Zone Output Groups Architecture Implementation

Overview

This implementation adds support for Atlas audio zones with multiple amplifier outputs (e.g., Mono+Sub, Stereo, etc.), where each output has independent volume control but shares the same source selection.

Use Case

The **Main Bar** zone is configured as “Mono + Sub” with two amplifier outputs:
- **Output 1: Main** (Mono Amp Out 1) - Primary speakers
- **Output 2: Sub** (Amp Out 2) - Subwoofer

Both outputs play the same source (e.g., Spotify, Mic 1, etc.), but the subwoofer volume can be adjusted independently from the main speakers.

Architecture Design

1. Data Model Changes

New Interface: `AtlasZoneOutput`

```
interface AtlasZoneOutput {
  index: number           // Output index within zone (0-based)
  name: string           // e.g., "Main", "Sub", "Left", "Right"
  type: string           // e.g., "main", "sub", "left", "right", "mono"
  volume?: number        // 0-100 percentage
  parameterName?: string // Atlas parameter (e.g., "ZoneOutput1Gain_0")
}
```

Updated Interface: `AtlasHardwareZone`

```
interface AtlasHardwareZone {
  // ... existing fields ...
  outputs?: AtlasZoneOutput[] // Array of amplifier outputs
}
```

2. Backend Implementation

Atlas Hardware Query (`atlas-hardware-query.ts`)

New Function: `queryZoneOutputs()`
- Probes Atlas processor to detect multiple outputs per zone
- Strategy 1: Query output count parameters
- Strategy 2: Probe individual output gain parameters
- Strategy 3: Fallback to single output using ZoneGain
- Returns array of `AtlasZoneOutput` objects

Parameter Patterns Attempted:

```
// Output count detection
ZoneOutputCount_${zoneIndex}
ZoneChannels_${zoneIndex}
ZoneAmpCount_${zoneIndex}
NumOutputs_${zoneIndex}

// Individual output gains
ZoneOutput${outIdx + 1}Gain_${zoneIndex}
AmpOutGain_${zoneIndex}_${outIdx}
ZoneAmp${outIdx}Gain_${zoneIndex}
Output${outIdx + 1}Gain_${zoneIndex}
```

Zone Status API (`zones-status/route.ts`)

Returns zone data with outputs:

```json
{
  "zones": [
    {
      "id": "zone_0",
      "name": "Main Bar",
      "currentSource": 4,
      "currentSourceName": "Spotify",
      "volume": 75,
      "outputs": [
        {
          "id": "output_0_0",
          "name": "Main",
          "type": "main",
          "volume": 80,
          "parameterName": "ZoneOutput1Gain_0"
        },
        {
          "id": "output_0_1",
          "name": "Sub",
          "type": "sub",
          "volume": 60,
          "parameterName": "ZoneOutput2Gain_0"
        }
      ]
    }
  ]
}
```

Control API (`control/route.ts`)

New Action: `output-volume`

Command format:

```json
{
  "action": "output-volume",
  "zone": 1,              // Zone number (1-based)
  "outputIndex": 1,       // Output index (0-based)
  "value": 60,           // Volume percentage (0-100)
  "parameterName": "ZoneOutput2Gain_0"
}
```

Function: `setZoneOutputVolume()`
- Accepts zone, outputIndex, volume, and optional parameterName
- Sends Atlas command to adjust specific output gain
- Maintains state consistency

3. Frontend Implementation

UI Component (`AudioZoneControl.tsx`)

New State:

```
const [expandedZones, setExpandedZones] = useState<Set<string>>(new Set())
```

New Handler:

```
const handleOutputVolumeChange = async (
  zoneId: string,
  outputId: string,
  newVolume: number
) => {
  // Update local state
  // Send API request to backend
}
```

Rendering Logic:

Multi-Output Zone (≥2 outputs):
- Show “Output Controls (N outputs)” header
- Expand/Collapse button
- Collapsed view: Summary of all output volumes
- Expanded view: Individual volume sliders for each output

Single-Output Zone (1 output):
- Show traditional single “Volume” control
- Backward compatible with existing zones

4. Backward Compatibility

The implementation maintains full backward compatibility:

1. **Zones without output configuration**: Automatically get single “Main” output
2. **Existing volume API**: Still works for single-output zones
3. **Database**: No schema changes required
4. **Legacy code**: All existing zone control code continues to work

5. Testing Strategy

Unit Testing

- Test `queryZoneOutputs()` with various parameter patterns
- Test `extractValueFromResponse()` with different Atlas response formats
- Test zone status API output formatting

Integration Testing

1. **Single-output zone**: Verify traditional volume control works

2. **Multi-output zone**:
   - Verify output detection
   - Verify independent volume control
   - Verify source applies to all outputs
3. **Main Bar zone**:
   - Verify Mono+Sub detection
   - Adjust Main and Sub independently
   - Verify mute affects both outputs

Manual Testing

```
# Start the development server
npm run dev

# Navigate to Audio Control Center
# Verify Main Bar shows 2 outputs
# Test expanding/collapsing output controls
# Test adjusting Main and Sub volumes independently
# Test source selection applies to both outputs
```

6. Atlas Processor Configuration

The Atlas AZMP8 processor must be configured with:
- Zone configured as “Mono + Sub” or “Stereo”
- Multiple amplifier outputs assigned to the zone
- Each output should have distinct gain parameters

Verify in Atlas web interface:
1. Navigate to zone configuration
2. Check “Output Type” (should show “Mono + Sub”, “Stereo”, etc.)
3. Verify amplifier outputs are listed
4. Confirm each output has gain control

7. API Reference

GET `/api/audio-processor/[id]/zones-status`

Response:

```
{
  "success": true,
  "zones": [
    {
      "id": "zone_0",
      "name": "Main Bar",
      "outputs": [
        { "id": "output_0_0", "name": "Main", "volume": 80 },
        { "id": "output_0_1", "name": "Sub", "volume": 60 }
      ]
    }
  ]
}
```

POST `/api/audio-processor/control`

Set zone-level volume (backward compatible):

```
{
  "processorId": "proc-id",
  "command": {
    "action": "volume",
    "zone": 1,
    "value": 75
  }
}
```

Set output-specific volume (new):

```
{
  "processorId": "proc-id",
  "command": {
    "action": "output-volume",
    "zone": 1,
    "outputIndex": 1,
    "value": 60,
    "parameterName": "ZoneOutput2Gain_0"
  }
}
```

8. Future Enhancements

1. **Output Presets**: Save/recall output volume configurations
2. **Output Muting**: Per-output mute controls
3. **Output EQ**: Per-output equalization
4. **Stereo Balance**: Automatic left/right balance control
5. **Subwoofer Modes**: Quick presets (Off, Low, Normal, High)
6. **Group Outputs**: Link multiple outputs for synchronized control

9. Troubleshooting

Problem: Zone shows only one output (Main) despite Mono+Sub configuration
- Verify Atlas processor zone configuration shows multiple outputs
- Check Atlas web interface for output parameter names
- Review `queryZoneOutputs()` console logs for detection attempts
- Manually test parameter patterns via Atlas TCP connection

Problem: Volume control doesn’t adjust output
- Verify parameterName is correct in API request
- Check Atlas processor logs for command reception
- Test parameter directly using Atlas web interface or TCP client
- Confirm output is not muted in Atlas configuration

Problem: Outputs detected but volumes don’t match Atlas
- Check Atlas parameter format (pct vs val vs dB)
- Verify `extractValueFromResponse()` handles response correctly
- Compare raw Atlas response with parsed values

10. Documentation References

- **Atlas Protocol Spec**: `ATS006993-B-AZM4-AZM8-3rd-Party-Control.pdf`
- **Parameter Reference**: Atlas Third Party Control Message Table (web UI)

- **Zone Configuration**: Atlas Zones tab in web interface
- **TCP Protocol**: Port 5321 (JSON-RPC 2.0)
- **HTTP Configuration**: Port 80 (web interface)

Implementation Status

✅ Data model updated with zone outputs
✅ Atlas hardware query detects multiple outputs
✅ Zone status API returns output details
✅ Control API supports per-output volume
✅ UI shows expandable/collapsible output controls
✅ Backward compatibility maintained
✅ Committed to feature branch
⏳ Pending: PR creation and testing on actual hardware

Notes for Deployment

1. **No breaking changes**: Existing zones continue to work
2. **No migration needed**: Database schema unchanged
3. **Gradual rollout**: Multi-output zones work when configured
4. **Testing priority**: Main Bar zone (Mono+Sub)
5. **Monitoring**: Watch for parameter detection logs