The Baton Pass: Context Management Between Sessions

Maintaining Continuity Across Claude Code Sessions

By Mark Kashef

Self-Improving AI Systems Package

Table of Contents

Introduction

When building complex systems with Claude Code, sessions end but projects don't. This guide shows you how to maintain continuity across sessions so you never lose momentum or context.

Related guides:

- *Architecture Guide* - System design and database schema
- *Rubric Template* - How to structure evaluation criteria
- *Quick Start* - Build your first system in 30 minutes

---

1. Why Context Management Matters

The Problem

Claude Code sessions have a context window. When you:

- Close the terminal
- Start a new conversation
- Hit context limits

...the AI loses all memory of what you built, what worked, what failed, and what's next.

Without context management, each session starts from zero:

- "What were we working on?"
- "Where's the code for X?"
- "Why did we decide to do it this way?"
- "What's left to build?"

The Solution

A **handoff document** is a structured markdown file that captures the state of your project at any point. It's like saving your game—you can resume exactly where you left off.

The Payoff

Without Handoff	With Handoff
10-15 min re-explaining context	"Refer to @HANDOFF.md and continue"
AI suggests already-tried approaches	AI knows what failed

Miss edge cases discovered earlier	Edge cases documented
Inconsistent architecture decisions	Design decisions preserved
Repeat debugging	Known bugs listed

---

2. The Handoff Document Structure

File Location and Naming

```
project-root/
■■■ HANDOFF.md          # &lt;-- Top level, easy to find
■■■ src/
■■■ package.json
■■■ ...
```

Name it `HANDOFF.md` (all caps for visibility) in your project root.

Template Structure

```
# Project Handoff Document

**Last Updated:** [Date and time]
**Last Session Summary:** [One sentence on what was accomplished]

---

## Current Phase

[What major milestone/phase is the project in?]

**Phase X of Y: [Phase Name]**

[2-3 sentences describing what this phase involves and its goals]

---

## Completed Items

### Phase 1: [Name]
- [x] Item 1
- [x] Item 2
- [x] Item 3

### Phase 2: [Name]
```

```
- [x] Item 1
- [x] Item 2
- [ ] Item 3 (in progress)

---

## In Progress

### Currently Working On
- [ ] [Specific task being worked on]

### Blocked By
- [Any blockers, if applicable]

---

## Known Bugs

| Bug | Severity | File(s) | Notes |
|-----|----------|---------|-------|
| Description | High/Med/Low | path/to/file.ts | Reproduction steps or context |

---

## Items to Investigate

- [ ] [Thing that needs research]
- [ ] [Performance issue to look into]
- [ ] [Alternative approach to consider]

---

## Database State

### Tables Created
- `table_name` - Description (status: populated/empty/schema-only)

### Migrations Pending
- [ ] Migration description

### Sample Data
- [Yes/No] - [How many records per table]

---

## Last Working State

**What was working at end of session:**
- Feature A: Working
- Feature B: Working with caveats (describe)
- Feature C: Not yet implemented

**How to verify:**
```

Commands to test the current state

npm run dev

Then navigate to localhost:3000/path

```
---

## Architecture Decisions

| Decision | Reasoning | Date |
|----------|-----------|------|
| Using X instead of Y | Because Z | YYYY-MM-DD |

---

## Files to Reference

Key files for continuing work:
- `path/to/important/file.ts` - What it does
- `path/to/another/file.ts` - What it does

---

## Next Session Should

1. First priority task
2. Second priority task
3. Third priority task

---

## Notes for Future Me

- Gotcha 1: [Something non-obvious]
- Gotcha 2: [Something you'll forget]
- [Any other context]
```

---

3. How to Use the Handoff Document

Starting a New Session

Begin your new Claude Code session with:

```
Refer to @HANDOFF.md and continue where we left off.
```

Or be more specific:

```
Refer to @HANDOFF.md. Let's work on the first item in &quot;Next Session Should&quot;.
```

During a Session

Update the handoff as you go, especially:

- When you complete something: Check it off
- When you discover a bug: Add it to Known Bugs
- When you make a decision: Add it to Architecture Decisions
- When you're interrupted: Update "Currently Working On"

Ending a Session

Before closing, ask Claude Code:

```
Update @HANDOFF.md with our progress. Include:
- What we completed
- Current state of what we were working on
- Any bugs discovered
- What should be done next
```

The "Quick Save" Pattern

If you need to stop suddenly:

```
Quick save to @HANDOFF.md - I need to stop.
```

This tells Claude Code to capture the essential state without being comprehensive.

---

4. Example Handoff Document from a Real Session

Here's an actual handoff from building a self-improving AI system:

```
# Project Handoff Document

**Last Updated:** 2024-01-15 3:45 PM
**Last Session Summary:** Completed reflection loop edge function and discovered UI bug in session si
```

Current Phase

Phase 3 of 5: Feedback Loop Implementation

Building the core self-improvement mechanism. This phase covers the reflection edge function, evaluation logic, and prompt update automation.

Completed Items

Phase 1: Foundation
- [x] Set up Next.js 14 with App Router
- [x] Configure Supabase MCP connection
- [x] Create database schema (users, sessions, messages)
- [x] Build basic chat UI

Phase 2: Chat Functionality
- [x] Implement chat-handler edge function
- [x] Add message persistence with prompt versioning
- [x] Create session management (new/switch/delete)
- [x] Add real-time message updates

Phase 3: Feedback Loop
- [x] Create system_prompts table with versioning
- [x] Build reflection_logs table
- [x] Implement reflection-loop edge function
- [x] Add rubric-based evaluation
- [ ] Create prompt-updater edge function
- [ ] Wire up automatic prompt updates

In Progress

Currently Working On
- [ ] prompt-updater edge function
  - Started the file at supabase/functions/prompt-updater/index.ts
  - Need to implement the improvement generation logic

Blocked By
- Nothing currently blocking

Known Bugs

Bug	Severity	File(s)	Notes
Session list doesn't refresh after new session created	Medium	SessionSidebar.tsx	Need to inva
Empty state shows briefly before messages load	Low	MessageList.tsx	Add loading skeleton
Reflection loop returns 500 if no messages in time window	High	reflection-loop/index.ts	Need

---

```
## Items to Investigate

- [ ] Should we add rate limiting to chat-handler?
- [ ] Look into Supabase edge function cold start times
- [ ] Consider adding message streaming instead of full responses

---

## Database State

### Tables Created
- `users` - Schema only (using test user ID for now)
- `sessions` - Populated (~5 test sessions)
- `messages` - Populated (~50 test messages)
- `system_prompts` - Populated (v1, v2, v3 - v3 is active)
- `reflection_logs` - Populated (3 test reflections)
- `suggestions` - Schema only (not yet used)

### Migrations Pending
- [ ] Add `locked` column to system_prompts for manual override
- [ ] Add index on messages(prompt_version)

### Sample Data
- Yes - Each session has 5-15 messages for testing

---

## Last Working State

**What was working at end of session:**
- Chat UI: Working - can send messages and receive responses
- Session management: Working - can create, switch, delete sessions
- Reflection loop: Working with caveat (500 error on empty result)
- Prompt versioning: Working - messages tagged with prompt version

**How to verify:**
```

cd frontend && npm run dev

Navigate to localhost:3000/chat

Send a message - should get response

Create new session - should work but sidebar won't auto-refresh

```
---

## Architecture Decisions

| Decision | Reasoning | Date |
|----------|-----------|------|
| Store prompt_version on each message | Enables analysis of which prompt produced which responses |
| Use 1-5 scoring scale | Granular enough for decisions, simple enough for AI to use consistently | 2
| Cooldown of 6 hours between reflections | Prevent rapid-fire updates while getting enough data | 20
| Auto-update threshold at 2.5 | Conservative - only bad scores trigger auto-update | 2024-01-15 |

---

## Files to Reference

Key files for continuing work:
- `supabase/functions/reflection-loop/index.ts` - The evaluation logic is here
- `supabase/functions/chat-handler/index.ts` - Reference for edge function patterns
- `frontend/src/lib/supabase.ts` - Database client setup
- `frontend/src/components/MessageList.tsx` - Where the bug is

---

## Next Session Should

1. Fix the 500 error in reflection-loop when no messages exist
2. Build the prompt-updater edge function
3. Wire up auto-update: reflection → prompt-updater
4. Fix the session sidebar refresh bug
5. Test complete feedback loop end-to-end

---

## Notes for Future Me

- The Anthropic API key is in Supabase edge function secrets, not .env
- To test reflection loop manually: `curl -X POST https://[project].supabase.co/functions/v1/reflecti
- The rubric is currently hardcoded in reflection-loop; should externalize later
- Don't forget: messages need `prompt_version` column populated for reflection to work
```

5. Advanced Patterns

The Multi-Branch Handoff

When exploring different approaches:

```
## Exploration Branches

### Branch A: Using Redis for caching
**Status:** Abandoned
```

```
**Why:** Cold start times unacceptable for edge functions
**Files:** Deleted, but pattern was in `/experiments/redis-cache/`

### Branch B: In-memory LRU cache (Current)
**Status:** Active
**Files:** `src/lib/cache.ts`
**Notes:** Working well, may need to tune size limits
```

The Handoff Checkpoint

Before major changes, create a checkpoint:

```
## Checkpoints

### Checkpoint: Pre-Auth-Refactor (2024-01-15)
Everything working with simple auth. Taking this checkpoint before
adding NextAuth complexity.

**To restore:**
```

git checkout abc123 # Commit hash

The Team Handoff

When multiple people work on the project:

```
## Handoff: [Your Name] → [Next Person]

**What I did:**
- Implemented X
- Fixed Y
- Started Z but didn't finish

**What you should know:**
- The auth token expires after 1 hour during testing
- I used a workaround in file.ts line 45 (TODO comment)
- The design doc is outdated; ignore section 3

**Questions for you:**
- Should we use approach A or B for the caching layer?
- Is the current error handling pattern okay?
```

---

6. Handoff Maintenance

Keep It Current

A stale handoff is worse than no handoff—it wastes time on outdated context.

Update triggers:

- Completing a task
- Discovering a bug
- Making a decision
- Ending a session
- Before any major change

Keep It Honest

Don't mark things complete that aren't. Don't hide bugs. The handoff is for you (or your future self).

Bad:

```
- [x] Implement user authentication
```

Good:

```
- [x] Implement user authentication
  - Note: Only email/password, no OAuth yet
  - Note: Password reset not implemented
```

Keep It Scannable

Your future self will skim, not read. Use:

- Headers liberally
- Checkboxes for status
- Tables for structured info
- Bold for emphasis

Quick Reference: Handoff Commands

When	Say This
Start session	"Refer to @HANDOFF.md and continue"
During session	"Update @HANDOFF.md with [specific thing]"

End session	"Update @HANDOFF.md with our progress"
Emergency stop	"Quick save to @HANDOFF.md"
Check status	"Summarize current state from @HANDOFF.md"
Resume specific task	"Refer to @HANDOFF.md, continue with [task from Next Session Should]"

This handoff system is part of the Self-Improving AI Systems package. See Architecture Guide for system design.