AAS Portal
API Schema + Netlify Pro Roadmap

Downloadable spec generated from your current portal source archive

Date: January 25, 2026
Source baseline: 69768b1feef6cadf10fc9641-36032244fbb2ed5a (zip)

1. Current State Snapshot

Your portal is already in a strong Netlify-native shape: static UI in /public plus serverless Functions in /netlify/functions. Door and Service pages call a single function (/api/service-router) to resolve door context from Google Sheets CSV sources.

1.1 Repo layout (from the uploaded zip)

Path	Purpose
public/	Static pages and assets (door, service, tech pages, CSS/JS)
public/door/index.html	Door detail page; fetches /api/service-router and mounts Copilot panel
public/service/index.html	Public service request page; fetches /api/service-router
public/tech/*	Internal dashboards: command center, doors, manuals, parts, summary
netlify/functions/service-router.mts	Door lookup API: Registry CSV + Door Master CSV -> JSON payload
netlify/functions/_csv.mts	CSV parser + door ID normalization helper
netlify/functions/copilot-playbooks.json	Deterministic playbooks (models/symptoms -> checklists/parts/manual refs)
docs/COPILOT-ARCHITECTURE.md	Target Copilot API contract and UI behavior
netlify.toml	Deploy config: publish/functions folders, redirects, headers

1.2 What works today

- Door context resolution works end-to-end:
- QR-friendly routes exist (/s, /d) and friendly pages (/service, /door).
- Command Center already does cross-domain search (manuals/parts/service/doors) with caching in localStorage.
- Copilot UI assets are already mounted on the Door page (copilot.js + copilot.css).

1.3 Key gaps to address next

- Hardcoded Google Sheets CSV URLs exist in both the function and front-end scripts.

- CSV fetching/parsing is duplicated across function and client, increasing maintenance and drift risk.
- Every door lookup currently fetches and parses two CSVs (no server-side caching).
- Your /assets/* caching header uses 'immutable' on non-fingerprinted filenames, which can cause users to get stuck on old JS/CSS after deploy.
- Debug mode in /api/service-router?debug=1 can expose raw sheet rows; useful during build, risky in production.

2. Direction Using Netlify Pro

The most practical Netlify Pro-aligned direction is to split the portal into two Netlify sites and centralize data access through Functions. This preserves a frictionless public QR flow while enabling a secured internal portal without custom auth work.

2.1 Two-site deployment pattern

Site	Audience	Purpose
Public QR Site	Customers / anyone scanning QR	Service request flow + limited door context
Tech/Admin Site (Password Protected)	Techs + admin	Command Center, dashboards, Copilot admin actions

Why split: deploy-level password protection is simplest for the internal site; keeping the QR site public avoids adding friction for scans.

2.2 Shared environment variables

Move all sheet URLs, Make webhook URLs, and any tokens into Netlify environment variables (Functions scope). On Pro, configure them once as shared variables and reuse across both sites.

- Recommended variables:
  - REGISTRY_CSV_URL
  - DOOR_MASTER_CSV_URL
  - MANUALS_CSV_URL
  - PARTS_CSV_URL
  - SERVICE_CSV_URL
  - MAKE_COPILOT_WEBHOOK_URL
  - COPILOT_DEBUG_KEY (optional)

3. Canonical Data Model

Standardize on one canonical JSON structure for each entity. Pages can use the canonical object directly, while a short-term 'compat' object can mirror existing field names so you can migrate without breaking UI.

3.1 Door object (canonical)


```
{
  "door_id": "MH-1.1",
  "door_key": "mh-1.1",
  "ids": { "asset_id": "3762", "limble_door_id": "12345" },
  "customer": { "name": "Manning Family Children's Hospital" },
  "location": { "address": "200 Henry Clay Ave, New Orleans, LA 70118", "door_location": "North entrance" },
  "hardware": { "door_type": "Fire door auto swing", "manufacturer": "Horton", "model": "4190" },
  "links": { "request_service": "https://...", "docs": "https://...", "door_page": "/door?id=MH-1.1" },
  "ops": { "status": "Active", "last_inspection": "2026-01-15", "technician": "Marcus", "notes": "Routine inspection, all normal" },
  "sources": { "registry": true, "master": true }
}
```


3.2 Manual object (canonical)


```
{
  "title": "Horton 4190 Installation & Service Manual",
  "manufacturer": "Horton",
  "model": "4190",
  "manual_type": "Service Manual",
  "tags": ["4190", "grinding", "belt", "drive"],
  "url": "https://drive.google.com/..."
}
```


3.3 Part object (canonical)


```
{
  "addison": "HOR-4190-BELT",
  "mfg_number": "4190-0012",
  "manufacturer": "Horton",
  "description": "Drive Belt - Horton 4190",
  "image_url": "https://..."
}
```

3.4 Service record (canonical)


```
{
  "door_id": "MH-1.1",
  "date": "2026-01-15",
  "type": "PM",
  "technician": "Marcus",
  "notes": "Routine inspection, all normal",
  "parts_used": []
}
```

4. API Contracts

All UI pages should call your Functions (/api/*) and receive normalized JSON. Sheets remain the source of truth, but the browser never needs to know sheet URLs or parse CSVs.

4.1 GET /api/door

Purpose: resolve door context for /door and /service. Replaces the need for the browser to touch Google Sheets.

- Request:

```
GET /api/door?doorid=MH-1.1
```

- Response (includes canonical + compat fields):

```
{
  "door": { /* canonical Door object */ },
  "compat": {
    "doorid": "MH-1.1",
    "asset_id": "3762",
    "limble_door_id": "12345",
    "customer": "...",
    "address": "...",
    "door_location": "...",
    "door_type": "...",
    "manufacturer": "...",
    "model": "...",
    "docs": "...",
    "limble_url": "...",
    "status": "...",
    "last_inspection": "...",
    "technician": "...",
    "notes": "...",
    "door_page": ""
  }
}
```

4.2 GET /api/search-index

Purpose: serve Command Center with a ready-to-search index (manuals, parts, service, doors) so the client does not fetch CSVs directly.

```
GET /api/search-index
```

- Response:

```
{
  "generated_at": "2026-01-25T21:30:00Z",
```

"counts": { "doors": 1824, "manuals": 241, "parts": 8392, "service": 12204 },
"items": [
{ "type": "door", "title": "MH-1.1", "subtitle": "Customer • Address • Location", "url": "/door?id=MH-1.1", "search_text": "..." },
{ "type": "manual", "title": "Horton 4190 Service Manual", "subtitle": "Horton • Service Manual", "url": "https://...", "search_text": "..." }
]
}

4.3 POST /api/copilot

Purpose: given door context + a symptom text, return a deterministic plan: checklist, parts, manuals, history glance, and admin actions. V1 uses rule-based playbooks (copilot-playbooks.json).

POST /api/copilot

{
"door_id": "MH-1.1",
"context": { "manufacturer": "Horton", "model": "4190", "door_type": "Fire door auto swing", "customer": "Manning..." },
"symptom": "grinding noise on close",
"tab": "diagnose"
}

- Response skeleton:

{
"copilot_session_id": "cp-20260125-213000-mh-1.1",
"next_actions": [{ "type": "checklist", "steps": [...], "confidence": 0.92, "source": "playbook:horton-4190:grinding" }],
"part_candidates": [...],
"manual_links": [...],
"history_glance": { ... },
"admin_actions": [...]
}

5. Implementation Plan

5.1 Phase 1 - Hardening what exists (1-2 days)

- Move sheet URLs to environment variables (Functions scope). Keep hardcoded fallback during transition.
- Add server-side caching for CSV fetches (Cache API in Netlify Functions) to reduce latency and sheet load.
- Gate or remove /api/service-router?debug=1 in production (require a secret key).
- Adjust /assets/* cache header: remove 'immutable' unless you fingerprint filenames.

5.2 Phase 2 - Centralize data (2-4 days)

- Add /api/door endpoint returning canonical + compat objects.
- Add /api/search-index endpoint, then update Command Center to prefer it (with CSV fallback during rollout).
- Add /api/manuals, /api/parts, /api/service-history for pages that need direct lists.

5.3 Phase 3 - Copilot V1 + Make automation (3-7 days)

- Implement /api/copilot using playbooks, manual links, and parts lookup enrichment.
- Implement Admin buttons to POST to Make webhook(s).
- Add idempotency key format to prevent double-creates.
- Log copilot_session_id, symptom, and selected actions into Service History.

5.4 Phase 4 - Fast, durable reads (future)

- Use Netlify Blobs as a read-optimized store for door/manual/parts/service indexes.
- Add a Scheduled Function to refresh blobs from Sheets on an interval.
- Serve all UI data from blobs for consistent, fast performance.

Appendix A. Field Mapping (Sheets -> Canonical)

This mapping reflects the keys currently used in netlify/functions/service-router.mts and public/assets/command-center.js.

A.1 Door Registry CSV -> Door

Registry header variants	Canonical field
Name / Door Name / Label ID / Label	door.door_id
Asset ID / AssetID / asset id	door.ids.asset_id
Door ID / DoorID / door id	door.ids.limble_door_id
Customer	door.customer.name
Address	door.location.address
Door location / Door Location	door.location.door_location
Door Type / Type	door.hardware.door_type
Manufacturer	door.hardware.manufacturer
Model	door.hardware.model
Docs / Documentation / Doc Link	door.links.docs
QR URL / Work Request URL / Limble Work Request URL / Problem URL	door.links.request_service

A.2 Door Master CSV -> Door.ops

Master header	Canonical field
Status	door.ops.status
Last Inspection / LastInspection	door.ops.last_inspection
Technician	door.ops.technician
Notes	door.ops.notes
QR Link / QRLink / QR link	door.links.door_page

Appendix B. Suggested Function Stubs

If you want a head start, implement new endpoints as separate functions that reuse _csv.mts. Suggested stubs: door.mts, search-index.mts, copilot.mts (playbook-driven).