closeTicket = false) {
  const body = JSON.stringify({
    message,
    close_ticket: closeTicket,
    sender_label: `${senderName} dari Admin`,
  });
  const ts = Date.now();
  const sig = "sha256=" + createHmac("sha256",
    process.env.EMBAN_TICKET_WEBHOOK_SECRET).update(`${ts}.${body}`).digest("hex");

  const res = await fetch(
    `https://api.rentalai.id/api/v1/tickets/${ticketId}/reply`,
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "X-EMBAN-Timestamp": String(ts),
        "X-EMBAN-Signature": sig,
        "Idempotency-Key": crypto.randomUUID(),
      },
      body,
    },
  );
  return res.json();
}

await postReply(
  "01J1TICKET001...",
  "Halo Pak Budi, password sudah kami reset manual. Silakan coba login lagi dengan email terdaftar + pa
  "Andi",
  true, // close ticket setelah ini
);
```

Step 4 — customer terima di WA

```
[Andi dari Admin] Halo Pak Budi, password sudah kami reset manual.
Silakan coba login lagi dengan email terdaftar + password 'temp123'.
Mohon ganti password setelah login pertama.
```

Status ticket → `closed`.

Contoh 3 — File delivery happy path + partial failure

Audience: Tenant IT. **Doc terkait**: #3 §6, #6.

Skenario

Tenant IT kirim callback sukses dengan 3 file attached: invoice PDF, foto unit, dan satu file yang hostname-nya belum di allowlist.

Request

```
await fetch("https://api.rentalai.id/api/v1/callbacks/01J1SUB...", {
  method: "POST",
  headers: { /* ... HMAC headers ... */ },
  body: JSON.stringify({
    status: "success",
    template_key: "booking_confirmed",
    data: { booking_id: "BK-2026-0412", car_model: "Toyota Avanza" },
    files: [
      {
        url: "https://files.rentbagus.com/invoice/BK-2026-0412.pdf",
        filename: "invoice.pdf",
        mime_type: "application/pdf",
        expected_size_bytes: 524288,
      },
      {
        url: "https://files.rentbagus.com/photo/avanza-2023.jpg",
        filename: "unit.jpg",
        mime_type: "image/jpeg",
      },
      {
        url: "https://new-cdn.rentbagus.com/terms.pdf",   // hostname belum di allowlist
        filename: "terms.pdf",
        mime_type: "application/pdf",
      },
    ],
  }),
});
```

Response (200)

```
{
  "ok": true,
  "delivered": true,
  "customer_channel": "whatsapp",
  "submission_id": "01J1SUB...",
  "files": {
    "delivered": 2,
    "rejected": 1,
    "failed": 0,
    "details": [
      { "url": "https://files.rentbagus.com/invoice/BK-2026-0412.pdf", "status": "delivered" },
      { "url": "https://files.rentbagus.com/photo/avanza-2023.jpg", "status": "delivered" },
      {
        "url": "https://new-cdn.rentbagus.com/terms.pdf",
        "status": "rejected",
        "error": "host_not_allowed"
      }
    ]
  }
}
```

Yang customer terima

1. Text template "Booking confirmed..." (dari `template_key`).
2. Invoice PDF (di-deliver).
3. Foto unit JPG (di-deliver).
4. **Tidak** terima terms.pdf — itu rejected di sisi EMBAN. Tenant IT bisa escalate tambah `new-cdn.rentbagus.com` ke allowlist, lalu kirim callback baru dengan hanya file itu saja.

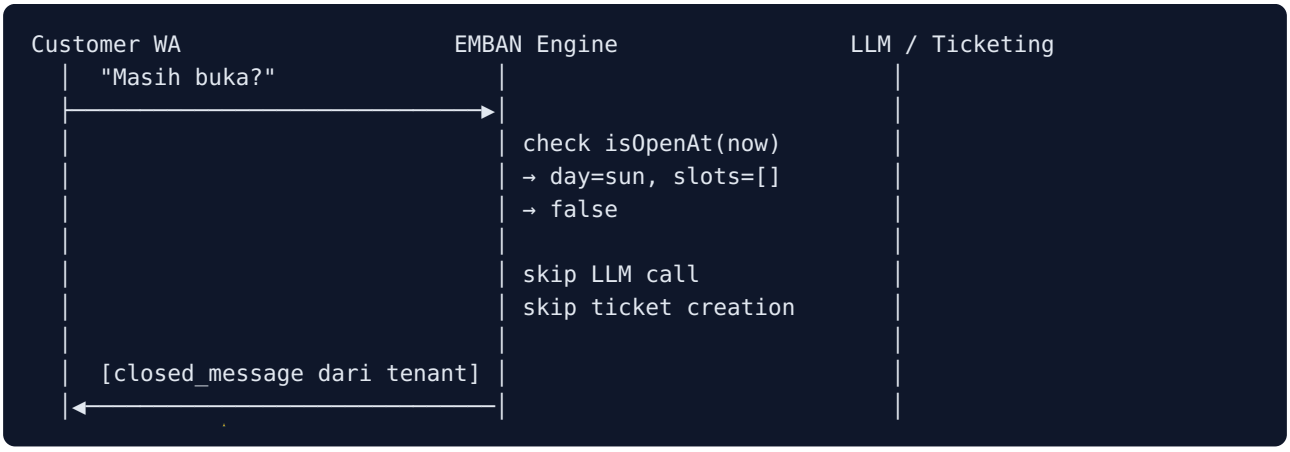
Contoh 4 — Working hours guard fires

Audience: Tenant IT (awareness). **Doc terkait**: #7.

Skenario

Tenant "Warung Mak Ida" jam operasional Senin-Jumat 09:00-17:00, Sabtu 09:00-13:00, Minggu tutup. Customer kirim WA Minggu jam 10 pagi: "Masih buka nggak?"

Flow



Yang customer terima

```
Halo! Warung Mak Ida libur hari Minggu. Kami buka lagi Senin 09:00.
Terima kasih!
```

Tenant IT perspective

- Tidak ada webhook `ticket.created` — karena tidak ada ticket yang dibuat.
- Tidak ada callback API yang terpicu — tidak ada action fire.
- Petugas di Tenant IT dashboard tidak melihat activity.

Kalau Tenant IT perlu tracking "inbound outside hours" untuk insight, harus pull dari log sendiri (channel-level) atau request feature ke tim EMBAN (roadmap `customer.message_dropped` event).

Contoh 5 — Product query (sync action) — RAI reference stub

Audience: Tenant IT. **Doc terkait**: #5.

Skenario

Tenant Rental AI Indonesia (RAI) ingin agent CS bisa jawab pertanyaan customer "Stok TV LED 55 inci tersedia berapa unit?" tanpa harus eskalasi ke petugas. Tenant register action `query_product_info` yang nge-call backend RAI sendiri.

Implementasi reference live di repo: `rai-backend/src/api/routes/emban-product-query.ts` + `rai-backend/src/services/product-query-stub.ts`. Stub mengembalikan katalog hardcoded; di production, ganti `findProducts()` dengan lookup ke inventory system. Wrapper HMAC + idempotency + response shape tetap identik.

Step 1 — Action definition

Tenant register action lewat percakapan dengan agent EMBAN master-bot. Definisi (sesuai schema `src/services/action-schema.ts`):

```
{
  "name": "query_product_info",
  "description": "Lookup ketersediaan + harga produk yang disewakan RAI berdasarkan keyword + category.
  "schema_version": "1.0",
  "trigger_hints": ["stok", "ketersediaan", "harga sewa", "rental"],
  "fields": {
    "query": {
      "type": "free_text",
      "required": true,
      "ask": { "id": "Mau cari produk apa? (sebut keyword, mis. 'tv 55', 'projector')" }
    },
    "category": {
      "type": "enum_single",
      "required": false,
      "default": "all",
      "ask": { "id": "Kategori spesifik? (kalau tidak yakin, ketik 'semua')" },
      "options": [
        { "value": "electronics",      "label": { "id": "Elektronik" } },
        { "value": "office_equipment", "label": { "id": "Peralatan Kantor" } },
        { "value": "all",              "label": { "id": "Semua kategori" } }
      ]
    }
  },
  "endpoint": {
    "url": "https://api.rentalai.id/webhooks/emban-product-query",
    "auth": {
      "scheme": "hmac_body_only",
      "secret_env": "EMBAN_PRODUCT_QUERY_SECRET"
    }
  },
  "response": {
    "mode": "sync",
    "template": {
      "id": "Hasil pencarian \"{{query}}\" ({{count}} produk):\n{{#items}}- {{name}} ({{sku}}): Rp{{dai
    }
  }
}
```

Step 2 — EMBAN POST ke RAI

Customer ngetik di WA: "Mau sewa TV LED, ada yang 55 inci?". Agent CS mengisi `query="tv 55"` + `category="electronics"` (auto-derived dari hint), lalu fire action.

```
POST https://api.rentalai.id/webhooks/emban-product-query
Content-Type: application/json
Idempotency-Key: 01J1ZXK7SUB...
X-EMBAN-Tenant-Id: 01J1TENANT...
X-EMBAN-Submission-Id: 01J1ZXK7SUB...
X-EMBAN-Schema-Version: 1.0
X-EMBAN-Signature: sha256=abc123...       (HMAC(secret, body) — body-only)

{
  "submission_id": "01J1ZXK7SUB...",
  "action": "query_product_info",
  "version": 1,
  "data": {
    "query": "tv 55",
    "category": "electronics"
  },
  "callback_url": "https://api.rentalai.id/api/v1/callbacks/01J1ZXK7SUB...",
  "customer": {
    "channel": "whatsapp",
    "platform_id": "+628123456789",
    "display_name": "Budi Santoso"
  },
  "tenant_id": "01J1TENANT...",
  "timestamp": "2026-04-20T10:30:00.000Z"
}
```

Step 3 — RAI handler verifies + responds (SYNC)

Snippet lengkap di `rai-backend/src/api/routes/emban-product-query.ts`:

```
app.post(
  "/webhooks/emban-product-query",
  {
    preHandler: [
      hmacVerifyPreHandler({
        getSecret: () => env.EMBAN_PRODUCT_QUERY_SECRET || null,
        bodyOnly: true,        // Action Engine: HMAC(secret, body), no ts prefix
      }),
    ],
  },
  async (req, reply) => {
    const parsed = ActionPayloadSchema.safeParse(req.body);
    if (!parsed.success) return reply.code(400).send({
      ok: false, error_code: "schema_invalid",
      details: parsed.error.flatten(),
    });

    const products = findProducts(parsed.data.data.query, parsed.data.data.category ?? "all");

    return reply.code(200).send({
      submission_id: parsed.data.submission_id,
      query: parsed.data.data.query,
      category: parsed.data.data.category ?? "all",
      count: products.length,
      items: products,
    });
  },
);
```

Step 4 — Response body (sync — di-render langsung ke customer)

```
{
  "submission_id": "01J1ZXK7SUB...",
  "query": "tv 55",
  "category": "electronics",
  "count": 1,
  "items": [
    {
      "sku": "ELEC-TV-LED-55",
      "name": "TV LED 55 inci",
      "category": "electronics",
      "daily_price_idr": 250000,
      "available_units": 6,
      "description": "Smart TV LED 55 inci 4K. Bundle dengan stand mobile + HDMI cable."
    }
  ]
}
```

Step 5 — Customer terima rendered template

```
[Agent EMBAN]
Hasil pencarian "tv 55" (1 produk):
- TV LED 55 inci (ELEC-TV-LED-55): Rp250000/hari, stok 6 unit
```

Kenapa stub, bukan real DB?

Tujuan G3 di `docs/integrator/` adalah **kontrak integrasi**, bukan implementasi inventory RAI. Live tenant ganti `findProducts()` di `product-query-stub.ts` dengan call ke ERP / SQL lookup / GraphQL — wrapper HMAC + Zod validate + response shape **tidak perlu diubah**. Pisahnya jelas: route handler = transport, service module = domain logic.

Body-only HMAC bukan ts-prefixed

Action Engine pakai skema HMAC tanpa timestamp prefix (`HMAC(secret, body)`) karena engine kontrol retry sendiri — tidak ada risk replay dari pihak ketiga. Lihat doc #5 §3 catatan dan code di `rai-backend/src/api/middleware/hmac.ts` (option `bodyOnly: true`).

Ringkasan pola umum

Dari 5 contoh di atas:

Pola	Terlihat di
HMAC sign body + timestamp (ts-prefixed)	Contoh 1 Step 4 (callback), Contoh 2
HMAC sign body only (action outbound)	Contoh 1 Step 1, Contoh 5
`Idempotency-Key` untuk retry safety	Contoh 1, 5
`submission_id` / `ticket_id` sebagai natural dedup	Contoh 1, 2, 5
Partial success di response	Contoh 3
No-op di engine saat guard fire	Contoh 4
Sync mode response (template render direct)	Contoh 5

Pola-pola ini konsisten lintas semua endpoint. Kalau Anda mengikuti satu endpoint, endpoint lain akan terasa familiar.

Lanjut: **#10 Glossary** untuk referensi istilah kalau ada yang asing, atau **#11 Reference implementation** untuk lihat code yang menjalankan Contoh 2 + 5 di production.

10 — Glossary

Audience: Tenant IT. **Prasyarat**: tidak ada — file ini referensi cepat istilah yang dipakai di doc #1-#9.

Diorganisir tematik (bukan alfabetis murni) supaya istilah yang berkaitan dekat dibaca bersama. Cari dengan Ctrl-F.

Aktor

Tenant — Pelanggan yang subscribe plan EMBAN. Punya 1 role + 1 plan. Tidak bicara API; cuma chat lewat WA/TG/Email.

Tenant owner — Manusia yang memiliki akun Tenant. Set konfigurasi via chat dengan agent (set_working_hours, register action, dsb). Bukan developer.

Tenant IT — Tim developer di sisi Tenant (khususnya tenant CS) yang membangun backend untuk integrasi: ticketing internal + custom action endpoint. Memegang ticket webhook secret + per-action secret. **Audience utama doc ini.**

Customer — End-user yang dilayani agent CS (khusus role CS). Pelanggan dari Tenant. Bicara via WA/TG/Email.

Agent — Engine LLM yang menjawab message di sisi EMBAN. Per-tenant. Behavior didikte oleh role + plan + tools yang aktif.

Upstream — Pihak yang menjual/menyewakan akses EMBAN-AI ke tenant. Memegang secret platform-level untuk provisioning. **Tidak relevan dengan tugas Anda** — disebut di sini hanya supaya Anda paham siapa yang bikin tenant Anda di awal.

Plan & role

Plan — Tier billing. `basic` (PA dasar), `pro` (PA + tools premium spt web_search, find_nearby), `cs` (untuk role CS).

Role — Tipe agent. `personal_assistant` (asisten owner), `customer_service` (agent CS untuk customer), `store_admin` (admin toko, roadmap). Tidak bisa di-switch setelah create.

1 tenant = 1 role = 1 plan. Combo invalid (mis. `plan=cs` + `role=personal_assistant`) ditolak di provisioning.

Identitas & secret

`tenant_id` — ULID, primary key tenant di EMBAN. Stable lifetime. Anda akan terima value ini di header outbound (`X-EMBAN-Tenant-Id`).

`api_key` — ULID, di-return saat create. Reserved untuk endpoint v2 yang per-tenant. Saat ini tidak aktif.

Ticket webhook secret — Per-tenant. Sign outbound `ticket.created` webhook DAN inbound `/tickets/:id/reply`. Disimpan di env EMBAN sebagai `tenants.ticket_webhook_secret_env` (referensi nama env var).

Action secret — Per-tenant per-action. Sign outbound action request DAN inbound `/callbacks/:submission_id`. Referensi via `endpoint.auth.secret_env`.

Channel

Channel — Saluran tempat agent bicara. `whatsapp` (Baileys), `telegram` (bot API), `email` (SMTP2Go inbound + outbound).

`platform_id` — ID customer/contact di channel. WA = E.164 phone number, TG = numeric user_id, Email = email address.

HMAC & wire-format

HMAC — Hash-based Message Authentication Code. EMBAN pakai SHA-256.

Signing input (ts-prefixed) — `<timestamp_ms> "." <raw_body>`. Untuk inbound ke EMBAN (`/callbacks` , `/tickets/:id/reply`) dan ticket webhook outbound dari EMBAN ke Anda. Default scheme.

Signing input (body-only) — `<raw_body>` saja. Untuk Action Engine outbound dari EMBAN (#5 §3). Engine kontrol retry sendiri jadi tidak perlu replay protection via timestamp.

Signing input (action outbound) — `<raw_body>` saja, **tanpa timestamp**. Berbeda dari incoming. Khusus `POST` EMBAN → action endpoint Tenant IT.

`X-EMBAN-Signature` — Header berisi `sha256=<hex>` (lowercase). Wajib di semua signed request.

`X-EMBAN-Timestamp` — Header berisi Unix millis (string angka). Wajib untuk skema 2-bagian.

Timestamp skew — Selisih `now - X-EMBAN-Timestamp`. Limit 5 menit (`API_HMAC_TIMESTAMP_SKEW_MS`). Drift di luar window = 401 `timestamp_skew`.

Replay window — Same sebagai timestamp skew. Window kecil + idempotency key = proteksi replay yang cukup.

Idempotency

`Idempotency-Key` — Header opsional UUID/ULID di endpoint write. Cache 24 jam per-endpoint. Retry dengan key sama → response cached, no side-effect.

Natural key — Identifier domain (mis. `submission_id` di callback, `ticket_id` di ticket reply) yang juga jadi dedup secara implisit. Pakai bareng `Idempotency-Key` untuk dua lapis proteksi.

Dedup window — Waktu di mana request duplikat di-suppress. `Idempotency-Key` cache 24 jam.

Action engine

Action — Spesifikasi yang membuat customer bisa picu side-effect di backend Tenant IT lewat percakapan natural. Disimpan di `tenant_actions` dengan version.

Action definition — JSON schema action (lihat `src/services/action-schema.ts`): `name`, `fields`, `endpoint`, `response`, `rate_limit`, dll.

Field types — `string`, `number`, `email`, `phone`, `url`, `date`, `enum_single`, `enum_multi`, `boolean`, `file`.

Autofill — Field yang diisi engine otomatis dari context (`$sender.platform_id`, `$tenant.id`, `$datetime.now`, dll). Customer tidak ditanya.

Sync mode (`response.mode: "sync"`) — EMBAN tunggu response HTTP, render langsung. Tidak ada callback.

Async mode (`response.mode: "async"`) — EMBAN kirim `immediate_ack`, tunggu callback dari Tenant IT.

Submission — Satu instance eksekusi action. Row di `action_submissions` dengan ULID. Status: `pending` → `success` / `error` (sync) atau `pending` → `awaiting_callback` → `callback_delivered` (async).

`callback_url` — URL `https://api.rentalai.id/api/v1/callbacks/<submission_id>`. Diberikan ke Tenant IT di outbound action request body. Pakai untuk POST callback async.

`callback_timeout_ms` — Limit menunggu callback. Lewat itu, EMBAN render `callback_timeout_message` ke customer; callback late → 409.

`secret_env` — Reference ke nama env var di sisi EMBAN yang menyimpan secret HMAC. Konvensi: `EMBAN_<ACTION_NAME>_SECRET`.

Schema version — `"1.0"` saat ini. Engine reject definition dengan version asing.

Template rendering

Template — String dengan placeholder `{{var}}` yang di-render dengan data dari callback / response. Two backing structures: - `response.success_template` — sync mode success. - `response.error_templates` — map `key → template`. Dipakai juga untuk async via `template_key`. - `response.immediate_ack` — async mode initial reply. - `response.callback_timeout_message` — fallback kalau callback late.

Mustache-lite — Parser sederhana EMBAN. Support `{{field}}` + `{{nested.key}}`. **Tidak ada** loop (`{{#each}}`), conditional (`{{#if}}`), partial. Logika apapun di-handle Tenant IT sebelum kirim.

`template_key` — Override eksplisit di callback body. Override fallback chain default.

Localizable — Object `{ "id": "..." }` (ID = bahasa Indonesia). Format yang dipakai di template untuk dukung multi-bahasa (saat ini hanya ID).

Ticketing

Ticket — Entitas escalation dari customer ke human petugas Tenant IT. Dibuat oleh agent CS saat tidak mampu/tidak boleh jawab otomatis.

`ticket_id` — ULID, primary key.

`ticket_number` — Customer-facing string `{prefix}-{yyyymmdd}-{seq6}`, contoh `RAI-20260420-000007`.

Status lifecycle: `open` → `fwd_to_tenant` → `awaiting_reply` → `tenant_replied` → `closed` / `reopened`.

`sla_due_at` — Deadline tenant respond. Dipakai untuk SLA breach digest internal. Tidak fatal kalau lewat.

Sender label — Prefix di pesan customer: `[Andi dari Admin] <message>`. Default kalau tidak diset: `[Admin kami membalas:]`.

Forward — Outbound webhook `ticket.created` ke `tenants.ticket_webhook_url` saat ticket dibuat.

File delivery

`files[]` — Array attachment di body callback (max 10). Setiap entry: `url`, `mime_type`, `filename` opsional, `expected_size_bytes` opsional.

`file_download_allowlist` — JSON array hostname per-tenant. Empty/null = semua URL ditolak (opt-in).

MIME allowlist — Set MIME yang diterima EMBAN. Hardcoded di engine (PDF, image PNG/JPG/WebP/GIF, doc/docx/xls/xlsx, csv/txt, zip).

Content sniff — Verifikasi magic bytes file vs declared MIME. Mismatch → reject.

Partial success — Setiap file di-process independen. Reply membawa per-file status: `delivered`, `rejected`, `fetch_failed`, `delivery_failed`.

TTL — Lifetime file di disk EMBAN. 24 jam. Cleanup scheduler hapus + tandai status `expired`.

Working hours

Schedule — Object keyed by 3-letter day (`sun` / `mon` /.../ `sat`), value = array of slot.

Slot — Object `{open, close}` HH:MM. Inclusive-open, exclusive-close.

`24:00` — Sentinel sah hanya di `close`. Artinya "sampai akhir hari".

Multi-slot — Beberapa slot dalam satu hari, mis. break makan siang.

`[]` — Tutup seharian.

`closed_message` — String fallback kalau customer kirim di luar jam. Plain text, no template.

Cross-midnight — Tidak didukung. Split jadi dua hari.

Always open — Behavior default kalau `tenants.working_hours = null`.

Anti-spam (engine internal — bukan API)

Burst silence — Kalau customer kirim > 10 message dalam 60 detik → silenced 5 menit. Tidak ada reply LLM. Owner bisa unsilence via tool.

Action throttle — Per-action rate limit (`per_customer_per_minute` / `hour` / `day`, `global_per_minute` / `hour`). Hit → render `error_templates.rate_limited` ke customer, no outbound.

External rate limit — Per-customer limit jumlah message/jam ke agent (default 20/jam, `EXTERNAL_RATE_LIMIT_MAX`). Hit → canned throttle response.

Convention & glossary teknis

ULID — Universally Unique Lexicographically Sortable Identifier. 26-char string. Dipakai untuk semua primary key (tenant, agent, ticket, submission, dsb).

IANA timezone — String standar `Asia/Jakarta`, `UTC`, `Europe/London`. Bukan abbreviation seperti `WIB`.

`webhook_base_url` — Host canonical untuk call EMBAN dari tenant Anda. Diberikan saat onboarding. Simpan di env, jangan hardcode.

`request_id` — Header/field di setiap response error. Format `req_<ULID>`. Sertakan di ticket support.

Soft delete — Tenant ditandai `deleted_at` tapi row tetap. Hard delete tidak ada di API v1.

Channel adapter — Layer engine yang convert outbound message generic ke API channel spesifik (Baileys send, TG sendMessage, SMTP send).

Notify callback — Function-pointer internal yang dipanggil engine untuk forward message ke channel. Wired di startup.

`api_v1` — Versi API saat ini. Stable across patch + minor; major bump = `/api/v2/...`.

Singkatan & istilah produk

EMBAN-AI — Nama produk. EMBAN = "Asisten Pribadi". **RAI** — Rental AI Indonesia. Vendor utama yang menyewakan EMBAN-AI. **CS** — Customer Service (role). **PA** — Personal Assistant (role). **SA** — Store Admin (role, roadmap). **LLM** — Large Language Model (Gemini Flash, Claude Sonnet/Haiku, dll). **SOP** — Standard Operating Procedure (dokumen yang diupload tenant CS, di-bake ke prompt). **SLA** — Service Level Agreement (deadline tenant respond ticket).

Selesai. Kalau ada istilah belum di-cover, eskalasi ke tim EMBAN untuk ditambah.

Untuk implementasi nyata receiver (ticket webhook + action stub): **#11 Reference implementation**.

12 — Template Library (Business-Data Trio)

Audience: Tenant IT. **Prasyarat**: `01-overview.md`, `02-authentication.md`, `05-action-engine.md` (§2 action JSON skeleton + §3 outbound request signing).

Doc ini adalah **library** dari 3 generic action template yang sudah disediakan EMBAN engine. Owner cukup register action JSON yang sudah disediakan EMBAN ke tenant mereka (lewat `register_tenant_action` di percakapan dengan agent CS/SA), lalu Tenant IT implement satu endpoint backend per template — dispatch internal berdasarkan field `metric_name` / `resource` / `query`. Tidak perlu custom-define action per-query.

> **Kenapa template library?** Praktek lapangan: tenant register 1 custom action per query (`metric_orders_today` , `metric_revenue_mtd` , ...) skalanya jelek. Lebih bersih: 1 endpoint backend menerima dispatch key, internal switch ke handler per-metric. Template library trio (`business_metric` , `business_list` , `business_search`) cover 80% read-only query use case.

Daftar isi

---

1. Kapan pakai template library

Use case	Template	Contoh consumer call
Single-number metric (count/sum/avg/persen/durasi)	`business_metric`	"Berapa order hari ini?" → `query_business_metric(metric_name='orders_today')`
Paginated list dari resource	`business_list`	"List 20 client terbaru" → `query_business_list(resource='clients')`
Free-text search di resource	`business_search`	"Cari produk kemeja batik" → `query_business_search(resource='products', query='kemeja batik')`

Jangan pakai template library untuk: - Side-effect write (create/update/delete) — pakai custom action dengan `response.mode='sync'` atau `'async'` - Multi-step workflow yang butuh field collection conversational — pakai custom action - Sensitive data yang tidak boleh masuk hint_llm — pakai custom action dengan response template yang strip sensitive fields

2. Pola umum

2.1 Structured response mode

Ketiga template memakai `response.mode: "structured"` di action JSON. Konsekuensi: - EMBAN engine TIDAK render `success_template` — consumer tool (EMBAN internal) compose user-facing message dari JSON response mentah - Backend tenant WAJIB return JSON yang strict match contract per template (lihat per-section di bawah) - Field yang tidak di-spec di contract akan diabaikan oleh consumer tool — boleh ada di response untuk debugging, tapi jangan andalkan untuk muncul ke owner

2.2 HMAC signing

Sama dengan `05-action-engine.md §3`:

```
X-EMBAN-Signature: sha256=<hex(HMAC(secret, body))>
```

Secret di-store sebagai env var (atau lewat EMBAN claim-token flow ke DB ciphertext — lihat `05-action-engine.md §6`). Convention: 1 secret per template (`EMBAN_BUSINESS_METRIC_SECRET`, `EMBAN_BUSINESS_LIST_SECRET`, `EMBAN_BUSINESS_SEARCH_SECRET`).

2.3 Register flow (owner-side)

Owner panggil `register_tenant_action` di chat dengan agent CS/SA, paste JSON template:

```
register_tenant_action(definition_json = <JSON dari §3/§4/§5>)
```

Engine validate schema → simpan ke DB → return success. Tool tersedia langsung untuk LLM consumer (`query_business_metric` / `query_business_list` / `query_business_search`).

2.4 Authentication context

Endpoint backend dapat header tambahan dari EMBAN: - `X-EMBAN-Tenant-Id` — tenant ID - `X-EMBAN-Submission-Id` — ULID submission, idempotency key - `X-EMBAN-Schema-Version: 1.0`

Body shape:

```
{
  "submission_id": "01J1ZXK7ABCDE",
  "action": "business_metric",
  "version": 1,
  "data": { ... fields per template ... },
  "context": {
    "tenant_id": "...",
    "channel": "owner-internal"
  }
}
```

Backend ekstrak `data.<field>` untuk dispatch.

---

3. Template 1 — `business_metric`

Single-number query. Backend dispatch berdasarkan `metric_name` string ke per-metric handler.

3.1 Action JSON (paste ke `register_tenant_action`)

```
{
  "name": "business_metric",
  "description": "Generic count/sum/avg/percentage/duration metric query.",
  "schema_version": "1.0",
  "fields": {
    "metric_name": {
      "type": "string",
      "required": true,
      "ask": { "id": "Metric apa yang mau dilihat?" },
      "validation": { "pattern": "^[a-z][a-z0-9_]{0,62}$" }
    },
    "filter_period": {
      "type": "string",
      "required": false,
      "validation": { "max_length": 32 }
    }
  },
  "endpoint": {
    "method": "POST",
    "url": "https://backend.tenant.example/emban/business-metric",
    "auth": {
      "type": "hmac_sha256",
      "secret_env": "EMBAN_BUSINESS_METRIC_SECRET"
    },
    "timeout_ms": 8000,
    "retry": { "max": 2, "backoff": "exponential" }
  },
  "response": {
    "mode": "structured"
  },
  "rate_limit": {
    "global_per_minute": 60
  }
}
```

3.2 Backend response contract

Backend WAJIB return:

```
{
  "metric_name": "orders_today",
  "value": 42,
  "value_type": "count",
  "label": "Total order hari ini",
  "filter_period": "today",
  "as_of": "2026-05-13T10:00:00+07:00"
}
```

Field	Type	Required	Catatan
`metric_name`	string	optional	Echo back `data.metric_name`. Kalau missing, consumer pakai value yang dikirim
`value`	number	**wajib**	Finite number. Integer untuk count, decimal untuk currency/percentage/duration
`value_type`	enum	**wajib**	`"count"` | `"currency"` | `"percentage"` | `"duration_seconds"`. Tool format berbeda per tipe
`label`	string	**wajib**	Non-empty. Dipakai consumer untuk human-readable framing ("Total order hari ini")
`filter_period`	string	optional	Echo back; kalau missing, consumer pakai input
`as_of`	ISO 8601	optional	Timestamp data backend. Kalau ada, di-include di hint_llm

`value_type` **semantics**: - `count` — integer, formatted dengan locale separator (id-ID: "1.234") - `currency` — IDR, formatted `Rp 12.500.000` - `percentage` — 0-100 ratio (bukan 0-1), formatted `12.50%` - `duration_seconds` — detik, formatted otomatis ke detik/menit/jam/hari

3.3 Reference implementation (Node.js / TypeScript)

```
// src/api/routes/emban-business-metric.ts
import type { FastifyInstance } from "fastify";
import { createHmac, timingSafeEqual } from "node:crypto";

interface MetricRequestBody {
  submission_id: string;
  action: string;
  version: number;
  data: {
    metric_name: string;
    filter_period?: string;
  };
  context: {
    tenant_id: string;
    channel: string;
  };
}

interface MetricResponse {
  metric_name: string;
  value: number;
  value_type: "count" | "currency" | "percentage" | "duration_seconds";
  label: string;
  filter_period?: string | null;
  as_of?: string;
}

const SECRET = process.env.EMBAN_BUSINESS_METRIC_SECRET;
if (!SECRET) {
  throw new Error("EMBAN_BUSINESS_METRIC_SECRET env var required");
}

function verifySignature(rawBody: string, header: string | undefined): boolean {
  if (!header || !header.startsWith("sha256=")) return false;
  const provided = Buffer.from(header.slice(7), "hex");
  const expected = createHmac("sha256", SECRET!).update(rawBody).digest();
  if (provided.length !== expected.length) return false;
  return timingSafeEqual(provided, expected);
}

// Per-metric dispatcher. Add entries here as your business needs grow.
async function dispatchMetric(
  metricName: string,
  filterPeriod: string | undefined,
): Promise<MetricResponse | { error_code: string; error_message: string }> {
  switch (metricName) {
    case "orders_today": {
      const count = await queryOrderCountToday();
      return {
        metric_name: "orders_today",
        value: count,
        value_type: "count",
        label: "Total order hari ini",
        filter_period: filterPeriod ?? "today",
        as_of: new Date().toISOString(),
      };
    }
    case "revenue_mtd": {
```

```
      const totalIdr = await queryRevenueMonthToDate();
      return {
        metric_name: "revenue_mtd",
        value: totalIdr,
        value_type: "currency",
        label: "Pendapatan bulan ini",
        filter_period: filterPeriod ?? "mtd",
        as_of: new Date().toISOString(),
      };
    }
    case "avg_resolution_time": {
      const seconds = await queryAvgResolutionSeconds(filterPeriod);
      return {
        metric_name: "avg_resolution_time",
        value: seconds,
        value_type: "duration_seconds",
        label: "Rata-rata waktu resolusi ticket",
        filter_period: filterPeriod ?? "last_30d",
        as_of: new Date().toISOString(),
      };
    }
    default:
      return {
        error_code: "unknown_metric",
        error_message: `Metric '${metricName}' belum di-implement`,
      };
  }
}

// Backend stub functions — replace with your real query layer
async function queryOrderCountToday(): Promise<number> {
  // SELECT COUNT(*) FROM orders WHERE created_at::date = CURRENT_DATE
  return 0;
}
async function queryRevenueMonthToDate(): Promise<number> {
  // SELECT COALESCE(SUM(total_idr), 0) FROM orders
  // WHERE date_trunc('month', created_at) = date_trunc('month', CURRENT_DATE)
  return 0;
}
async function queryAvgResolutionSeconds(_period?: string): Promise<number> {
  return 0;
}

export function registerMetricRoute(app: FastifyInstance) {
  app.post("/emban/business-metric", async (req, reply) => {
    const rawBody = req.body as string;
    const signature = req.headers["x-emban-signature"] as string | undefined;
    if (!verifySignature(rawBody, signature)) {
      return reply.code(401).send({
        error_code: "signature_invalid",
        error_message: "HMAC signature verification failed",
      });
    }
    const body = JSON.parse(rawBody) as MetricRequestBody;
    const result = await dispatchMetric(
      body.data.metric_name,
      body.data.filter_period,
    );
    if ("error_code" in result) {
```

```
      return reply.code(400).send(result);
    }
    return reply.send(result);
  });
}
```

> **Note**: handler harus menerima `rawBody` sebagai string sebelum JSON.parse — HMAC dihitung atas byte body persis (raw). Di Fastify, pakai `rawBody` plugin atau preserve via `addContentTypeParser`. Lihat `11-reference-implementation.md §5` untuk pola complete.

3.4 Consumer tool (EMBAN internal — owner tidak panggil langsung)

LLM agent CS/SA panggil tool `query_business_metric`:

```
query_business_metric(
  metric_name: "orders_today",  // required, lowercase_alnum
  filter_period: "today",        // optional
  action_name: "business_metric" // optional override, default 'business_metric'
)
```

Tool wrap call ke action engine + parse structured response + compose `hint_llm` natural Indonesian sentence dengan `label` dari backend + `value_formatted` (locale-aware).

---

4. Template 2 — `business_list`

Paginated resource list. Backend dispatch berdasarkan `resource` string.

4.1 Action JSON

```
{
  "name": "business_list",
  "description": "Generic paginated resource list.",
  "schema_version": "1.0",
  "fields": {
    "resource": {
      "type": "string",
      "required": true,
      "ask": { "id": "Resource apa yang mau di-list?" },
      "validation": { "pattern": "^[a-z][a-z0-9_]{0,62}$" }
    },
    "limit": {
      "type": "number",
      "required": false,
      "validation": { "min": 1, "max": 100 }
    },
    "cursor": {
      "type": "string",
      "required": false,
      "validation": { "max_length": 256 }
    }
  },
  "endpoint": {
    "method": "POST",
    "url": "https://backend.tenant.example/emban/business-list",
    "auth": {
      "type": "hmac_sha256",
      "secret_env": "EMBAN_BUSINESS_LIST_SECRET"
    },
    "timeout_ms": 8000
  },
  "response": {
    "mode": "structured"
  },
  "rate_limit": {
    "global_per_minute": 60
  }
}
```

4.2 Backend response contract

```
{
  "resource": "orders",
  "items": [
    {
      "id": "ORD-001",
      "title": "Order #001",
      "summary": "Rp 250.000 — Andi (WA)",
      "as_of": "2026-05-13T10:00:00+07:00"
    }
  ],
  "next_cursor": "eyJhZnRlciI6IjAwMSJ9",
  "total": 1234,
  "limit": 20,
  "as_of": "2026-05-13T10:00:00+07:00"
}
```

Field	Type	Required	Catatan
`resource`	string	optional	Echo back
`items`	array	**wajib**	Bisa empty array `[]` (no results)
`items[i].id`	string	**wajib per item**	Non-empty
`items[i].title`	string	**wajib per item**	Non-empty. Dipakai consumer untuk display
`items[i].summary`	string	optional	One-line description. Embedded di hint_llm
`items[i].as_of`	string	optional	Per-item timestamp
`next_cursor`	string | null	optional	Cursor opaque untuk halaman berikutnya. `null` atau missing = end
`total`	integer	optional	Total count keseluruhan (estimasi OK). Untuk progress indicator di hint_llm
`limit`	integer	optional	Echo limit yang di-apply backend
`as_of`	string	optional	Timestamp top-level

Cursor convention: opaque string yang backend interpret sendiri. Bisa base64-encoded JSON, SQL row offset, etc. Consumer pass-through ke parameter `cursor` panggilan berikutnya.

4.3 Reference implementation (Node.js / TypeScript)

```
// src/api/routes/emban-business-list.ts
interface ListRequestBody {
  data: {
    resource: string;
    limit?: number;
    cursor?: string | null;
  };
}

interface ListItem {
  id: string;
  title: string;
  summary?: string;
  as_of?: string;
}

interface ListResponse {
  resource: string;
  items: ListItem[];
  next_cursor?: string | null;
  total?: number;
  limit: number;
  as_of: string;
}

async function dispatchList(
  resource: string,
  limit: number,
  cursor: string | null,
): Promise<ListResponse | { error_code: string; error_message: string }> {
  switch (resource) {
    case "orders": {
      const { rows, nextCursor, total } = await queryRecentOrders(limit, cursor);
      return {
        resource: "orders",
        items: rows.map((r) => ({
          id: r.id,
          title: `Order #${r.short_id}`,
          summary: `${formatIdr(r.total_idr)} — ${r.customer_name} (${r.channel})`,
        })),
        next_cursor: nextCursor,
        total,
        limit,
        as_of: new Date().toISOString(),
      };
    }
    case "clients": {
      const { rows, nextCursor, total } = await queryClients(limit, cursor);
      return {
        resource: "clients",
        items: rows.map((r) => ({
          id: r.id,
          title: r.name,
          summary: r.last_interaction
            ? `Terakhir kontak ${r.last_interaction}`
            : "Belum ada interaksi",
        })),
        next_cursor: nextCursor,
```

```
        total,
        limit,
        as_of: new Date().toISOString(),
      };
    }
    default:
      return {
        error_code: "unknown_resource",
        error_message: `Resource '${resource}' belum di-support`,
      };
  }
}

// Cursor helper: base64-encoded JSON { after: lastId, sort: "created_desc" }
function decodeCursor(cursor: string | null): { after: string } | null {
  if (!cursor) return null;
  try {
    return JSON.parse(Buffer.from(cursor, "base64").toString());
  } catch {
    return null;
  }
}
function encodeCursor(payload: { after: string }): string {
  return Buffer.from(JSON.stringify(payload)).toString("base64");
}

async function queryRecentOrders(
  limit: number,
  cursorRaw: string | null,
): Promise<{
  rows: Array<{
    id: string;
    short_id: string;
    total_idr: number;
    customer_name: string;
    channel: string;
  }>;
  nextCursor: string | null;
  total: number;
}> {
  const cursor = decodeCursor(cursorRaw);
  // SELECT * FROM orders
  // WHERE ($cursor IS NULL OR id < $cursor)
  // ORDER BY id DESC LIMIT $limit + 1
  // → if returned > limit, last row's id = nextCursor.after
  // Replace stub below with real query layer
  const rows: typeof queryRecentOrders extends never ? never : Array<{
    id: string;
    short_id: string;
    total_idr: number;
    customer_name: string;
    channel: string;
  }> = [];
  const nextCursor =
    rows.length > limit ? encodeCursor({ after: rows[limit - 1]!.id }) : null;
  return { rows: rows.slice(0, limit), nextCursor, total: 0 };
}

async function queryClients(_limit: number, _cursor: string | null) {
```

```
  return { rows: [] as Array<{ id: string; name: string; last_interaction: string | null }>, nextCursor
}

function formatIdr(amount: number): string {
  return `Rp ${amount.toLocaleString("id-ID")}`;
}
```

4.4 Consumer tool

```
query_business_list(
  resource: "orders",     // required, lowercase_alnum
  limit: 20,              // optional, default 20, max 100
  cursor: "eyJhZnRlciI...", // optional, dari next_cursor sebelumnya
  action_name: "business_list" // optional override
)
```

Consumer tool clamping limit di range 1-100. Cursor passthrough ke backend.

---

5. Template 3 — `business_search`

Free-text search. Backend dispatch berdasarkan `resource` + apply `query` string.

5.1 Action JSON

```
{
  "name": "business_search",
  "description": "Generic free-text search on a resource.",
  "schema_version": "1.0",
  "fields": {
    "resource": {
      "type": "string",
      "required": true,
      "validation": { "pattern": "^[a-z][a-z0-9_]{0,62}$" }
    },
    "query": {
      "type": "string",
      "required": true,
      "validation": { "min_length": 1, "max_length": 200 }
    },
    "limit": {
      "type": "number",
      "required": false,
      "validation": { "min": 1, "max": 50 }
    }
  },
  "endpoint": {
    "method": "POST",
    "url": "https://backend.tenant.example/emban/business-search",
    "auth": {
      "type": "hmac_sha256",
      "secret_env": "EMBAN_BUSINESS_SEARCH_SECRET"
    },
    "timeout_ms": 8000
  },
  "response": {
    "mode": "structured"
  },
  "rate_limit": {
    "global_per_minute": 60
  }
}
```

5.2 Backend response contract

```
{
  "resource": "products",
  "query": "kemeja batik",
  "items": [
    {
      "id": "P-001",
      "title": "Kemeja Batik Premium",
      "summary": "Size M-XL, Rp 175.000",
      "score": 0.95,
      "as_of": "2026-05-13T10:00:00+07:00"
    }
  ],
  "total_matched": 12,
  "limit": 10,
  "as_of": "2026-05-13T10:00:00+07:00"
}
```

Field	Type	Required	Catatan
`resource`	string	optional	Echo back
`query`	string	optional	Echo back
`items`	array	**wajib**	Bisa empty `[]` (no matches)
`items[i].id`	string	**wajib per item**	Non-empty
`items[i].title`	string	**wajib per item**	Non-empty
`items[i].summary`	string	optional	One-line description
`items[i].score`	number	optional	Relevance score (0-1 atau range backend). Embedded di hint_llm
`items[i].as_of`	string	optional	Per-item timestamp
`total_matched`	integer	optional	Total matches before limit
`limit`	integer	optional	Echo limit yang di-apply
`as_of`	string	optional	Top-level timestamp

5.3 Reference implementation (Node.js / TypeScript)

```
// src/api/routes/emban-business-search.ts
interface SearchRequestBody {
  data: {
    resource: string;
    query: string;
    limit?: number;
  };
}

async function dispatchSearch(
  resource: string,
  query: string,
  limit: number,
): Promise<unknown> {
  switch (resource) {
    case "products": {
      const { rows, totalMatched } = await searchProducts(query, limit);
      return {
        resource: "products",
        query,
        items: rows.map((r) => ({
          id: r.id,
          title: r.name,
          summary: `${r.size_range}, Rp ${r.price_idr.toLocaleString("id-ID")}`,
          score: r.relevance,
        })),
        total_matched: totalMatched,
        limit,
        as_of: new Date().toISOString(),
      };
    }
    case "customers": {
      const { rows, totalMatched } = await searchCustomers(query, limit);
      return {
        resource: "customers",
        query,
        items: rows.map((r) => ({
          id: r.id,
          title: r.name,
          summary: r.phone ?? r.email ?? "Kontak: -",
          score: r.relevance,
        })),
        total_matched: totalMatched,
        limit,
        as_of: new Date().toISOString(),
      };
    }
    default:
      return {
        error_code: "unknown_resource",
        error_message: `Resource '${resource}' belum di-support untuk search`,
      };
  }
}

async function searchProducts(
  _query: string,
  _limit: number,
```

```
): Promise<{
  rows: Array<{
    id: string;
    name: string;
    size_range: string;
    price_idr: number;
    relevance: number;
  }>;
  totalMatched: number;
}> {
  // SELECT id, name, size_range, price_idr, ts_rank(...) AS relevance
  // FROM products
  // WHERE search_tsv @@ plainto_tsquery('indonesian', $query)
  // ORDER BY relevance DESC LIMIT $limit
  return { rows: [], totalMatched: 0 };
}

async function searchCustomers(
  _query: string,
  _limit: number,
): Promise<{
  rows: Array<{
    id: string;
    name: string;
    phone: string | null;
    email: string | null;
    relevance: number;
  }>;
  totalMatched: number;
}> {
  return { rows: [], totalMatched: 0 };
}
```

5.4 Consumer tool

```
query_business_search(
  resource: "products", // required, lowercase_alnum
  query: "kemeja batik", // required, 1-200 char
  limit: 10,             // optional, default 10, max 50 (lebih kecil dari list)
  action_name: "business_search" // optional override
)
```

---

6. Error code mapping

Consumer tool surface error code ke owner via `error_code` field:

Error code	Asal	Owner-facing meaning	Action
`business_only`	EMBAN gate	Tool dipanggil dari role non-business	LLM otomatis pakai tool lain
`validation_error`	EMBAN tool	metric_name / resource / query format invalid	LLM minta clarification
`action_not_registered`	EMBAN tool	Owner belum register action template ini	Owner panggil `register_tenant_action` dengan JSON dari doc ini
`action_not_structured`	EMBAN tool	Action terdaftar tapi `response.mode != "structured"`	Owner re-register dengan mode benar
`action_definition_corrupt`	EMBAN tool	DB row JSON parse fail	Owner re-register
`endpoint_error`	Action engine	Backend tenant return 4xx/5xx atau network failure	Tenant IT cek backend log
`unexpected_async`	EMBAN tool	Engine return `awaiting_async` (tidak seharusnya untuk structured)	Owner re-register dengan structured murni
`backend_response_invalid`	EMBAN tool	Response JSON shape salah (lihat contract per template)	Tenant IT fix backend
Custom code (mis. `unknown_metric`, `db_down`)	Tenant backend	Domain-specific error	Tenant IT log + propagate ke owner

Backend tenant boleh return custom error_code dalam 4xx/5xx response body:

```
{
  "error_code": "unknown_metric",
  "error_message": "Metric 'orders_thisyear' belum di-implement di backend"
}
```

Consumer tool surface `error_code` value ini ke owner — owner dapat info actionable tanpa harus debug EMBAN engine.

---

7. FAQ + anti-patterns

7.1 "Boleh skip HMAC?"

Tidak. Setiap endpoint backend WAJIB verify HMAC walau resource read-only — kalau attacker temukan endpoint URL, mereka bisa enumerate metric_name / resource secara brute force. HMAC stops that. Selama development OK pakai `auth.type = "none"` untuk smoke test (lihat smoke script), tapi production WAJIB HMAC.

7.2 "Backend saya beda language (PHP/Python/Go) — masih bisa?"

Bisa. Skema doc ini language-agnostic — Node.js reference impl di §3.3, §4.3, §5.3 cuma example. Pola HMAC + JSON contract sama persis untuk semua bahasa. Lihat `02-authentication.md` untuk pola HMAC di language lain (PHP openssl_, Python hmac module, etc).

7.3 "Boleh return field tambahan di response?"

Boleh untuk debugging (`request_id` , `cache_hit` , `query_duration_ms`). Tapi consumer tool ignore field yang tidak di-spec di contract — jangan andalkan untuk muncul ke owner. Kalau butuh field ke owner, propose extension ke contract via Github issue.

7.4 "Backend saya lambat (>30 detik) — boleh async?"

Tidak. Template library trio strict structured mode (sync). Kalau backend lambat: 1. Optimize query (index, materialized view, cache layer) 2. Atau bikin custom action mode='async' dengan `immediate_ack` + callback (lihat `03-callbacks-api.md`). Custom action TIDAK pakai template library — perlu custom consumer tool.

7.5 "Multiple tenant beda endpoint per template?"

Tidak masalah. Setiap tenant register action JSON sendiri dengan `endpoint.url` mengarah ke backend mereka. Action engine resolve per-tenant — tidak ada konflik nama.

7.6 Anti-pattern: paraphrase response di backend

Jangan format string di backend (mis. return `"Total order hari ini: 42"`). Backend WAJIB return raw structured field (`value: 42, label: "...", value_type: "count"`). Consumer tool yang compose user-facing message — kalau backend pre-format, tool format Rupiah/ persen/dll tidak akan jalan.

7.7 Anti-pattern: 1 action per metric

```
// JANGAN seperti ini
register_tenant_action({ name: "metric_orders_today", ... });
register_tenant_action({ name: "metric_revenue_mtd", ... });
register_tenant_action({ name: "metric_avg_order_value", ... });
// (20 action terpisah, 20 endpoint backend, 20 secret env var)
```

```
// PAKAI INI (template library pattern)
register_tenant_action({ name: "business_metric", ... }); // sekali register
// Backend dispatch internal berdasarkan metric_name
// → 1 endpoint, 1 secret, 1 action
```

Ratusan metric → 1 action, 1 endpoint, internal switch case di backend.

Cross-references

- `05-action-engine.md` — action JSON skeleton, HMAC outbound, structured mode hingga sini diadopsi
- `02-authentication.md` — HMAC implementation patterns (TS, PHP, Python, Go)
- `08-error-codes.md` — error envelope shape umum
- `11-reference-implementation.md` — RAI backend sebagai pola receiver standar (Node.js + Fastify + rawBody)