Audio Control React Error #31 - Fix Summary

Date: October 23, 2025

Problem

The Audio Control section was displaying React error #31: “Objects are not valid as a React child (found: object with keys {param, str})”. Zone information from the Atlas controller was not rendering properly, showing “Something went wrong!” error.

Root Cause Analysis

The Atlas hardware returns parameter values wrapped in objects/arrays with metadata instead of primitive values:

Example of problematic data structure:

```
{
  "name": [{"param": "ZoneName_0", "str": "Main Bar"}],
  "volume": [{"param": "ZoneGain_0", "pct": 100}],
  "currentSource": [{"param": "ZoneSource_0", "val": -1}]
}
```

Expected data structure:

```
{
  "name": "Main Bar",
  "volume": 100,
  "currentSource": -1
}
```

The code was passing these complex structures directly to React components, causing React to attempt rendering objects as children, which triggered error #31.

Solution Implemented

1. Added Helper Functions in `atlas-http-client.ts`

Created two helper functions to extract primitive values from Atlas parameter responses:

```
private extractStringValue(value: any, defaultValue: string): string {
  if (typeof value === 'string') return value
  if (Array.isArray(value) && value.length > 0) {
    const first = value[0]
    if (typeof first === 'object' && first !== null) {
      return first.str || first.val || defaultValue
    }
  }
  if (typeof value === 'object' && value !== null) {
    return value.str || value.val || defaultValue
  }
  return defaultValue
}

private extractNumericValue(value: any, defaultValue: number): number {
  if (typeof value === 'number') return value
  if (Array.isArray(value) && value.length > 0) {
    const first = value[0]
    if (typeof first === 'object' && first !== null) {
      return first.pct ?? first.val ?? defaultValue
    }
  }
  if (typeof value === 'object' && value !== null) {
    return value.pct ?? value.val ?? defaultValue
  }
  return defaultValue
}
```

2. Updated `parseJsonMessageTable()` Method

Modified the method to use helper functions when parsing source and zone names:

```
// Before
name: source.name || `Source ${index + 1}`

// After
name: this.extractStringValue(source.name, `Source ${index + 1}`)
```

3. Updated `zones-status` API Route

Added the same helper functions and applied them when formatting zone data:

```
const zoneName = extractStringValue(zone.name, `Zone ${zone.index + 1}`)
const currentSource = extractNumericValue(zone.currentSource, -1)
const volume = extractNumericValue(zone.volume, 50)
```

Files Modified

1. `src/lib/atlas-http-client.ts` - Added helper functions and updated parseJsonMessageTable
2. `src/app/api/audio-processor/[id]/zones-status/route.ts` - Added helper functions and updated response formatting

Testing Results

✅ Zone names display correctly (e.g., “Main Bar”, “Dining Room”, “Party Room West”, “Party Room East”, “Patio”)
✅ Volume levels show as numbers (e.g., 100%, 65%, 70%, 50%)
✅ Current source assignments display properly (“Not Set”)
✅ No React error #31 in console
✅ Audio Control section loads without errors
✅ All 8 zones from Atlas controller render correctly

Deployment

- **Branch**: `fix/audio-zone-render`
- **Pull Request**: #234
- **Commit**: c38cb409e29544563acf6d3c199ac398582870f1
- **Deployed to**: Production server (24.123.87.42:3000)
- **Status**: ✅ Successfully deployed and verified

Console Output (Success)

```
Real Atlas configuration loaded from hardware: {
  processor: 'Main Bar',
  model: 'AZMP8',
  sources: 9,
  matrixInputs: 4,
  zones: 8,
  queriedAt: '2025-10-23T01:05:10.441Z'
}
```

Impact

- Fixed critical bug preventing Audio Control section from loading
- Zone information now properly displays data from Atlas hardware
- Improved user experience for audio zone management
- Resolved React rendering error that was blocking the entire Audio Control interface

Technical Notes

The Atlas AZMP8 audio processor returns parameter values in a JSON-RPC-like format with metadata. This is by design for the Atlas protocol, but our React components expect primitive values. The helper functions bridge this gap by extracting the actual values while maintaining backward compatibility with both formats (direct primitives and wrapped objects).