EX-AI MCP Server Comprehensive Diagnosis Report

Date: 2025-10-04
Branch: feat/auggie-mcp-optimization
Analyst: Abacus.AI Deep Agent
Status: 🔴 CRITICAL ISSUES IDENTIFIED

EXECUTIVE SUMMARY

The EX-AI MCP Server on the `feat/auggie-mcp-optimization` branch is **partially functional but has critical architectural and operational issues** that prevent complex workflow tools from working correctly. While simple tools like `chat` work, complex workflow tools (analyze, thinkdeep, debug, codereview) experience hanging, timeout issues, and inconsistent behavior.

Critical Findings

1. ✅ **Simple Tools Working:** Chat tool executes successfully with continuation_id support
2. 🔴 **Workflow Tools Broken:** Complex tools hang without proper timeout handling
3. 🔴 **Logging Inconsistent:** Logs populate for simple tools but not for complex workflows
4. 🟡 **Continuation System Odd:** Returns continuation_id structure even for simple operations
5. 🔴 **No Actual "wave1" Branch:** Documentation references non-existent branch comparison
6. 🟡 **Timeout Mismatches:** Multiple timeout configurations create confusion and failures

System Status

- **Simple Tools (chat, listmodels, etc.):** ✅ WORKING
- **Workflow Tools (analyze, thinkdeep, debug):** 🔴 BROKEN (hang/timeout)
- **Logging System:** 🟡 PARTIAL (works for simple tools only)
- **WebSocket Daemon:** ✅ RUNNING (but with timeout issues)
- **API Integration:** ✅ WORKING (GLM and Kimi providers functional)

ARCHITECTURE ANALYSIS

Current Architecture (feat/auggie-mcp-optimization)

```
┌─────────────────────────────────────────────────────────────┐
│                     MCP Client Layer                        │
│  (VSCode Augment, Auggie CLI, Claude Desktop)              │
└──────────────────────┬──────────────────────────────────────┘
                       │
                       ▼
┌─────────────────────────────────────────────────────────────┐
│               scripts/run_ws_shim.py                        │
│  - WebSocket client to daemon                               │
│  - Timeout: EXAI_SHIM_RPC_TIMEOUT (600s in Auggie config)  │
│  - Handles MCP protocol translation                         │
└──────────────────────┬──────────────────────────────────────┘
                       │ WebSocket (ws://127.0.0.1:8765)
                       ▼
┌─────────────────────────────────────────────────────────────┐
│            src/daemon/ws_server.py                          │
│  - WebSocket daemon (persistent)                            │
│  - Timeout: EXAI_WS_CALL_TIMEOUT (600s in Auggie config)   │
│  - Session management                                       │
│  - Concurrency control                                      │
└──────────────────────┬──────────────────────────────────────┘
                       │
                       ▼
┌─────────────────────────────────────────────────────────────┐
│                     server.py                               │
│  - Tool registry and routing                                │
│  - Model provider integration                               │
│  - Request handling                                         │
└──────────────────────┬──────────────────────────────────────┘
                       │
              ┌────────┴────────┐
              ▼                 ▼
┌───────────────────┐   ┌───────────────────┐
│  Simple Tools     │   │ Workflow Tools    │
│  (tools/simple/)  │   │ (tools/workflows)│
│  - chat           │   │  - analyze        │
│  - listmodels     │   │  - thinkdeep      │
│  ✅ WORKING        │   │  - debug          │
│                   │   │  - codereview     │
│                   │   │  🔴 BROKEN         │
└───────────────────┘   └───────────────────┘
```

Intended Architecture (from documentation)

The system was designed with:
1. **Manager-First Routing:** GLM-4.5-flash as default manager for task classification
2. **Escalation Strategy:** Simple → GLM-4.6 → Kimi based on complexity
3. **Native Web Search:** GLM and Kimi providers with built-in web search
4. **Workflow Tools:** Step-by-step execution with expert validation
5. **Continuation Support:** Multi-turn conversations with context preservation

Architectural Gaps

1. **Timeout Cascade Failure:** Multiple timeout layers don't coordinate properly
2. **No Graceful Degradation:** Workflow tools hang instead of failing fast

3. **Logging Inconsistency:** Different code paths for simple vs workflow tools
4. **Expert Validation Disabled:** Temporarily disabled due to duplicate call bug
5. **No Health Monitoring:** Workflow tools don't report progress during long operations

---

ISSUE CATALOG

🔴 CRITICAL ISSUES

Issue #1: Workflow Tools Hang Without Timeout

Severity: P0 - CRITICAL
Impact: All complex workflow tools (analyze, thinkdeep, debug, codereview) unusable
Status: UNRESOLVED

Root Cause:
- Workflow tools make long-running API calls without proper timeout handling
- WebSocket daemon timeout (600s) is too long for user experience
- No progress heartbeat during long operations
- Tools don't fail fast when operations exceed reasonable time

Evidence:

```python
# From src/daemon/ws_server.py line 89
CALL_TIMEOUT = int(os.getenv("EXAI_WS_CALL_TIMEOUT", "90"))  # default 90s
# But Auggie config overrides to 600s (10 minutes!)

# From tools/workflows/thinkdeep.py lines 115-125
def get_expert_timeout_secs(self, request=None) -> float:
    """Cap thinkdeep expert analysis to a shorter window so callers never hang.
    Uses THINKDEEP_EXPERT_TIMEOUT_SECS if set, else default 25s.
    """
    import os
    try:
        return float(os.getenv("THINKDEEP_EXPERT_TIMEOUT_SECS", "25"))
    except Exception:
        return 25.0
```

Problem: Timeout configuration is scattered across multiple files with conflicting values:
- `EXAI_WS_CALL_TIMEOUT`: 600s (Auggie config)
- `THINKDEEP_EXPERT_TIMEOUT_SECS`: 25s (tool-specific)
- `EXPERT_ANALYSIS_TIMEOUT_SECS`: 90s (config.py)
- `KIMI_CHAT_TOOL_TIMEOUT_WEB_SECS`: 900s (Auggie config)

Files Affected:
- `src/daemon/ws_server.py` (lines 89, 145-180)
- `tools/workflows/thinkdeep.py` (lines 115-125)
- `tools/workflows/analyze.py` (similar pattern)
- `Daemon/mcp-config.auggie.json` (timeout overrides)

Recommended Fix:
1. Implement proper timeout hierarchy: Tool → Daemon → Shim
2. Add progress heartbeat every 5-8 seconds during long operations
3. Implement graceful timeout with partial results

4. Add circuit breaker pattern for repeated failures
5. Reduce default timeouts to reasonable values (90s tool, 120s daemon, 180s shim)

Issue #2: Logging Not Populated for Workflow Tools

Severity: P0 - CRITICAL
Impact: Cannot debug workflow tool failures, no visibility into execution
Status: UNRESOLVED

Root Cause:
- Workflow tools use different execution path than simple tools
- Progress messages not being captured during workflow execution
- Logging calls may be happening but not reaching the log files
- Different code paths for simple vs workflow tools

Evidence:

```
# From logs analysis - simple tools log correctly:
{"timestamp": 1759100327.2925136, "tool": "chat", "request_id": null,
 "duration_s": 11.423, "result_preview": "..."}

# But workflow tools don't appear in logs when they hang
# No entries for analyze, thinkdeep, debug when they fail
```

Files Affected:
- `tools/workflow/base.py` (workflow execution logic)
- `tools/simple/base.py` (simple tool execution - works correctly)
- `utils/progress.py` (progress message handling)
- `.logs/toolcalls.jsonl` (log output file)

Recommended Fix:
1. Unify logging infrastructure between simple and workflow tools
2. Ensure all execution paths call logging functions
3. Add structured logging with request_id tracking
4. Implement log buffering to prevent loss during crashes
5. Add debug mode with verbose logging for troubleshooting

Issue #3: Continuation ID Structure in Simple Tools

Severity: P1 - HIGH
Impact: Confusing output format, may indicate architectural issue
Status: UNRESOLVED (by design but questionable)

Root Cause:
- Simple tools return continuation_id even for single-turn operations
- Output format includes metadata that should be internal
- MCP protocol translation may be exposing internal structures

Evidence:

```json
{
  "status": "continuation_available",
  "content": "Chat tool working...",
  "content_type": "text",
  "metadata": {
    "tool_name": "chat",
    "conversation_ready": true,
    "model_used": "glm-4.5-flash",
    "provider_used": "glm"
  },
  "continuation_offer": {
    "continuation_id": "62d15167-479e-4f32-9464-88c7db08b734",
    "note": "You can continue this conversation for 19 more exchanges.",
    "remaining_turns": 19
  }
}
```

Files Affected:
- `tools/simple/base.py` (lines 400-500, response formatting)
- `tools/simple/mixins/continuation_mixin.py` (continuation handling)
- `scripts/run_ws_shim.py` (lines 60-75, response cleaning)

Analysis:
This appears to be intentional design for conversation continuity, but:
1. Should be optional (not forced for all responses)
2. Metadata should be in separate channel (not in content)
3. Continuation offer should only appear when explicitly requested
4. Format is verbose and clutters simple responses

Recommended Fix:
1. Make continuation_id optional based on request parameter
2. Move metadata to separate response field
3. Only include continuation_offer when conversation mode is active
4. Simplify response format for single-turn operations

🟡 HIGH PRIORITY ISSUES

Issue #4: No "wave1" Branch Exists

Severity: P1 - HIGH
Impact: Documentation references non-existent branch, cannot compare changes
Status: CONFIRMED

Root Cause:
- Documentation references `wave1` branch that doesn't exist in repository
- Branch comparison document (`BRANCH_COMPARISON_wave1-to-auggie-optimization.md`) references non-existent branch
- Actual branch is `docs/wave1-complete-audit` (different name)

Evidence:

```
# From git ls-remote output:
80396b35eada5ca2d0f3703ff03a9a7319aef2de    refs/heads/docs/wave1-complete-audit
# No refs/heads/wave1 exists

# But documentation says:
"Previous Branch: docs/wave1-complete-audit"
"Current Branch: feat/auggie-mcp-optimization"
```

Files Affected:
- `/home/ubuntu/Uploads/BRANCH_COMPARISON_wave1-to-auggie-optimization.md`
- Documentation references throughout project

Impact:
- Cannot perform accurate branch comparison
- Unclear what “working wave1” state actually was
- May be comparing against wrong baseline

Recommended Fix:
1. Clarify which branch is the “working” baseline
2. Update documentation to reference correct branch names
3. Create proper branch comparison using actual branches
4. Document what functionality worked in baseline vs current

Issue #5: Timeout Configuration Chaos

Severity: P1 - HIGH
Impact: Unpredictable behavior, difficult to tune performance
Status: UNRESOLVED

Root Cause:
- Multiple timeout configurations across different layers
- Auggie CLI config overrides with very long timeouts (600s+)
- No clear timeout hierarchy or documentation
- Different tools have different timeout expectations

Evidence:

```
# From Daemon/mcp-config.auggie.json:
"EXAI_SHIM_RPC_TIMEOUT": "600"  # 10 minutes
"EXAI_WS_CALL_TIMEOUT": "600"   # 10 minutes
"KIMI_CHAT_TOOL_TIMEOUT_WEB_SECS": "900"  # 15 minutes

# From config.py:
EXPERT_ANALYSIS_TIMEOUT_SECS=90  # 90 seconds

# From tools/workflows/thinkdeep.py:
THINKDEEP_EXPERT_TIMEOUT_SECS=25  # 25 seconds (default)
```

Timeout Hierarchy (Current - Broken):

```
Tool Level:        25s (thinkdeep) / 90s (expert analysis)
Daemon Level:      600s (EXAI_WS_CALL_TIMEOUT)
Shim Level:        600s (EXAI_SHIM_RPC_TIMEOUT)
Provider Level:    900s (KIMI_CHAT_TOOL_TIMEOUT_WEB_SECS)
```

Problem: Inner timeouts (25s, 90s) will never trigger because outer timeouts (600s, 900s) are much longer. This creates a situation where:
1. Tool thinks it has 25s to complete
2. But daemon waits 600s before timing out
3. User experiences 10-minute hang instead of 25s timeout
4. No progress updates during the wait

Files Affected:
- `Daemon/mcp-config.auggie.json` (Auggie CLI overrides)
- `Daemon/mcp-config.augmentcode.json` (VSCode Augment overrides)
- `Daemon/mcp-config.claude.json` (Claude Desktop overrides)
- `config.py` (default timeouts)
- `src/daemon/ws_server.py` (daemon timeouts)
- All workflow tool files (tool-specific timeouts)

Recommended Fix:
1. Establish clear timeout hierarchy: Tool < Daemon < Shim < Client
2. Document timeout strategy in central location
3. Implement timeout coordination (inner timeout = 80% of outer timeout)
4. Add timeout warnings when approaching limits
5. Reduce Auggie config timeouts to reasonable values:
- Tool: 60s (simple) / 120s (workflow)
- Daemon: 180s
- Shim: 240s
- Client: 300s

Issue #6: Expert Validation Disabled

Severity: P1 - HIGH
Impact: Workflow tools missing key feature, quality degraded
Status: KNOWN (documented in MASTER_TASK_LIST)

Root Cause:
- Expert validation was calling analysis multiple times (duplicate calls)
- Temporarily disabled to prevent 300+ second hangs
- Bug not yet fixed, feature remains disabled

Evidence:

```
# From MASTER_TASK_LIST_2025-10-04.md:
### 1.1: Expert Validation System 🔴 CRITICAL BUG DISCOVERED
**Status:** DUPLICATE EXPERT ANALYSIS CALLS - Temporarily disabled
**NEW BUG:** Expert analysis being called MULTIPLE TIMES - DISCOVERED
- Temporarily disabled expert validation (DEFAULT_USE_ASSISTANT_MODEL=false)
```

Files Affected:
- `tools/workflow/expert_analysis.py` (expert validation logic)
- `tools/workflow/conversation_integration.py` (removed stub method)
- `.env` (DEFAULT_USE_ASSISTANT_MODEL=false)

Impact:
- Workflow tools don't get expert validation
- Quality of analysis reduced
- Missing key differentiator of the system

Recommended Fix:
1. Debug why expert analysis is called multiple times
2. Implement call deduplication
3. Add request tracking to prevent duplicate calls
4. Re-enable expert validation with proper safeguards
5. Add circuit breaker to prevent runaway calls

🟢 MEDIUM PRIORITY ISSUES

Issue #7: Native Web Search Integration Unclear

Severity: P2 - MEDIUM
Impact: Web search may not work as intended, unclear behavior
Status: PARTIALLY RESOLVED (per documentation)

Root Cause:
- Web search implementation split between GLM and Kimi
- GLM uses native web search tool (hidden from registry)
- Kimi uses `$web_search` builtin function
- Integration points not clearly documented in code

Evidence:

```
# From MASTER_TASK_LIST:
### 1.2: Web Search Integration in Chat Tool ✅ COMPLETE
- GLM web search function is HIDDEN from tool registry (internal function only)
- AI Manager (GLM-4.5-Flash) auto-triggers web search when use_websearch=true

### 1.3: Kimi Web Search Configuration ✅ COMPLETE
- Kimi uses `$web_search` builtin function (correct Moonshot API format)
```

Files Affected:
- `server.py` (line 260, glm_web_search hidden)
- `tools/simple/base.py` (lines 502-508, web search auto-injection)
- `src/providers/capabilities.py` (lines 45-81, web search schemas)
- `src/providers/orchestration/websearch_adapter.py`

Analysis:
According to documentation, web search is implemented and working:
- GLM: Native web search via tools schema
- Kimi: Builtin `$web_search` function
- Auto-injection in SimpleTool.execute()

However, code inspection shows:
1. Web search tool is hidden from registry (correct)
2. Auto-injection logic exists (correct)
3. But unclear if it's actually being called
4. No logging of web search activation
5. No tests verifying web search works

Recommended Fix:
1. Add logging when web search is activated
2. Add tests for web search integration
3. Document web search flow in architecture docs
4. Add metrics for web search usage
5. Verify web search works in both GLM and Kimi

Issue #8: MCP Configuration Inconsistency

Severity: P2 - MEDIUM
Impact: Different behavior across clients, hard to maintain
Status: UNRESOLVED

Root Cause:
- Three different MCP configurations (Auggie, Augment, Claude)
- Each has different timeout and concurrency settings
- No clear documentation of differences
- Auggie config has extreme values (600s+ timeouts)

Evidence:

```
// Daemon/mcp-config.auggie.json
"EXAI_SHIM_RPC_TIMEOUT": "600"
"EXAI_WS_CALL_TIMEOUT": "600"
"EXAI_WS_SESSION_MAX_INFLIGHT": "6"

// Daemon/mcp-config.augmentcode.json
// (Different values - need to check)

// Daemon/mcp-config.claude.json
// (Different values - need to check)
```

Files Affected:
- `Daemon/mcp-config.auggie.json`
- `Daemon/mcp-config.augmentcode.json`
- `Daemon/mcp-config.claude.json`

Recommended Fix:
1. Standardize configurations across clients
2. Document why differences exist (if necessary)
3. Create base configuration with client-specific overrides
4. Add validation for configuration values
5. Test all three configurations regularly

Issue #9: Bootstrap Module Complexity

Severity: P2 - MEDIUM
Impact: Harder to maintain, potential initialization issues
Status: NEW ARCHITECTURE (from refactoring)

Root Cause:
- New bootstrap modules created during refactoring
- Consolidates initialization code (good)
- But adds another layer of indirection
- May have initialization order dependencies

Evidence:

```python
# From scripts/run_ws_shim.py:
from src.bootstrap import load_env, get_repo_root, setup_logging

# Bootstrap: Setup path and load environment
load_env()  # Must be called first
logger = setup_logging("ws_shim", ...)  # Then logging
```

Files Affected:
- `src/bootstrap/__init__.py`
- `src/bootstrap/env_loader.py`
- `src/bootstrap/logging_setup.py`
- All entry point scripts

Analysis:
This is actually an improvement from the previous duplicated code, but:
1. Initialization order matters (env → logging → everything else)
2. No validation of initialization success
3. Silent failures possible
4. No rollback on partial initialization

Recommended Fix:
1. Add initialization validation
2. Implement proper error handling
3. Add initialization status tracking
4. Document initialization order requirements
5. Add tests for bootstrap module

🔵 LOW PRIORITY ISSUES

Issue #10: File Path Validation Too Strict

Severity: P3 - LOW
Impact: User experience issue, already has workaround
Status: RESOLVED (per documentation)

Root Cause:
- File path validation required absolute paths
- `EX_ALLOW_RELATIVE_PATHS` defaulted to false
- Fixed by changing default to true

Evidence:

```
# From COMPREHENSIVE_TOOL_TESTING_2025-10-03.md:
### Fix Applied
1. Changed default from "false" to "true" in base_tool_file_handling.py line 96
2. Added documentation to .env.example
3. Added EX_ALLOW_RELATIVE_PATHS=true to .env
```

Status: ✅ RESOLVED

Issue #11: Continuation ID Expiration

Severity: P3 - LOW
Impact: User experience issue, conversations expire
Status: BY DESIGN

Root Cause:
- Conversations expire after 3 hours
- No warning before expiration
- Error message could be clearer

Evidence:

```
{
  "timestamp": 1759100315.8598037,
  "tool": "chat",
  "error": "Conversation thread 'ctx-ce818efc' was not found or has expired..."
}
```

Files Affected:
- Conversation storage system
- Error message formatting

Recommended Fix:
1. Add warning when conversation approaching expiration
2. Improve error message with recovery instructions
3. Consider longer expiration time (6-12 hours)
4. Add conversation persistence option

BREAKING CHANGES ANALYSIS

Changes from "wave1" to feat/auggie-mcp-optimization

Note: Cannot perform accurate comparison because "wave1" branch doesn't exist. Using `docs/wave1-complete-audit` as baseline.

Major Changes (from BRANCH_COMPARISON document):

1. **Bootstrap Modules Created** (NEW)
   - Consolidated initialization code

- Reduced duplication across entry points
- **Impact:** Positive (cleaner code)

2. **Mixin Pattern Implementation** (NEW)
   - Extracted mixins from monolithic classes
   - Better separation of concerns
   - **Impact:** Positive (maintainability)

3. **Auggie MCP Config Optimization** (MODIFIED)
   - Extended timeouts (600s+)
   - Increased concurrency limits
   - **Impact:** NEGATIVE (causes hanging issues)

4. **Critical Bug Fixes** (FIXED)
   - Server crash on startup (status.py)
   - Web search integration (text_format_handler.py)
   - Legacy “zen” references removed
   - **Impact:** Positive (stability)

5. **Expert Validation Disabled** (CHANGED)
   - Temporarily disabled due to duplicate call bug
   - **Impact:** NEGATIVE (missing feature)

What Broke:

1. **Workflow Tools:** Worked in baseline, broken in current
   - Root cause: Timeout configuration changes
   - Auggie config extended timeouts too much
   - No progress heartbeat added

2. **Logging:** Worked in baseline, inconsistent in current
   - Root cause: Different code paths for workflow tools
   - Refactoring may have broken logging integration

3. **Expert Validation:** Worked in baseline, disabled in current
   - Root cause: Duplicate call bug discovered
   - Temporarily disabled, not yet fixed

What Improved:

1. **Code Organization:** Better with bootstrap and mixins
2. **Bug Fixes:** Several critical bugs fixed
3. **Documentation:** Extensive documentation added (77 files)
4. **Testing:** New test infrastructure added

API INTEGRATION ANALYSIS

GLM Provider (ZhipuAI/Z.ai)

Status: ✅ WORKING

Configuration:

```
GLM_API_KEY=configured
GLM_BASE_URL=https://api.z.ai/v1
GLM_DEFAULT_MODEL=glm-4.6
```

Available Models:
- glm-4.6 (flagship, 200K context)
- glm-4.5-flash (manager, fast/cheap)
- glm-4.5, glm-4.5-air, glm-4.5-x

Features:
- ✅ Native web search (via tools schema)
- ✅ Streaming support
- ✅ Tool calling
- ✅ SDK integration (zai-sdk v0.0.4)

Issues:
- Web search tool hidden from registry (correct)
- No logging of web search activation
- Unclear if web search actually works

Kimi Provider (Moonshot)

Status: ✅ WORKING

Configuration:

```
KIMI_API_KEY=configured
KIMI_BASE_URL=https://api.moonshot.ai/v1
KIMI_DEFAULT_MODEL=kimi-k2-0905-preview
```

Available Models:
- kimi-k2-0905-preview (recommended, 256K context)
- kimi-thinking-preview (deep reasoning)
- kimi-k2-turbo-preview (fast)
- Legacy: moonshot-v1-128k, moonshot-v1-32k, moonshot-v1-8k

Features:
- ✅ Native web search (`$web_search` builtin)
- ✅ Streaming support
- ✅ File upload/extract
- ✅ Advanced caching (automatic)
- ✅ OpenAI-compatible API

Issues:
- File cleanup may not be happening
- Cache management unclear
- No metrics for cache hit rate

Manager-First Routing

Status: 🟡 PARTIALLY IMPLEMENTED

Design:

```
Level 1: GLM-4.5-flash (Manager)
  ↓ (if complex)
Level 2: GLM-4.6
  ↓ (if specialized)
Level 3: Kimi
```

Issues:
- Routing logic exists but unclear if it’s working
- No logging of routing decisions
- No metrics for routing effectiveness
- Manager may not be making optimal decisions

Files:
- `src/router/service.py`
- `src/router/classifier.py`
- `src/router/unified_router.py`

TOOL EXECUTION FLOW ANALYSIS

Simple Tools (WORKING)

```
1. Client sends MCP request
   ↓
2. run_ws_shim.py receives request
   ↓
3. WebSocket connection to daemon (ws://127.0.0.1:8765)
   ↓
4. ws_server.py receives request
   ↓
5. server.py routes to tool
   ↓
6. SimpleTool.execute() runs
   ↓
7. Provider API call (GLM/Kimi)
   ↓
8. Response formatted with continuation_id
   ↓
9. Response sent back through WebSocket
   ↓
10. run_ws_shim.py cleans response
   ↓
11. Client receives response
```

Timing: 1-15 seconds (typical)
Logging: ✅ Works correctly
Issues: Continuation_id structure verbose

Workflow Tools (BROKEN)

```
1. Client sends MCP request
   ↓
2. run_ws_shim.py receives request
   ↓
3. WebSocket connection to daemon
   ↓
4. ws_server.py receives request
   ↓
5. server.py routes to workflow tool
   ↓
6. WorkflowTool.execute() runs
   ↓
7. Step-by-step execution begins
   ↓
8. Provider API call (GLM/Kimi)
   ↓
9. [HANGS HERE - No progress updates]
   ↓
10. [Timeout after 600s]
   ↓
11. [No response or error]
```

Timing: 600+ seconds (hangs)
Logging: 🔴 Not working
Issues:
- No progress heartbeat
- No timeout handling
- No logging during execution
- Expert validation disabled

LOGGING SYSTEM ANALYSIS

Current Logging Implementation

Log Files:
- `.logs/toolcalls.jsonl` - Tool execution logs
- `.logs/metrics.jsonl` - Performance metrics
- `.logs/ws_daemon.log` - Daemon logs
- `.logs/ws_shim.log` - Shim logs

Logging for Simple Tools: ✅ WORKING

```
{
  "timestamp": 1759100327.2925136,
  "tool": "chat",
  "request_id": null,
  "duration_s": 11.423,
  "result_preview": "...",
  "prompt_bullets": ["..."],
  "summary_words": 600,
  "summary_text": "..."
}
```

Logging for Workflow Tools: 🔴 NOT WORKING

- No entries in toolcalls.jsonl when tools hang
- No progress messages captured
- No error messages logged
- Silent failures

Root Cause Analysis

Simple Tools:

```
# From tools/simple/base.py
# Logging happens in execute() method
# Progress messages sent via send_progress()
# Results logged to toolcalls.jsonl
```

Workflow Tools:

```
# From tools/workflow/base.py
# Different execution path
# Progress messages may not be captured
# Logging may not be called
# Different error handling
```

Problem: Workflow tools use different code path that doesn’t integrate with logging system properly.

CLIENT CONFIGURATION ANALYSIS

VSCode Augment Configuration

File: `Daemon/mcp-config.augmentcode.json`

Status: Need to inspect (not fully analyzed)

Expected Issues:
- May have different timeout values
- May have different concurrency limits
- Should be tested separately

Auggie CLI Configuration

File: `Daemon/mcp-config.auggie.json`

Status: 🔴 PROBLEMATIC

Configuration:

```
{
  "EXAI_SHIM_RPC_TIMEOUT": "600",
  "EXAI_WS_CALL_TIMEOUT": "600",
  "EXAI_WS_SESSION_MAX_INFLIGHT": "6",
  "EXAI_WS_GLOBAL_MAX_INFLIGHT": "16",
  "KIMI_CHAT_TOOL_TIMEOUT_WEB_SECS": "900",
  "EX_SESSION_SCOPE_STRICT": "false",
  "EX_SESSION_SCOPE_ALLOW_CROSS_SESSION": "true"
}
```

Issues:
1. **Timeouts Too Long:** 600s (10 min) causes hanging perception
2. **Concurrency Reduced:** From 12 to 6 (may be intentional)
3. **Session Scope Relaxed:** May cause cross-contamination

Rationale (from documentation):
- Extended timeouts for “max thinking mode”
- Support 30-60 minute autonomous sessions
- Better workflow continuity

Problem: While rationale makes sense, implementation causes:
- User perceives system as hanging
- No progress updates during long waits
- Timeout hierarchy broken
- No graceful degradation

Claude Desktop Configuration

File: `Daemon/mcp-config.claude.json`

Status: Need to inspect (not fully analyzed)

Expected Issues:
- May have different timeout values
- May have different concurrency limits
- Should be tested separately

CODE QUALITY ASSESSMENT

Architectural Strengths

1. **Modular Design:** Good separation between providers, tools, routing
2. **Mixin Pattern:** Clean extraction of concerns into mixins
3. **Bootstrap Modules:** Consolidated initialization code
4. **Provider Abstraction:** Clean interface for GLM and Kimi
5. **MCP Protocol:** Proper implementation of MCP standard

Architectural Weaknesses

1. **Timeout Management:** Scattered across multiple files, no coordination
2. **Logging Inconsistency:** Different paths for simple vs workflow tools

3. **Error Handling:** Silent failures, no graceful degradation
4. **Progress Reporting:** Missing for long-running operations
5. **Configuration Complexity:** Too many environment variables, unclear hierarchy

Code Smells

1. **Magic Numbers:** Timeout values hardcoded in multiple places
2. **Duplicate Logic:** Timeout handling duplicated across tools
3. **Silent Failures:** Errors not propagated properly
4. **Missing Validation:** Configuration values not validated
5. **Inconsistent Patterns:** Simple vs workflow tools use different patterns

Design Issues

1. **Tight Coupling:** Timeout configuration tightly coupled to implementation
2. **Missing Abstraction:** No timeout manager or coordinator
3. **No Circuit Breaker:** Repeated failures not handled
4. **No Health Checks:** Long operations don't report health
5. **No Metrics:** Can't measure system performance

PRIORITY RANKING

P0 - CRITICAL (Must Fix Immediately)

1. **Workflow Tools Hanging** (Issue #1)
   - Impact: System unusable for complex tasks
   - Effort: Medium (2-3 days)
   - Fix: Implement timeout hierarchy + progress heartbeat

2. **Logging Not Working** (Issue #2)
   - Impact: Cannot debug issues
   - Effort: Medium (2-3 days)
   - Fix: Unify logging infrastructure

P1 - HIGH (Fix Soon)

1. **Timeout Configuration Chaos** (Issue #5)
   - Impact: Unpredictable behavior
   - Effort: Medium (2-3 days)
   - Fix: Standardize timeout hierarchy

2. **Expert Validation Disabled** (Issue #6)
   - Impact: Missing key feature
   - Effort: High (3-5 days)
   - Fix: Debug duplicate call issue

3. **Branch Comparison Issue** (Issue #4)
   - Impact: Cannot verify changes
   - Effort: Low (1 day)
   - Fix: Clarify baseline branch

P2 - MEDIUM (Fix When Possible)

1. **Continuation ID Structure** (Issue #3)
   - Impact: Confusing output
   - Effort: Low (1-2 days)
   - Fix: Make optional, simplify format

2. **Web Search Integration** (Issue #7)
   - Impact: Feature may not work
   - Effort: Medium (2-3 days)
   - Fix: Add logging, tests, verification

3. **MCP Config Inconsistency** (Issue #8)
   - Impact: Different behavior across clients
   - Effort: Medium (2-3 days)
   - Fix: Standardize configurations

P3 - LOW (Nice to Have)

1. **Bootstrap Module Complexity** (Issue #9)
   - Impact: Maintenance burden
   - Effort: Low (1-2 days)
   - Fix: Add validation, tests

2. **Continuation Expiration** (Issue #11)

   - Impact: User experience
   - Effort: Low (1 day)
   - Fix: Better warnings, longer expiration

RECOMMENDED FIX SEQUENCE

Phase 1: Critical Fixes (Week 1)

Goal: Make workflow tools functional

1. **Day 1-2: Implement Timeout Hierarchy**
   - Define clear timeout hierarchy
   - Implement timeout coordination
   - Add timeout warnings
   - Test with workflow tools

2. **Day 3-4: Add Progress Heartbeat**
   - Implement progress reporting for long operations
   - Add heartbeat every 5-8 seconds
   - Test with workflow tools
   - Verify user experience

3. **Day 5: Fix Logging**
   - Unify logging infrastructure
   - Ensure workflow tools log correctly
   - Add structured logging
   - Test logging for all tools

Phase 2: High Priority Fixes (Week 2)

Goal: Restore full functionality

1. **Day 6-8: Fix Expert Validation**
   - Debug duplicate call issue
   - Implement call deduplication
   - Re-enable expert validation
   - Test with workflow tools

2. **Day 9-10: Standardize Configurations**
   - Create base configuration
   - Standardize timeout values
   - Document configuration hierarchy
   - Test all three clients

Phase 3: Medium Priority Fixes (Week 3)

Goal: Improve reliability and usability

1. **Day 11-12: Simplify Continuation System**
   - Make continuation_id optional
   - Simplify response format
   - Move metadata to separate field
   - Test with all tools

2. **Day 13-14: Verify Web Search**
   - Add web search logging
   - Create web search tests
   - Verify GLM and Kimi integration
   - Document web search flow

3. **Day 15: Documentation Update**
   - Update all documentation
   - Document timeout hierarchy
   - Document configuration options
   - Create troubleshooting guide

TESTING RECOMMENDATIONS

Unit Tests Needed

1. **Timeout Management**
   - Test timeout hierarchy
   - Test timeout coordination
   - Test timeout warnings
   - Test graceful timeout

2. **Logging System**
   - Test simple tool logging
   - Test workflow tool logging

- Test structured logging
- Test log rotation

3. **Continuation System**
   - Test continuation creation
   - Test continuation retrieval
   - Test continuation expiration
   - Test cross-session continuations

Integration Tests Needed

1. **Workflow Tools**
   - Test analyze tool end-to-end
   - Test thinkdeep tool end-to-end
   - Test debug tool end-to-end
   - Test codereview tool end-to-end

2. **Web Search**
   - Test GLM web search
   - Test Kimi web search
   - Test web search auto-injection
   - Test web search results

3. **Expert Validation**
   - Test expert validation flow
   - Test duplicate call prevention
   - Test expert timeout handling
   - Test expert result integration

Performance Tests Needed

1. **Timeout Behavior**
   - Measure actual timeout values
   - Test timeout cascade
   - Test progress heartbeat frequency
   - Test graceful degradation

2. **Concurrency**
   - Test session concurrency limits
   - Test global concurrency limits
   - Test provider concurrency limits
   - Test concurrent workflow tools

3. **Logging Performance**
   - Measure logging overhead
   - Test log file size growth
   - Test log rotation
   - Test log parsing performance

CONCLUSION

The EX-AI MCP Server on the `feat/auggie-mcp-optimization` branch has **critical architectural issues** that prevent complex workflow tools from functioning correctly. While simple tools work, the system is **not production-ready** for complex use cases.

Key Takeaways

1. **Simple Tools Work:** Chat and other simple tools function correctly
2. **Workflow Tools Broken:** Analyze, thinkdeep, debug, codereview hang without timeout
3. **Logging Inconsistent:** Works for simple tools, broken for workflow tools
4. **Timeout Chaos:** Multiple conflicting timeout configurations
5. **Expert Validation Disabled:** Key feature temporarily disabled due to bug
6. **Documentation Mismatch:** References non-existent “wave1” branch

Root Causes

1. **Timeout Hierarchy Broken:** Inner timeouts never trigger due to outer timeouts being too long
2. **No Progress Heartbeat:** Long operations don’t report progress
3. **Logging Path Divergence:** Workflow tools use different code path without logging
4. **Configuration Overrides:** Auggie config extends timeouts too much (600s+)
5. **Missing Error Handling:** Silent failures instead of graceful degradation

Immediate Actions Required

1. **Fix Timeout Hierarchy:** Implement proper timeout coordination
2. **Add Progress Heartbeat:** Report progress every 5-8 seconds
3. **Fix Logging:** Unify logging infrastructure for all tools
4. **Reduce Timeouts:** Change Auggie config to reasonable values (90-180s)
5. **Test Workflow Tools:** Verify all workflow tools work correctly

Long-Term Improvements

1. **Standardize Configurations:** Create base config with client overrides
2. **Fix Expert Validation:** Debug and re-enable expert validation
3. **Add Monitoring:** Implement health checks and metrics
4. **Improve Documentation:** Update all docs with accurate information
5. **Add Tests:** Create comprehensive test suite

Report Generated: 2025-10-04
Analyst: Abacus.AI Deep Agent
Repository: https://github.com/Zazzles2908/EX-AI-MCP-Server
Branch: feat/auggie-mcp-optimization
Commit: e27cf6f (docs: add branch comparison documentation)