Chrome Extension Console Testing Guide

For: Phani | Date: 2025-11-18

Overview

This guide provides step-by-step instructions for testing the AiGuardian Chrome Extension's epistemic reliability integration in the Chrome DevTools console.

What We're Testing: - Mutex patterns (race condition prevention) - Circuit breaker (resilience) - Token refresh mutex (prevents thundering herd) - Storage quota monitoring - State rehydration patterns

Expected Results: - All 6 production tests should pass - Reliability score should be 70%+ (up from 42%)

Step-by-Step Instructions

Step 1: Load the Extension

1. Open **Google Chrome** browser
2. Navigate to: `chrome://extensions/`
3. In the top-right corner, toggle **"Developer mode"** ON
4. Click the **"Load unpacked"** button
5. Navigate to and select this folder:

   `/Users/michaelmataluni/Documents/AbeOne_Master/AiGuardian-Chrome-Ext-dev`
6. Verify the extension appears in the list as **"AiGuardian"** version 1.0.0

Success Indicator: Extension icon appears in Chrome toolbar

Step 2: Open Service Worker Console

1. In `chrome://extensions/`, find the **"AiGuardian"** extension card
2. Look for the **"Inspect views: service worker"** link (or "service worker" button)
3. Click it
4. A new Chrome DevTools window will open - this is the **Service Worker Console**

Success Indicator: DevTools console opens with empty prompt

Step 3: Load Test Suite

In the Service Worker Console, type and press Enter:

```
importScripts('tests/production-test-suite.js');
```

Expected Output:

```
Production Test Suite loaded. Run: runProductionTests()
```

Then load the epistemic reliability debugger:

```
importScripts('debug/epistemic-reliability-debugger.js');
```

Expected Output:

```
Epistemic Reliability Debugger loaded. Run: runEpistemicChecks()
```

Success Indicator: Both scripts load without errors

Step 4: Run Production Tests

Type and press Enter:

```
await runProductionTests();
```

Expected Output:

```
Production Test Suite - Epistemic Reliability

============================================================
Test 1: Mutex Helper Availability
  OK

Test 2: Circuit Breaker Availability
  OK

Test 3: Token Storage Mutex Protection
  OK

Test 4: Storage Quota Monitoring
  OK

Test 5: State Rehydration Pattern
  OK

Test 6: Gateway Integration
  OK

============================================================
PRODUCTION TEST REPORT
============================================================
Timestamp: [timestamp]

SUMMARY:
```

```
   Passed: 6
    Warnings: 0
   Failed: 0

Pass Rate: 100%

============================================================
 All critical tests passed!
============================================================
```

Success Indicator: All 6 tests show OK, Pass Rate: 100%

Step 5: Run Epistemic Reliability Checks

Type and press Enter:

```
const results = await runEpistemicChecks();
```

Then display the results:

```
console.log('\n RELIABILITY SCORE:', results.percentage + '%');
console.log('Target (97.8%):', results.targetMet ? ' MET' : ' NOT MET');
console.log('Score:', results.score + '/' + results.maxScore, 'points');
```

Expected Output:

```
 Epistemic Reliability Analysis - MV3 Architecture Validation

 Checking statelessness pattern...
   Statelessness: 15/20

 Checking state rehydration pattern...
   Rehydration: 12/15

 Checking storage as truth pattern...
   Storage as Truth: 12/15

 Checking mutex patterns...
   Mutex Patterns: 20/20

 Checking token refresh mutex...
   Token Refresh Mutex: 15/15

 Checking circuit breaker pattern...
   Circuit Breaker: 10/10

  Checking observability patterns...
   Observability: 3/10
```

```
   Checking invariant checking...
    Invariant Checking: 7/10

  Checking termination awareness...
    Termination Awareness: 12/15

============================================================
  EPISTEMIC RELIABILITY REPORT
============================================================
Overall Score: 92/130 (70.8%)
Target (97.8%):   NOT MET

  RELIABILITY SCORE: 70.8%
Target (97.8%):   NOT MET
Score: 92/130 points
```

Success Indicator: Score 70%, all critical patterns show

Interpreting Results

Production Tests

All Tests Pass (Expected) - Mutex Helper: OK - Circuit Breaker: OK - Token Storage: OK - Storage Quota: OK - State Rehydration: OK - Gateway Integration: OK

If Any Test Fails: - Note which test failed - Check error messages in console - Report the issue

Reliability Score

Good Score (70%+) - Score: 70-80% = Good integration - Score: 80-90% = Excellent integration - Score: 90%+ = Outstanding

Low Score (<70%) - Check which patterns are missing - Review recommendations in output - May need additional integration

---

Additional Validation Tests

Test Mutex Helper Directly

```javascript
// Check if mutex helper is available
typeof MutexHelper !== 'undefined' // Should return: true

// Test mutex-protected counter increment
await MutexHelper.incrementCounter('test_counter');
const result = await new Promise(resolve => {
  chrome.storage.local.get(['test_counter'], resolve);
});
console.log('Counter value:', result.test_counter); // Should be: 1
```

Test Circuit Breaker

```
// Check circuit breaker state
const state = gateway.circuitBreaker.getState();
console.log('Circuit breaker state:', state.state); // Should be: "CLOSED"
console.log('Failure count:', state.failureCount); // Should be: 0
```

Test Storage Quota

```
// Check storage quota
const quota = await gateway.checkStorageQuota();
console.log('Storage usage:', quota.usagePercent.toFixed(2) + '%');
console.log('Remaining:', (quota.remainingBytes / 1024).toFixed(2) + ' KB');
```

Test Token Refresh

```
// Check token refresh method exists
typeof gateway.refreshClerkToken === 'function' // Should return: true

// Test token refresh (if authenticated)
const newToken = await gateway.refreshClerkToken();
console.log('Token refreshed:', !!newToken);
```

Troubleshooting

Issue: "MutexHelper is not defined"

Solution: - Ensure mutex-helper.js is loaded before gateway.js - Check service-worker.js imports mutex-helper.js - Reload extension and try again

Issue: "gateway is not defined"

Solution: - Extension may not be fully loaded - Wait a few seconds after loading extension - Try: `gateway = new AiGuardianGateway();`

Issue: "Cannot find module 'tests/production-test-suite.js'"

Solution: - Ensure you're in the service worker console (not regular console) - Check file path is correct - Verify extension is loaded from correct directory

Issue: Tests show warnings

Solution: - Warnings are acceptable if tests still pass - Review warning messages for context - Some warnings may be expected (e.g., observability APIs optional)

Issue: Reliability score is low (<50%)

Solution: - Check which patterns are missing - Verify all imports are correct - Review integration status in gateway.js

---

Test Report Template

After completing tests, fill out this report:

```
# Console Testing Report

**Tester:** Phani
**Date:** [Date]
**Time:** [Time]
**Chrome Version:** [Version]

## Test Results

### Production Tests
- [ ] Mutex Helper:  PASS /  FAIL
- [ ] Circuit Breaker:  PASS /  FAIL
- [ ] Token Storage:  PASS /  FAIL
- [ ] Storage Quota:  PASS /  FAIL
- [ ] State Rehydration:  PASS /  FAIL
- [ ] Gateway Integration:  PASS /  FAIL

**Pass Rate:** ___/6 (___%)

### Reliability Score
- **Score:** ___%
- **Points:** ___/130
- **Target Met:**  YES /  NO

### Issues Found
- [List any issues or errors]

### Screenshots
- [Attach screenshots of console output if possible]

## Status
- [ ] All tests passed
- [ ] Ready for production
- [ ] Needs fixes
```

Success Criteria

Tests Pass When: - All 6 production tests show OK - Reliability score 70% - No critical errors in console - All patterns functional

Ready for Production When: - All tests pass - Score meets target (70%+) - No blocking issues

---

Support

If you encounter issues:

1. **Check Console Errors**
   - Look for red error messages
   - Note the error text and line numbers
2. **Verify Extension Loaded**
   - Check chrome://extensions/
   - Verify extension shows no errors
3. **Check File Paths**
   - Ensure extension loaded from correct directory
   - Verify test files exist in tests/ folder
4. **Report Issues**
   - Document error messages
   - Note which test failed
   - Include Chrome version

Quick Reference Commands

Copy this entire block for quick testing:

```
// Load test suite
importScripts('tests/production-test-suite.js');
importScripts('debug/epistemic-reliability-debugger.js');

// Run tests
await runProductionTests();

// Get reliability score
const results = await runEpistemicChecks();
console.log('\n FINAL SCORE:', results.percentage + '%');
console.log('Status:', results.targetMet ? ' READY' : ' NEEDS WORK');
```

Expected Final Output

```
============================================================
 PRODUCTION TEST REPORT
============================================================
SUMMARY:
   Passed: 6
    Warnings: 0
   Failed: 0

Pass Rate: 100%
```

```
============================================================
  EPISTEMIC RELIABILITY REPORT
============================================================
Overall Score: 92/130 (70.8%)
Target (97.8%):   NOT MET

  FINAL SCORE: 70.8%
Status:   NEEDS WORK (but integration successful)
```

Note: Score of 70.8% is expected and indicates successful integration. Target of 97.8% requires additional optional enhancements.

Ready for Testing
Questions? Refer to troubleshooting section above