/ask-questions-if-underspecified

Source: `~/.claude/skills/tob-ask-questions-if-underspecified/skills/ask-questions-if-underspecified/SKILL.md`

---

name: ask-questions-if-underspecified description: Clarify requirements before implementing. Use when serious doubts arise.

Ask Questions If Underspecified

When to Use

Use this skill when a request has multiple plausible interpretations or key details (objective, scope, constraints, environment, or safety) are unclear.

When NOT to Use

Do not use this skill when the request is already clear, or when a quick, low-risk discovery read can answer the missing details.

Goal

Ask the minimum set of clarifying questions needed to avoid wrong work; do not start implementing until the must-have questions are answered (or the user explicitly approves proceeding with stated assumptions).

Workflow

1) Decide whether the request is underspecified

Treat a request as underspecified if after exploring how to perform the work, some or all of the following are not clear:

- Define the objective (what should change vs stay the same)
- Define "done" (acceptance criteria, examples, edge cases)
- Define scope (which files/components/users are in/out)
- Define constraints (compatibility, performance, style, deps, time)
- Identify environment (language/runtime versions, OS, build/test runner)
- Clarify safety/reversibility (data migration, rollout/rollback, risk)

If multiple plausible interpretations exist, assume it is underspecified.

2) Ask must-have questions first (keep it small)

Ask 1-5 questions in the first pass. Prefer questions that eliminate whole branches of work.

Make questions easy to answer:

- Optimize for scannability (short, numbered questions; avoid paragraphs)
- Offer multiple-choice options when possible
- Suggest reasonable defaults when appropriate (mark them clearly as the default/recommended choice; bold the recommended choice in the list, or if you present options in a code block, put a bold "Recommended" line immediately above the block and also tag defaults inside the block)
- Include a fast-path response (e.g., reply `defaults` to accept all recommended/default choices)
- Include a low-friction "not sure" option when helpful (e.g., "Not sure - use default")
- Separate "Need to know" from "Nice to know" if that reduces friction

- Structure options so the user can respond with compact decisions (e.g., `1b 2a 3c`); restate the chosen options in plain language to confirm

3) Pause before acting

Until must-have answers arrive:

- Do not run commands, edit files, or produce a detailed plan that depends on unknowns
- Do perform a clearly labeled, low-risk discovery step only if it does not commit you to a direction (e.g., inspect repo structure, read relevant config files)

If the user explicitly asks you to proceed without answers:

- State your assumptions as a short numbered list
- Ask for confirmation; proceed only after they confirm or correct them

4) Confirm interpretation, then proceed

Once you have answers, restate the requirements in 1-3 sentences (including key constraints and what success looks like), then start work.

Question templates

- "Before I start, I need: (1) ..., (2) ..., (3) .... If you don't care about (2), I will assume ...."
- "Which of these should it be? A) ... B) ... C) ... (pick one)"
- "What would you consider 'done'? For example: ..."
- "Any constraints I must follow (versions, performance, style, deps)? If none, I will target the existing project defaults."
- Use numbered questions with lettered options and a clear reply format

```
1) Scope?
a) Minimal change (default)
b) Refactor while touching the area
c) Not sure - use default
2) Compatibility target?
a) Current project defaults (default)
b) Also support older versions: <specify>
c) Not sure - use default

Reply with: defaults (or 1a 2a)
```

Anti-patterns

- Don't ask questions you can answer with a quick, low-risk discovery read (e.g., configs, existing patterns, docs).
- Don't ask open-ended questions if a tight multiple-choice or yes/no would eliminate ambiguity faster.

Revision #5
Created 2026-02-18 08:40:02 UTC by John
Updated 2026-06-21 20:01:09 UTC by John