AI CUSTOMER SUCCESS / FIELD GUIDE

Debugging Claude-Based AI Agents

A practical guide for turning vague customer complaints into reproducible evidence, likely cause, impact, and a verified outcome.

Core mindset: Debug the whole system, not just Claude. Find the first point where actual behavior stopped matching expected behavior, collect clear evidence, and help the right owner act.

Quick orientation

What is an agent?

Claude + instructions + data/context + tools + workflow rules that let it answer questions or take actions.

What is debugging?

Finding the first meaningful difference between expected and actual behavior.

What is a trace?

A step-by-step record of one request: inputs, model calls, tool calls, outputs, errors, timing, and usage.

Your value

Turn a vague complaint into evidence another team can reproduce, prioritize, and fix without re-interviewing the customer.

Simple example

Customer: "Has my order shipped?" Tool result: "label created" Agent: "Yes, it shipped."

Step	What happened	Debug question
Request	Asked whether order shipped.	Was the request understood?
Tool call	Looked up the right order.	Correct tool and ID?
Tool result	Status = label created.	Correct and current upstream data?
Interpretation	Treated label-created as shipped.	Were status definitions/instructions unclear?
Final answer	Incorrect confirmation.	Was a fallback/escalation rule missing?

A likely fix is to define which statuses actually confirm carrier possession, then test nearby statuses and wording variations before calling the issue fixed.

AI CUSTOMER SUCCESS / FIELD GUIDE

Where Agent Failures Come From

Classify the likely failing layer before deciding who should change what.

Common failure sources

Source	What it looks like	What to check
User request	Ambiguous wording, missing identifier.	Exact wording + needed account context.
Instructions	Policy missing, vague, conflicting, outdated.	Compare behavior with the written rule.
Conversation context	Earlier turns missing or misunderstood.	Read full thread; find first context drift.
Knowledge / retrieval	Wrong, stale, irrelevant, missing docs.	Source used, date, expected source.
Tool choice	Wrong tool or skipped needed tool.	Available descriptions vs selected tool.
Tool inputs	Wrong customer ID/date/amount.	Arguments vs known facts.
Tool result	API error, timeout, empty/bad data.	Raw result; separate upstream failure from interpretation.
Permissions	Access denied or action allowed incorrectly.	User, role, scope, approval, intended policy.
Model response	Correct facts, poor certainty/tone/policy.	Final wording vs expected behavior.
Workflow	Loops, repeated calls, bad handoff.	Retry count, stop and escalation rules.
Operations	Slow, expensive, intermittent.	Timing, usage, errors, customer impact.

The same bad final answer can come from very different causes. Route the fix based on the first wrong layer, not the symptom.

AI CUSTOMER SUCCESS / FIELD GUIDE

Trace Reading Workflow

Read left to right. Stop at the first input, decision, tool result, or interpretation that diverges from expectation.

Step-by-step debugging

1. **Clarify expected behavior.** One plain sentence: Given X, the agent should do Y. Confirm source-of-truth policy.
2. **Capture actual behavior.** Exact request, response, timestamps, conversation/request ID, account conditions.
3. **Assess urgency and safety.** Escalate privacy exposure, unauthorized actions, financial harm, security risk, or broad impact immediately.
4. **Find the trace.** Search by conversation/request ID, account, time, environment, or trace ID.
5. **Read from left to right.** User input → context/retrieval → model → tool calls → tool results → final output.
6. **Classify likely cause.** Instruction, context, retrieval, tool, permissions, model wording, workflow, or operations.
7. **Reproduce safely.** Start with the exact request in a test/approved environment; change one condition at a time.
8. **Compare.** Pair the failing trace with a similar good trace when possible.
9. **Route smallest fix.** CS may improve examples/tests; engineering changes code/tools; security changes policy; PM owns product tradeoffs.
10. **Regression + close loop.** Original, paraphrases, neighbors, permissions/failures, customer update, monitoring plan.

Trace-reading checklist

Input + instructions

Exact user wording, account facts, environment, instruction version.

Context + retrieval

Sources returned, freshness, missing expected source.

Tool decision

Chosen tool, skipped tool, ordering.

Arguments + result

Inputs, error/status, timing, retries.

Next model step

Did interpretation follow from the result?

Performance

Per-step latency, repeated work, tool/model usage.

AI CUSTOMER SUCCESS / FIELD GUIDE

Reproduction, Evals & Tool Calls

A fix is not one successful rerun. Build enough nearby cases to show the behavior improved without breaking adjacent behavior.

Turn a bug into a compact eval set

ORIGINAL
Exact reported failure.

PARAPHRASES
Same intent, different wording.

POSITIVE
Case where the agent should confidently act.

NEIGHBORING
Nearby status/intent that needs a different response.

MISSING DATA
Clarification and fallback behavior.

PERMISSION / FAILURE
Blocked action, timeout, empty result, escalation.

Tool-call mental model

Tool name + description → **Input schema** → **Structured tool call** → **Application executes** → **Structured result** → **Agent interprets**

Good descriptions reduce ambiguity

Vague	Clearer
check_order	Retrieve latest order/shipping status for a verified order ID. Status questions only; do not modify order.
refund	Create a refund only after eligibility and user authorization are confirmed. Requires order ID, amount, reason, approval token.
search_customer	Find a customer using verified email/account ID. Return possible matches; do not expose records without authorization.

Safe reproduction rule: never repeat a risky action on a real customer account just to prove it fails. Use approved test data and environments.

AI CUSTOMER SUCCESS / FIELD GUIDE

Safety, Performance & MCP

Operational quality includes permissions, escalation, speed, cost, reliability, and privacy - not only answer correctness.

Operational checks

Permissions
Least access needed; sensitive actions require correct scope and approval.

Escalation
Agent admits uncertainty, stops after defined failures, transfers useful context.

Speed
Record total and per-step latency; compare to a normal run.

Cost
Look for repeated calls, oversized context, needless retries, expensive paths.

Reliability
Consistent behavior, bounded retries, graceful timeout and fallback.

Privacy
Minimize sensitive data in tools/logs; follow incident policy.

Stop and escalate now: potential data exposure, unauthorized account changes, payment/refund harm, security abuse, repeated unsafe actions, or a high-volume production incident.

MCP in simple terms

MCP (Model Context Protocol) is a standard way for an AI application to connect to outside tools and information - like a common plug between the AI host and systems that expose capabilities.

MCP host
AI application
→
MCP client
connection component
→
MCP server
tools/resources
→
Connected system
CRM / DB / docs

MCP is not Claude, and it is not the CRM/database. It is the communication standard connecting an AI application to the server exposing those capabilities.

AI CUSTOMER SUCCESS / FIELD GUIDE

Role Boundaries & Handoffs

Own the investigation and customer context. Coordinate before changing production behavior outside your role.

Who owns what

Role	Typical ownership
AI Customer Success	Intake, context, expected behavior, accessible trace review, safe reproduction, impact/severity, examples/eval cases, issue reports, customer communication, fix validation.
AI Product Manager	Intended behavior, priorities, acceptance criteria, tradeoffs, rollout decisions.
AI / application engineer	Prompts/workflows/tools/MCP, code/schema changes, tracing instrumentation, technical fixes.
Security / privacy	Access policy, scopes, approvals, data handling, audit requirements, incident response.
Operations / SRE	Availability, monitoring, alerting, latency, capacity, incidents, rollback.
Data / knowledge owner	Source accuracy, freshness, taxonomy, and permissions.

A useful boundary test

You can usually do

Inspect approved conversation/trace data, reproduce in approved test space, write cases and expected outcomes, compare before/after, document findings.

Coordinate before doing

Change production prompts, code, tools/MCP servers, permissions, customer records, refund/payment logic, or monitoring policy.

Issue report: minimum useful packet

- ✓ One-line expected vs actual summary.
- ✓ Customer/account impact + scale.
- ✓ Severity and why.
- ✓ Environment/version + date/time.
- ✓ Conversation/request/trace IDs.
- ✓ Exact input + needed context only.
- ✓ Expected behavior + source of truth.
- ✓ Actual response/action.
- ✓ Minimal safe reproduction steps.
- ✓ First suspected failure point.
- ✓ Trace + comparison evidence.
- ✓ Risk/workaround + next owner.

AI CUSTOMER SUCCESS / FIELD GUIDE

Final Quick Reference

Use this page when a live report arrives and you need a fast debugging path.

When a report arrives

Expected → **Actual** → **Impact** → **Trace** → **First divergence** → **Reproduce** → **Classify + route**

If you see...

Symptom	First place to look
Wrong fact	Retrieved data, tool result, source freshness.
Wrong action	Tool choice, arguments, permissions, approval rules.
No action	Tool availability/description, missing input, permission denial, error.
Confident guess	Missing-data handling, uncertainty instructions, escalation.
Slow response	Per-step timing, repeated calls, retries, retrieval volume, model latency.
High cost	Context size, repeated steps, tool definitions, model choice.
Works sometimes	Version/environment, data conditions, nondeterminism, timeout, race.
Keeps looping	Stop condition, retry count, repeated result, fallback/escalation.

Beginner capability checklist

- ✓ Explain agent, trace, log, metric, and eval.
- ✓ Find a trace from a conversation/request ID.
- ✓ Narrate input → tools/context → output.
- ✓ Identify the first divergence.
- ✓ Separate model failure from upstream data/tool failure.
- ✓ Reproduce safely, one variable at a time.
- ✓ Create a compact regression eval set.
- ✓ Recognize privacy/security/financial escalation triggers.
- ✓ Check latency, retries, usage/cost, reliability.
- ✓ Explain MCP host/client/server/system.
- ✓ Route the fix to the correct owner.
- ✓ Validate the fix and close the customer loop.

Your job is not to know every implementation detail. It is to make the failure understandable: what should have happened, what actually happened, where the first divergence appeared, how severe it is, and what evidence shows the fix worked.