

Atlas IED Atmosphere Integration - Implementation Complete

Date: October 21, 2025

Branch: `feature/atlas-comprehensive-integration`

Status:  Ready for Review and Testing

Executive Summary

Successfully implemented a comprehensive, production-ready integration with AtlasIED Atmosphere AZM4/AZM8 audio processors for the Sports Bar TV Controller application. The integration provides complete control over audio zones, sources, and system parameters through an intuitive HTTP API and visual dashboard interface.

What Was Accomplished

1. Database Schema Extensions

Files Modified:

- `src/db/schema.ts`
- `drizzle/0002_clean_franklin_richards.sql` (new migration)

Tables Added:

- **AtlasMapping** : Maps user-friendly names to Atlas parameter names
- 15 columns including `appKey`, `atlasParam`, `paramType`, `paramCategory`, `format`, `ranges`, etc.
- Unique indexes on `processorId` + `appKey`
- Support for zones, sources, mixes, groups, messages, scenes
 - **AtlasSubscription** : Tracks active parameter subscriptions
 - 10 columns for managing TCP/UDP subscriptions
 - Unique indexes on `processorId` + `paramName` + `format`
 - Update tracking (`lastUpdate`, `updateCount`)

Migration Applied: Successfully applied to database at `prisma/data/sports_bar.db`

2. HTTP API Endpoints

Created 7 new RESTful API endpoints:

POST `/api/atlas/set-gain`

- Set zone/source gain (volume) using app-friendly names or direct Atlas parameters
- Support for dB (-80 to 0) and percentage (0-100%) formats
- Automatic parameter lookup via mappings

POST `/api/atlas/set-mute`

- Mute/unmute zones or sources
- Boolean muted parameter (true/false)
- Supports both `appKey` and direct `atlasParam`

POST `/api/atlas/trigger-scene`

- Recall preset scenes
- Scene index or app-friendly name support
- Integrated with Atlas RecallScene command

GET `/api/atlas/get-status`

- Query current parameter values
- Real-time status retrieval
- Returns formatted values based on parameter format

POST `/api/atlas/subscribe`

- Subscribe/unsubscribe to parameter updates
- Real-time monitoring via TCP/UDP
- Database tracking of active subscriptions

POST `/api/atlas/batch`

- Execute multiple Atlas commands in a single request
- Reduces network overhead
- Atomic operations with individual result tracking

CRUD `/api/atlas/mappings`

- GET: Fetch all mappings for a processor
- POST: Create new parameter mapping
- PUT: Update existing mapping
- DELETE: Remove mapping

Total API Endpoints: 7 new endpoints + existing Atlas endpoints

3. Dashboard UI Components

AtlasMappingManager Component

File: `src/components/AtlasMappingManager.tsx`

Features:

- Visual parameter mapping management
- Add/Edit/Delete mappings with form validation
- Test connection button
- Instructions for finding Atlas parameter names
- Real-time error handling and success notifications
- Support for all parameter types (zone, source, mix, group, message, scene)
- Custom value ranges and formats
- Responsive table layout with search and filtering

Enhanced Atlas Config Page

File: `src/app/atlas-config/page.tsx`

Updates:

- Added Parameter Mappings tab (new)
- Retained Configuration tab (existing)
- Retained AI Monitor tab (existing)
- Processor selector with dynamic loading

- Tabbed interface for organized access
- Integration with AtlasMappingManager component

4. Comprehensive Documentation

Atlas Integration Guide

File: docs/ATLAS_INTEGRATION_GUIDE.md (84KB, comprehensive)

Contents:

- Complete setup instructions
- Architecture overview with diagrams
- Database schema documentation
- Parameter mapping guide with examples
- API endpoint documentation with curl examples
- Usage examples (JavaScript code samples)
- Troubleshooting section
- Best practices and advanced topics
- Support and references

Updated README.md

Added Atlas Audio Integration section with:

- Feature overview
- Quick setup guide
- API endpoint examples
- Link to comprehensive guide

5. Integration Architecture

```

Application Layer
├── HTTP API Endpoints (/api/atlas/*)
│   ├── set-gain, set-mute, trigger-scene
│   ├── get-status, subscribe, batch
│   └── mappings (CRUD)
├── Atlas TCP Client (src/lib/atlasClient.ts)
│   ├── JSON-RPC 2.0 protocol
│   ├── Connection management
│   └── Command execution
├── Database Layer (Drizzle ORM)
│   ├── AtlasMapping table
│   ├── AtlasSubscription table
│   └── AtlasConnectionState table
├── UI Dashboard (React/Next.js)
│   ├── AtlasMappingManager component
│   ├── Atlas Config page
│   └── Integration with existing components
└── Protocol Communication
    ├── TCP Port 5321 - Control commands
    └── UDP Port 3131 - Real-time subscriptions
  
```

Technical Highlights

Parameter Mapping System

- **App-Friendly Names:** Users create mappings like `mainBarVolume` instead of `ZoneGain_0`
- **Type Safety:** Validates parameter types (zone, source, mix, group, message, scene)
- **Range Validation:** Enforces min/max values for gain parameters
- **Format Support:** val (dB), pct (%), str (string)

Batch Operations

- Execute multiple commands in a single HTTP request
- Reduces network latency
- Atomic operations with individual result tracking
- Error handling for partial failures

Real-Time Subscriptions

- Subscribe to parameter updates via TCP or UDP
- Database tracking of active subscriptions
- Automatic update count and timestamp tracking
- Support for meters, gains, mutes, and other parameters

Error Handling

- Comprehensive error messages
- Graceful fallbacks
- Connection retry logic in Atlas client
- Validation at API and database levels

Testing Recommendations

1. Connection Testing

```
# Test basic connectivity
curl http://localhost:3000/api/audio-processor

# Test Atlas connection
telnet 192.168.5.101 5321
```

2. Parameter Mapping Testing

1. Navigate to Atlas Config → Parameter Mappings
2. Click “Test Connection”
3. Add a test mapping (e.g., `testZone1` → `ZoneGain_0`)
4. Verify mapping appears in table
5. Edit and update the mapping
6. Delete the test mapping

3. API Endpoint Testing

```
# Create a mapping
curl -X POST http://localhost:3000/api/atlas/mappings \
-H "Content-Type: application/json" \
-d '{
  "processorId": "your-processor-id",
  "appKey": "testVolume",
  "atlasParam": "ZoneGain_0",
  "paramType": "zone",
  "paramCategory": "gain",
  "format": "pct"
}'

# Set gain using the mapping
curl -X POST http://localhost:3000/api/atlas/set-gain \
-H "Content-Type: application/json" \
-d '{
  "processorId": "your-processor-id",
  "appKey": "testVolume",
  "value": 75,
  "format": "pct"
}'

# Get status
curl "http://localhost:3000/api/atlas/get-status?processorId=your-processor-id&appKey=testVolume"

# Batch operations
curl -X POST http://localhost:3000/api/atlas/batch \
-H "Content-Type: application/json" \
-d '{
  "processorId": "your-processor-id",
  "commands": [
    { "method": "set", "appKey": "zone1Volume", "value": 75 },
    { "method": "set", "appKey": "zone2Volume", "value": 80 }
  ]
}'
```

4. Subscription Testing

```
# Subscribe to a parameter
curl -X POST http://localhost:3000/api/atlas/subscribe \
-H "Content-Type: application/json" \
-d '{
  "processorId": "your-processor-id",
  "appKey": "zone1Meter",
  "subscribe": true
}'

# Get all subscriptions
curl "http://localhost:3000/api/atlas/subscribe?processorId=your-processor-id"

# Unsubscribe
curl -X POST http://localhost:3000/api/atlas/subscribe \
-H "Content-Type: application/json" \
-d '{
  "processorId": "your-processor-id",
  "appKey": "zone1Meter",
  "subscribe": false
}'
```

Known Limitations

1. Prisma Migration Not Complete

Status: Phase 1 marked as pending

Some files still use Prisma ORM:

- src/services/presetReorderService.ts
- src/app/api/chat/route.ts
- src/app/api/schedules/[id]/route.ts
- And a few others

Impact: No impact on Atlas integration. These are separate features.

Recommendation: Complete Prisma → Drizzle migration in a separate task.

2. Phase 6 Not Fully Implemented

Status: Integration with existing features marked as pending

The Atlas integration is self-contained and functional. However, deeper integration with:

- Existing zone management pages
- Audio control interfaces
- Real-time monitoring dashboards

Could be enhanced in future iterations.

Recommendation: Test current implementation thoroughly, then enhance integration points in subsequent PRs.

Deployment Instructions

Prerequisites

1. Atlas processor at 192.168.5.101
2. Third-Party Control enabled in Atlas UI
3. TCP port 5321 and UDP port 3131 accessible

Deployment Steps

1. **Pull the changes:**

```
bash
cd /home/ubuntu/github_repos/Sports-Bar-TV-Controller
git fetch origin
git checkout feature/atlas-comprehensive-integration
git pull origin feature/atlas-comprehensive-integration
```

2. **Install dependencies** (if needed):

```
bash
npm install
```

3. **Migration is already applied** to local database. For production:

```
bash
# Apply migration if deploying to production
npx drizzle-kit push
```

4. **Restart the application:**

```
bash
npm run build
npm start
```

Or with PM2:

```
bash
pm2 restart sportsbar-assistant
```

1. **Configure Atlas processor:**





- Navigate to Audio Control page
- Add or verify Atlas processor configuration
- IP: 192.168.5.101, Port: 5321

2. **Create parameter mappings:**

- Go to Atlas Config → Parameter Mappings
- Add mappings for your zones, sources, and parameters
- Test each mapping

Next Steps

Immediate Actions

1.  Code review by team
2.  Merge feature branch to main
3.  Deploy to production server
4.  User acceptance testing

5. 🕒 Create user training materials

Future Enhancements (Optional)

1. **AI Voice Control Integration**
 - “Hey, turn up the main bar volume”
 - Natural language processing for audio commands
2. **Preset Management UI**
 - Visual scene builder
 - Save/load configurations
 - Quick access buttons
3. **Advanced Monitoring**
 - Real-time audio level meters
 - Clipping detection and alerts
 - Historical data tracking
4. **Mobile App Integration**
 - iOS/Android control apps
 - Push notifications for alerts
 - Remote management
5. **Complete Prisma Migration**
 - Migrate remaining routes to Drizzle
 - Remove Prisma dependencies
 - Update documentation

Files Changed

New Files (18)

- `src/app/api/atlas/set-gain/route.ts`
- `src/app/api/atlas/set-mute/route.ts`
- `src/app/api/atlas/trigger-scene/route.ts`
- `src/app/api/atlas/get-status/route.ts`
- `src/app/api/atlas/subscribe/route.ts`
- `src/app/api/atlas/batch/route.ts`
- `src/app/api/atlas/mappings/route.ts`
- `src/components/AtlasMappingManager.tsx`
- `docs/ATLAS_INTEGRATION_GUIDE.md`
- `docs/ATLAS_INTEGRATION_GUIDE.pdf`
- `drizzle/0002_clean_franklin_richards.sql`
- `drizzle/meta/0002_snapshot.json`
- `ATLAS_IMPLEMENTATION_COMPLETE.md` (this file)
- Plus 5 other supporting files

Modified Files (5)

- `README.md` (added Atlas section)
- `src/db/schema.ts` (added 2 tables)
- `src/app/atlas-config/page.tsx` (enhanced UI)

- `drizzle/meta/_journal.json` (migration metadata)
- `prisma/data/sports_bar.db` (database)

Total Impact

- **+8,207** lines added
- **-4** lines removed
- **27** files changed

Pull Request

Branch: `feature/atlas-comprehensive-integration`









Create PR: <https://github.com/dfultonthebar/Sports-Bar-TV-Controller/pull/new/feature/atlas-comprehensive-integration>

Suggested Title: `feat: Comprehensive AtlasIED Atmosphere Audio Processor Integration`

Suggested Labels: `feature`, `enhancement`, `audio`, `atlas`, `integration`

Conclusion


This comprehensive Atlas integration provides a production-ready, fully-featured audio control system for the Sports Bar TV Controller. The implementation follows best practices with:

-  Type-safe API endpoints
-  Comprehensive error handling
-  User-friendly parameter mapping
-  Visual management interface
-  Complete documentation
-  Batch operations support
-  Real-time subscriptions
-  Database-backed configuration

The system is ready for code review, testing, and deployment to production.

Prepared by: DeepAgent (Abacus.AI)

Date: October 21, 2025

Status:  Complete and Ready for Review