

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 nilainya.

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

open (baru dibuat)

|

▼

fwd_to_tenant ← EMBAN forward sukses

|

▼

awaiting_reply ← EMBAN mulai hitung SLA

|

▼

tenant_replied ← Tenant IT POST /reply (tidak close)

|

| —► reopened (customer balas lagi)

|

▼

closed ← Tenant IT POST /reply dengan close_ticket=true

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
<code>Content-Type</code>	<code>application/json</code>
<code>X-ASPRI-Signature</code>	<code>sha256=<hex> — signing input = timestamp . raw_body</code>
<code>X-ASPRI-Timestamp</code>	Unix millis
<code>X-ASPRI-Tenant-Id</code>	tenant ID terkait
<code>Idempotency-Key</code>	= <code>ticket.id</code> (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.ASPRI_TICKET_WEBHOOK_SECRET;

const SKEW_MS = 5 * 60 * 1000;

const app = express();

// capture raw body untuk HMAC

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

app.post("/aspri/tickets", (req, res) => {

  const ts = Number(req.header("X-ASPRI-Timestamp"));

  const sig = req.header("X-ASPRI-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["ASPRI_TICKET_WEBHOOK_SECRET"].encode()

SKEW_MS = 5 * 60 * 1000

app = Flask(__name__)

@app.post("/aspri/tickets")

def handle_ticket():

    ts_hdr = request.headers.get("X-ASPRI-Timestamp", "")

    sig_hdr = request.headers.get("X-ASPRI-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 <code>details</code>
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 <code>closed</code>	buka ulang lewat prosedur terpisah (belum tersedia di

HTTP	error_code	Artinya	Aksi
			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.ASPRI_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-ASPRI-Timestamp": String(ts),

        "X-ASPRI-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["ASPRI_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-ASPRI-Timestamp": str(ts),

        "X-ASPRI-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**.