Atlas Integration Analysis & Fix Plan

Date: October 24, 2025
Issue: Application showing incorrect/mock data instead of real Atlas configuration

---

Current Situation

What Atlas Actually Has (from Web Interface at http://24.123.87.42:8888)

Zones (7 zones):
1. Bar Main (BM) - Mono
2. Bar Sub (BS) - Mono
3. Dining Room (DR) - Mono
4. Red Bird Room (RB) - Mono
5. Party Room (PR) - Mono
6. Outside (O) - Mono
7. Bath (B) - Mono

Groups (6 groups):
1. Bar (B) - 2 zones
2. Dining (D) - 1 zone
3. Red Bird (RB) - 1 zone
4. Party East (PE) - 1 zone
5. Patio (P) - 1 zone
6. Bath Rooms (BR) - 1 zone

Sources (9 sources):
1. Matrix 1 (M1) - Mono
2. Matrix 2 (M2) - Mono
3. Matrix 3 (M3) - Mono
4. Matrix 4 (M4) - Mono
5. Mic 1 (M1) - Mono
6. Mic 2 (M2) - Mono
7. Spotify (S) - Mono
8. Party Room East (PR) - Stereo
9. Party Room West (PR) - Stereo

Atlas Device Info:
- External IP: 24.123.87.42:8888
- Internal IP: 192.168.5.101
- Third Party Control: ENABLED
- Allowed IPs: 192.168.5.99 (TV Controller), 192.168.5.100 (Sports Bar Server)

What the Application is Doing Wrong

Problem 1: Hardcoded Loop Limits

File: `src/app/api/atlas/groups/route.ts`

```
for (let i = 0; i < 8; i++) {  // ❌ WRONG - queries 8 groups when only 6 exist
```

The API is hardcoded to query groups 0-7 (8 groups), but Atlas only has 6 groups configured. This causes:
- Queries for non-existent groups (Group 6 and Group 7)
- Potential errors or empty data
- Confusion about which groups are real

Problem 2: Hardcoded Source Count

File: `src/components/AtlasGroupsControl.tsx`

```
for (let i = 0; i < 14; i++) {  // ❌ WRONG - queries 14 sources when only 9 exist
```

The component queries 14 sources when only 9 exist in the Atlas device.

Problem 3: No Dynamic Discovery

The application doesn't query the Atlas device to discover:
- How many zones actually exist
- How many groups actually exist
- How many sources actually exist
- Which groups are active vs inactive

Problem 4: Database vs Reality Mismatch

The application stores zone/group data in a database, but this data may not match the actual Atlas configuration. The Atlas device is the source of truth, not the database.

Root Cause Analysis

The application was likely developed with:
1. **Mock data** during development (8 zones, 8 groups, 14 sources)
2. **Assumptions** about the Atlas model (probably assumed AZM8 with max capacity)
3. **No dynamic discovery** of actual hardware configuration
4. **Database-first approach** instead of hardware-first approach

The user's requirement is clear: **"Groups should always be active at this location"** and **"System should control real hardware (no mock data)"**

Fix Strategy

Phase 1: Query Real Configuration

1. Add API endpoint to query Atlas device capabilities:
   - Number of zones (via `ZoneName_X` queries until failure)
   - Number of groups (via `GroupName_X` queries until failure)
   - Number of sources (via `SourceName_X` queries until failure)
2. Cache this configuration data to avoid repeated queries

Phase 2: Fix Hardcoded Loops

1. Update `/api/atlas/groups/route.ts` to dynamically determine group count
2. Update `AtlasGroupsControl.tsx` to dynamically determine source count
3. Remove all hardcoded limits (8, 14, etc.)

Phase 3: Fix Group Active State

The Atlas web interface shows all 6 groups exist, but the API needs to check `GroupActive_X` to determine if they're active. The user wants all groups to be active.

Phase 4: Ensure Real Hardware Control

1. Verify all API calls go to the real Atlas device (192.168.5.101)
2. Remove any mock data fallbacks
3. Test that source changes actually control the hardware

Implementation Plan

Step 1: Create Configuration Discovery API

New file: `src/app/api/atlas/discover-config/route.ts`
- Query Atlas to discover actual zone/group/source counts
- Return configuration object

Step 2: Update Groups API

File: `src/app/api/atlas/groups/route.ts`
- Remove hardcoded `for (let i = 0; i < 8; i++)`
- Use discovered configuration
- Only return groups that actually exist

Step 3: Update Groups Control Component

File: `src/components/AtlasGroupsControl.tsx`
- Remove hardcoded `for (let i = 0; i < 14; i++)`
- Use discovered configuration
- Fetch actual source count from Atlas

Step 4: Test Real Hardware Control

- Deploy changes
- Test on bartender remote page
- Verify source changes control real hardware

- Confirm no errors

Expected Outcome

After fixes:
- ✅ Application shows exactly 7 zones (matching Atlas)
- ✅ Application shows exactly 6 groups (matching Atlas)
- ✅ Application shows exactly 9 sources (matching Atlas)
- ✅ All groups show as “active” (not “inactive”)
- ✅ Source changes send real commands to Atlas hardware
- ✅ No mock data or fallbacks
- ✅ No errors in console or logs
- ✅ Bartender remote page works correctly

SSH Connection Issue

Note: SSH connection to 24.123.87.42:2222 is currently timing out. This prevents us from:
- Checking PM2 logs on the remote server
- Deploying directly via SSH

Workaround: We’ll commit fixes to GitHub and provide deployment instructions for the user to pull and restart the application manually.