AI Diagnostic Quick Reference

For AI Assistant Use

This is a quick reference guide for the AI assistant to use when running diagnostics on the Sports Bar TV Controller system.

Common User Requests and Responses

“Check system health” / “Run diagnostics”

Action: Run comprehensive diagnostics

```
POST /api/ai/run-diagnostics
Body: { "checks": ["all"], "detailed": true }
```

Response Template:
“I’ll run a comprehensive system diagnostic for you. Let me check all components...”

[After receiving results]
“System Health Report:
- Overall Health: [healthy/warning/critical]
- Health Score: [X]%
- Passed: [X] checks
- Warnings: [X] checks
- Failed: [X] checks

[List any issues found with details]
[Provide recommendations based on results]”

“Are the AI providers working?”

Action: Check AI provider status

```
GET /api/ai-providers/status
```

Response Template:
“Let me check the AI provider status...”

[After receiving results]
“AI Provider Status:
- Active Providers: [X] of [Y]
- Health: [healthy/warning/error]
- Health Score: [X]%

[List each provider with status]
[Provide recommendations if issues found]”

“Check device mapping” / “Are devices configured correctly?”

Action: Check device mapping

```
GET /api/diagnostics/device-mapping
```

Response Template:
“I’ll analyze the device mapping configuration...”

[After receiving results]
“Device Mapping Analysis:
- Total Matrix Inputs: [X]
- Mapped Inputs: [X]
- Unmapped Inputs: [X]
- Mapping Percentage: [X]%

[List any unmapped devices]
[Provide configuration recommendations]”

“Something is wrong with [device/system]”

Action: Run targeted diagnostics

```
POST /api/ai/run-diagnostics
Body: {
  "checks": ["matrix_inputs", "ir_devices", "device_mapping"],
  "detailed": true
}
```

Response Template:
“Let me run diagnostics on the device system to identify the issue...”

[After receiving results]
“I found the following:
[List relevant findings]
[Provide troubleshooting steps]
[Offer to execute automated fixes if available]”

“Fix [specific issue]”

Action: Execute automated fix

```
POST /api/devices/execute-fix
Body: { "actionId": "[action_id]", "deviceId": "[device_id]" }
```

Response Template:
“I can attempt an automated fix for this issue. Let me execute the repair...”

[After receiving results]
“Fix Execution Results:
- Action: [action_name]
- Status: [success/failed]
- Result: [improvement/reason]

[Provide next steps or additional recommendations]”

Diagnostic Check Types

Quick Checks (Fast, Basic Info)

```
GET /api/ai/run-diagnostics?quick=true
```

Use when: User wants a quick status update

Specific Checks (Targeted)

```
POST /api/ai/run-diagnostics
Body: { "checks": ["database", "ai_providers"] }
```

Use when: User reports a specific issue

Comprehensive Checks (Detailed)

```
POST /api/ai/run-diagnostics
Body: { "checks": ["all"], "detailed": true }
```

Use when: User wants full system analysis or troubleshooting

Interpreting Results

Health Scores

- **90-100%**: Excellent - System is healthy
- **70-89%**: Good - Minor issues, monitor
- **50-69%**: Fair - Several issues, attention needed
- **Below 50%**: Poor - Critical issues, immediate action required

Status Levels

- **pass**: Component is functioning correctly
- **warning**: Component has minor issues or needs attention
- **fail**: Component has critical issues requiring immediate action

Categories

- **infrastructure**: Database, API endpoints, network
- **ai_system**: AI providers, knowledge base, models
- **hardware**: Matrix inputs, IR devices, physical equipment
- **configuration**: Device mapping, system settings
- **monitoring**: Logs, error rates, performance metrics

Proactive Monitoring

When to Run Diagnostics Automatically

1. **User mentions issues**: Always run relevant diagnostics

2. **After system changes**: Verify changes didn't break anything
3. **Periodic checks**: Suggest running diagnostics if not done recently
4. **Before major events**: Recommend pre-event health check
5. **After errors**: Run diagnostics to identify root cause

Best Practices

1. **Start Broad, Then Narrow**: Begin with comprehensive checks, then focus on problem areas
2. **Always Use Detailed Mode**: When troubleshooting, request detailed information
3. **Explain Results**: Translate technical results into user-friendly language
4. **Provide Context**: Explain what each check means and why it matters
5. **Offer Solutions**: Always provide actionable recommendations
6. **Follow Up**: After fixes, re-run diagnostics to verify success

Example Diagnostic Workflow

User: "The system seems slow"

1. **Run comprehensive diagnostics**
   ```
   javascript
   POST /api/ai/run-diagnostics
   Body: { "checks": ["all"], "detailed": true }
   ```

2. **Analyze results** - Look for:
   - High error rates in system logs
   - API endpoint response times
   - Device connection issues
   - Database performance

3. **Check specific components** based on findings:
   ```
   javascript
   GET /api/devices/ai-analysis
   ```

4. **Provide recommendations**:
   - Specific issues identified
   - Suggested fixes
   - Preventive measures

5. **Execute fixes if available**:
   ```
   javascript
   POST /api/devices/execute-fix
   Body: { "actionId": "...", "deviceId": "..." }
   ```

6. **Verify fix**:
   ```
   javascript
   POST /api/ai/run-diagnostics
   Body: { "checks": ["relevant_checks"], "detailed": true }
   ```

Error Handling

API Call Fails

- Inform user that diagnostic system is unavailable

- Suggest manual checks
- Recommend contacting system administrator

Partial Results

- Report what was successfully checked
- Note which checks failed
- Provide recommendations based on available data

No Issues Found

- Confirm system appears healthy
- Suggest monitoring for patterns
- Offer to check specific components if user has concerns

Integration with Other Features

With AI Teaching Interface

- After user adds documentation, suggest running knowledge base check
- Verify knowledge base is properly indexed

With Device Configuration

- After device changes, run device mapping diagnostics
- Verify new devices are properly configured

With System Admin

- Cross-reference diagnostic results with system logs
- Provide links to relevant log entries

With Intelligent Troubleshooter

- Use diagnostic results to inform troubleshooting recommendations
- Suggest automated fixes based on diagnostic findings

Response Templates

Healthy System

“Great news! I’ve completed a comprehensive system diagnostic and everything looks healthy:
✅ All [X] checks passed
✅ Health Score: [X]%
✅ No critical issues detected

Your Sports Bar TV Controller system is operating optimally.”

System with Warnings

“I’ve completed the diagnostic scan. The system is functional but has some items that need attention:
⚠️ Health Score: [X]%
✅ [X] checks passed
⚠️ [X] warnings detected

Issues found:
[List warnings with details]

Recommendations:
[Provide specific actions to address warnings]”

System with Errors

“I’ve identified some critical issues that need immediate attention:
❌ Health Score: [X]%
❌ [X] critical issues detected
⚠️ [X] warnings

Critical Issues:
[List errors with details]

I can help fix some of these automatically. Would you like me to:
1. [Automated fix option 1]
2. [Automated fix option 2]
Or I can provide manual troubleshooting steps.”

Remember

- **Be Proactive**: Offer to run diagnostics when appropriate
- **Be Clear**: Explain technical results in simple terms
- **Be Helpful**: Always provide actionable next steps
- **Be Thorough**: Use detailed mode when troubleshooting
- **Be Efficient**: Use quick checks for simple status updates
- **Be Accurate**: Base recommendations on actual diagnostic results
- **Be Supportive**: Guide users through fixes step-by-step

This diagnostic system is a powerful tool for maintaining system health and providing excellent user support. Use it frequently and effectively!