

## 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/aspri/book",  
  
  "auth": {  
  
    "type": "hmac_sha256",  
  
    "secret_env": "ASPRI_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 ASPRI

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

#### Headers

Header	Nilai
<b>Content-Type</b>	<b>application/json</b>

Header	Nilai
Idempotency-Key	<code>submission_id</code> (ULID)
X-ASPRI-Tenant-Id	tenant ID
X-ASPRI-Submission-Id	sama dengan <code>submission_id</code>
X-ASPRI-Schema-Version	"1.0" (sesuai action def)
X-ASPRI-Signature	<code>sha256=&lt;hex(HMAC(secret, body))&gt;</code> — <i>hanya body</i> , <b>tanpa timestamp</b> (skema berbeda dari doc #2)
custom headers	apapun yang dikonfigurasi di <code>endpoint.headers</code>

**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.ASPRI_BOOK_CAR_SECRET;

const app = express();

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

app.post("/aspri/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-ASPRI-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 ASPRI

        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
<code>\$sender.platform_id</code>	nomor WA / Telegram user_id / email customer
<code>\$sender.display_name</code>	nama customer (bisa null)
<code>\$sender.channel</code>	"whatsapp" / "telegram" / "email"
<code>\$tenant.id</code>	ULID tenant
<code>\$tenant.name</code>	nama tenant
<code>\$datetime.now</code>	ISO 8601 UTC, mis. "2026-04-20T10:30:00.000Z"
<code>\$datetime.today</code>	"2026-04-20"
<code>\$session.customer_id</code>	ULID customer di sisi EMBAN
<code>\$session.ticket_id</code>	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; <code>validation.pattern</code> regex
number	integer/float; <code>validation.min/max</code>
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 <code>options[]</code>
enum_multi	array dari <code>options[]</code> , <code>min_selections/max_selections</code>
boolean	true/false atau "ya"/"tidak"/"yes"/"no"/"1"/"0"
file	auto-ingest dari inbound attachment; LLM tidak pass value. Wire: { <code>filename</code> , <code>mime_type</code> , <code>size_bytes</code> , <code>content_base64</code> }. Default max 5MB per file, override via <code>max_size_mb</code>

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
<code>per_customer_per_minute / _hour / _day</code>	per (customer, action) pair
<code>global_per_minute / _hour</code>	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`:

<code>status</code>	Artinya
<code>pending</code>	persisted, HTTP request sedang berjalan
<code>success</code>	sync 2xx, customer sudah terima template
<code>awaiting_callback</code>	async 2xx, engine menunggu Tenant IT POST callback
<code>callback_delivered</code>	callback masuk, customer terima render
<code>error</code>	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**.