Software Analysis Intake Template

AXIOM Analysis

Software Analysis Intake Template

Version 1.0 | April 12, 2026
Working template for structured software runs in the AXIOM portal.

Purpose

Use this template to prepare a software analysis run in AXIOM.

It is designed for:

- code review
- architecture review
- delivery-risk review
- reliability and maintainability assessment
- rapid first-pass technical scans

This is a structured analysis intake, not a substitute for full engineering review, security testing, or production sign-off.

Best AXIOM Settings

Code / Change Review

- Domain: `Software`
- Modules: `review`
- Report Depth: `Standard`
- Reasoning Engine: `AXIOM Gateway` or `Auto`
- Live sources: `off`

Use this when the main goal is to examine:

- likely defects
- regressions
- missing validation or tests
- unsafe assumptions
- rollout risks

Architecture Review

- Domain: `Software`
- Modules: `architecture`
- Report Depth: `Standard`
- Reasoning Engine: `AXIOM Gateway` or `Auto`
- Live sources: `off`

Use this when the main goal is to examine:

- system boundaries
- coupling
- flow
- stress points
- scaling or reliability choke points

Combined Technical Review

- Domain: `Software`
- Modules: `review`, `architecture`
- Report Depth: `Standard` first, `Deep` second if needed
- Reasoning Engine: `AXIOM Gateway` or `Auto`

- Live sources: `off`

Use this when you want the broadest decision-support pass across code and structure.

Quick Triage Only

- Domain: `Software`
- Modules: `fast`
- Report Depth: `Compact` or `Standard`
- Reasoning Engine: `AXIOM Gateway`
- Live sources: `off`

Use this only for a rapid first-pass scan, not your best final review.

Important Input Note

For the cleanest current behavior:

- upload the actual source material, not just a vague description
- keep the request prompt short
- include expected behavior and suspected risk areas
- do not expect strong review output if the code or architecture detail is missing

Best current upload shape:

- `01_request.md`
- `02_system_context.md`
- `03_code_or_change_scope.md`
- `04_architecture_and_runtime.md`
- `05_logs_tests_and_risks.md`

You do not need all five files for every run.

For first tests, do one of these:

1. Paste the full technical material directly into the request box and upload nothing.
2. Upload the material files and use the short prompt file only.

Do not duplicate the entire same code/context in both places unless you have a reason to.

Data Collection Checklist

Collect as much of the following as possible before running the analysis.

1. Run Basics

- System / project name:
- Run type:
  - code review
  - architecture review
  - combined technical review
  - quick scan
- Reviewer:
- Date prepared:
- Main decision or question:

2. System Basics

- Product / service name:
- Main language(s):
- Framework(s):
- Runtime / platform:
- Environment:
  - local
  - staging
  - production
- Team / ownership:
- Current delivery stage:

3. Review Goal

- What exactly is being reviewed:

- Why now:
- What changed:
- What are you worried about:
- What would make this unsafe to ship:

4. Scope Definition

Collect:

- affected modules
- touched files
- service boundaries
- APIs or interfaces involved
- database tables or schemas involved
- infra or config changes involved

5. Expected Behavior

Collect:

- expected system behavior
- current observed behavior
- known bug or incident symptoms
- critical user flows
- success criteria

6. Code / Change Evidence

For each important change or code area, collect:

- file or module name
- what it does
- what changed
- whether behavior changed intentionally or accidentally
- any tests added or missing

7. Architecture Context

Collect:

- component boundaries
- request / event / data flow
- state ownership
- shared dependencies
- cross-service coupling
- deployment assumptions
- external systems involved

8. Risk Set

Collect the strongest known risks:

- correctness risk
- regression risk
- security-sensitive area
- performance / scaling risk
- reliability / retry risk
- consistency or concurrency risk
- migration / rollout risk
- observability gap
- ownership ambiguity

9. Validation Material

Collect:

- test plan
- existing tests
- missing tests
- logs
- stack traces
- metrics
- repro steps
- screenshots if relevant, but convert important details to text

10. Environment / Runtime Assumptions

Collect:

- environment-specific behavior
- feature flags
- secrets or config dependencies
- job scheduling assumptions
- queue / cache / database dependencies
- network or third-party dependencies

11. Known Weak Spots

List anything already suspected:

- brittle module
- unclear ownership
- hidden dependency
- migration concern
- no rollback path
- no monitoring
- unusual performance sensitivity

12. Source Map

Separate clearly:

- directly observed code or config
- logs or runtime evidence
- ticket or human description
- your own inference or concern

This is one of the most important quality steps. Do not mix them together.

13. Missing Inputs

List what is not yet known:

- incomplete code scope
- missing architecture diagram
- no logs
- no repro steps
- no tests
- no expected behavior statement
- no rollout plan

14. Output Goal

What should the AXIOM report help you do?

- find likely bugs
- catch regressions
- see architecture stress points
- identify missing validation
- understand whether risk is localized or systemic
- decide what to test or inspect next

Suggested File Pack

File 1: `01_request.md`

Use this for the decision context:

```
System / project name:
Run type:
Decision to support:

Top questions:
1.
2.
3.

Known constraints:

Known uncertainties:
```

File 2: `02_system_context.md`

Use this for the basic system picture:

```
System overview:

Main components:

Languages / frameworks:

Runtime / deployment context:

Expected behavior:
```

File 3: `03_code_or_change_scope.md`

Use this for the review target:

```
Files / modules in scope:

What changed:

Behavioral change intended:

Known concerns:
```

File 4: `04_architecture_and_runtime.md`

Use this for structure and interfaces:

```
Component boundaries:

Data / control flow:

Dependencies:

Runtime assumptions:

Stress points already suspected:
```

File 5: `05_logs_tests_and_risks.md`

Use this for evidence and validation:

```
Logs / traces:

Tests present:

Tests missing:

Risks:

Open questions:
```

Fill-In Intake Block

Copy this block into a working `.md` file and complete it.

```
System / project name:
Run type:
Prepared by:
Prepared on:

Decision to support:

Primary review questions:
1.
2.
3.

Product / service:
Languages:
Frameworks:
Runtime / platform:
Environment:
Team / ownership:
Delivery stage:

What is being reviewed:
Why now:
What changed:
What worries us most:
What would make this unsafe to ship:

Scope:

Expected behavior:

Observed behavior:

Code / change evidence:

Architecture context:

Validation material:

Key risks:

Runtime assumptions:

Known weak spots:

Source map:

Missing information:

What a useful AXIOM output should focus on:
```

Recommended Prompt Pairing

Use this template together with:

- SOFTWARE_ANALYSIS_PROMPT_2026-04-12.txt

Best pattern:

1. fill out this intake
2. upload this intake plus the actual technical material
3. use the short prompt file in the portal request box

Practical Notes For Better Results

- Start with `Standard` depth, not `Deep`
- Use `review` for bug, regression, and validation-focused analysis
- Use `architecture` for system-boundary, coupling, and stress-point analysis
- Use `review + architecture` when you want the broadest pass
- Use `fast` only for a quick triage
- If the result is weak, improve the technical evidence package before making the prompt longer
- If the output feels generic, the system probably did not receive enough code, logs, or architecture detail

---

Caution

AXIOM can help surface:

- likely issues
- technical risks
- architecture stress points
- missing validation
- information gaps

It should not be treated as a full security audit, full performance benchmark, or final production sign-off.

End of Software Analysis Intake Template v1.0