This **Design Basis Report (DBR)** focuses specifically on the **Backend Engine** for the unified Home Assistant Add-on. This backend serves as the bridge between the physical hardware (KNX, Zigbee, Wi-Fi), the **KNX Sentinel** diagnostic logic, and the multi-tenant branded interfaces for **ATS** and **Manara**.

---

1. Backend System Overview

- **Architecture:** Containerized Python-based microservice (FastAPI / Asyncio).
- **Core Role:** Unified Data Bus (UDB) that aggregates Home Assistant state data, performs real-time diagnostics, and enforces brand-specific access policies.
- **Integrations**: * **Inbound:** HA WebSocket API, Supervisor API (Docker health), MQTT (Zigbee2MQTT).
    - **Outbound:** Branded Web UI, External InfluxDB/Grafana (Optional), Push Notification Services.

---

2. Technical Stack & Environment

- **Language:** Python 3.11+ (leveraging `asyncio` for non-blocking I/O).
- **API Framework: FastAPI** (for high-speed UI communication and automatic OpenAPI documentation).
- **Diagnostic Engine:** Custom Python modules (utilizing `numpy` or `pandas` for Z-Score and bus-load calculations).
- **Database:** SQLite (local `/data` partition) for persistent configuration, audit logs, and short-term diagnostic telemetry.

---

3. Core Functional Modules

3.1 The Adaptive Connection Engine (ACE)

- **Logic:** Detects client source IP. If within the local subnet (MikroTik/Aruba range), it optimizes for raw speed. If via Tailscale/VPN, it switches to a compressed data stream to save bandwidth.
- **Heartbeat:** Emits a 60-second "System Healthy" pulse to both the UI and the integrator's remote monitor.

3.2 Identity & Access Management (IAM)

- **Policy Engine:** Intercepts every request from the UI. It references an `access_policies.json` to filter entities.
- **Role Enforcement:** * **Admin:** Full read/write + Diagnostic access.
    - **Standard:** Read/Write on assigned entities.
    - **Tenant:** Filtered view; read-only for system-critical components.

3.3 KNX Sentinel (Diagnostic Integration)

- **Bus Monitor:** Real-time tracking of Telegrams per second (TPS).
- **Anomaly Detection:** Runs the Z-Score algorithm on incoming sensor data to detect "chattering" devices or bus congestion.
- **Health Scoring:** Generates a 0-100 "Home Health Score" passed to the UI.

---

4. Multi-Tenant Branding Logic

The backend acts as the "source of truth" for the brand identity.

- **Header Injection:** Upon login, the backend identifies the deployment (ATS vs. Manara) and sends a JSON payload containing:
    - `brand_id`: "ATS" or "Manara"
    - `primary_color`: HEX code
    - `support_endpoint`: URL
    - `logo_url`: Local path to SVG/PNG

---

5. Security & Data Integrity

- **Token Handling:** Validates Home Assistant Long-Lived Access Tokens (LLAT).
- **Encryption:** AES-256 for any sensitive configuration data stored locally.
- **Isolation:** The diagnostic "Sentinel" loop runs as a background task to ensure that heavy calculations never block the `async` web server handling the UI toggles.

---

6. Performance Targets

---

7. Operational Workflow

1. **Initialization:** Backend starts → Authenticates with HA Supervisor → Loads ATS/Manara branding config.
2. **Monitoring:** Sentinel loop begins monitoring the KNX bus and Zigbee mesh.
3. **UI Handshake:** User opens app → ACE determines connection path → IAM applies profile filters.
4. **Action:** User toggles a light → Backend validates permission → Commands HA Core → Sentinel logs the event.