Zen-CLI Schema Validation Fixes Required

Executive Summary

After fixing critical schema validation issues in zen-mcp-server's two-stage token optimization architecture, analysis of the zen-cli codebase reveals **similar Pydantic schema validation issues** that need immediate attention. While zen-cli doesn't use the two-stage optimization, it shares the same problematic Literal constraint patterns that cause MCP protocol violations.

Status: zen-cli is currently on version 5.12.0 and needs these fixes to maintain MCP compatibility.

Background Context

In zen-mcp-server v5.12.1, we fixed multiple schema validation issues: - Literal constraint mismatches between tools and response models - Required vs optional field inconsistencies - Case sensitivity issues in Literal values - Mode name standardization

zen-cli uses the original full-schema architecture (43k tokens) and doesn't have two-stage optimization, so mode_selector.py and mode_executor.py fixes don't apply. However, it has the same underlying Pydantic request models with identical validation issues.

Critical Issues Found in zen-CLI

1. Literal Constraint Inconsistencies

Problem: Multiple tools use Literal constraints that are inconsistent with response models in `models.py`.

Confidence Level Mismatches

```python
# In models.py line 289 (DiagnosticHypothesis)
confidence: Literal["high", "medium", "low"]  # lowercase

# In models.py line 314 (DebugHypothesis)
confidence: Literal["High", "Medium", "Low"]  # capitalized

# In refactor.py line 112 (RefactorRequest)
confidence: Optional[Literal["exploring", "incomplete",
        "partial", "complete"]]  # different values

# In codereview.py line 95 (CodeReviewRequest)
confidence: Optional[str] = Field("low", exclude=True)  # no
        Literal constraint
```

Fix Required: Standardize all confidence fields to use lowercase Literal values: `["high", "medium", "low"]`

Trace Mode Inconsistencies

```
# In tracer.py line 107 (TracerRequest)
trace_mode: Optional[Literal["precision", "dependencies", "ask"]]

# In models.py line 262 (TraceComplete)
trace_type: Literal["precision", "dependencies"]  # missing
        "ask", different field name
```

Fix Required: Ensure trace_mode values match between request and response models.

2. Schema Validation Files Requiring Updates

Files with Literal Constraints (Priority Order):

1. `/src/zen_cli/tools/models.py` (lines 289, 314)
    - Fix confidence case inconsistency
    - Standardize across DiagnosticHypothesis and DebugHypothesis
2. `/src/zen_cli/tools/refactor.py` (lines 112, 123)
    - Update confidence values to match models.py standard
    - Ensure refactor_type values are consistent
3. `/src/zen_cli/tools/tracer.py` (line 107)
    - Update trace_mode values to match response models
    - Fix field name consistency (trace_mode vs trace_type)
4. `/src/zen_cli/tools/codereview.py` (line 95)
    - Replace `Optional[str]` with proper Literal constraint for confidence
    - Remove exclude=True for consistency
5. `/src/zen_cli/tools/analyze.py` (lines 121, 124)
    - Verify analysis_type and output_format values match usage
6. `/src/zen_cli/tools/secaudit.py` (lines 124, 130, 133)
    - Check threat_level, audit_focus, severity_filter consistency
7. `/src/zen_cli/tools/precommit.py` (lines 94, 115)
    - Verify precommit_type and severity_filter values

3. Complete Literal Constraint Inventory

From grep analysis, zen-cli has Literal constraints in:

```
refactor.py:        confidence, refactor_type
analyze.py:         analysis_type, output_format
precommit.py:       precommit_type, severity_filter
models.py:          Multiple status and type fields
codereview.py:      review_validation_type, review_type,
severity_filter
tracer.py:          trace_mode
secaudit.py:        threat_level, audit_focus, severity_filter
```

Recommended Fix Implementation Steps

Step 1: Standardize Confidence Fields

```
# Everywhere confidence is used, standardize to:
confidence: Optional[Literal["high", "medium", "low"]] = Field(
    "low",
    description="Confidence level in the analysis"
)
```

Step 2: Fix models.py Inconsistencies

```
# Line 289 - DiagnosticHypothesis (already correct)
confidence: Literal["high", "medium", "low"] = Field(...)

# Line 314 - DebugHypothesis (needs fix)
confidence: Literal["high", "medium", "low"] = Field(...)  #
        Change from "High", "Medium", "Low"
```

Step 3: Update Tool Request Models

For each tool file, ensure Literal values match the corresponding response models in models.py:

```
# refactor.py - Replace exploration-specific confidence with
        standard
confidence: Optional[Literal["high", "medium", "low"]] =
        Field("low", ...)

# tracer.py - Ensure trace_mode matches models.py trace_type
trace_mode: Optional[Literal["precision", "dependencies"]] =
        Field("precision", ...)

# codereview.py - Add proper Literal constraint
confidence: Optional[Literal["high", "medium", "low"]] =
        Field("low", ...)
```

Step 4: Version Update

Update `/src/zen_cli/config.py`:

```
__version__ = "5.12.1"  # Match zen-mcp-server fixed version
__updated__ = "2025-09-05"
```

Testing Recommendations

After applying fixes:

1. **Unit Tests**: Run existing test suite

```
python -m pytest tests/ -v
```

1. **Schema Validation**: Create simple MCP client test to validate all tool schemas
2. **Integration Test**: Test each tool with various confidence levels to ensure validation passes

Files NOT Requiring Changes

Two-stage optimization files (don't exist in zen-cli): - `tools/mode_selector.py` - N/A (zen-cli doesn't have two-stage) - `tools/mode_executor.py` - N/A (zen-cli doesn't have two-stage)

Provider files: Should be similar to zen-mcp-server and likely don't need changes unless they have Literal constraints.

Risk Assessment

Low Risk: These are schema validation fixes, not functional changes **High Impact**: Prevents MCP protocol violations and client errors **Backward Compatibility**: Maintained (only fixes validation, doesn't change behavior)

Success Criteria

- [ ] All Literal constraint values are consistent between request and response models
- [ ] Confidence levels standardized to lowercase format across all tools
- [ ] Version updated to 5.12.1
- [ ] All existing tests pass
- [ ] No MCP schema validation errors

Implementation Time Estimate

2-3 hours for careful implementation: - 1 hour: Fix all Literal constraints in tool files - 30 minutes: Update models.py inconsistencies
- 30 minutes: Version update and testing - 30 minutes: Validation and edge case testing

---

Generated by: zen-mcp-server analysis (version 5.12.1)
Target: zen-cli codebase (currently version 5.12.0)
Priority: High - MCP compatibility and schema validation
Date: 2025-09-05