Eigenflow

GUIDE · RELAIS

Le handoff contre le context rot

La consigne à copier, le modèle de handoff.md en six rubriques, /clear puis la reprise, et notre vrai test face au compactage de Claude Code.

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté RELAIS, voici le guide. Le principe : avant d'arrêter une longue session de Claude Code, tu lui fais écrire un fichier `handoff.md` qui tient le travail en six rubriques. Puis tu repars d'une conversation vide qui lit ce fichier.

Tu trouves ici la consigne à copier, le modèle du fichier, la phrase de reprise, et les cas où ça ne sert à rien. Le 20 novembre 2026, on a relu la documentation de Claude Code et l'article d'Anthropic sur le sujet, puis on a essayé la méthode sur une vraie session à nous, à côté du compactage de Claude Code. Relu le 25 novembre 2026 : la documentation dit toujours la même chose sur tous ces points.

À savoir avant de lire

- Le handoff n'est pas une fonction de Claude Code. C'est un fichier que tu demandes à Claude d'écrire : rien à installer, rien à activer.
- Il faut Claude Code, donc un forfait Claude payant ou une clé de la Console. Le forfait gratuit ne l'inclut pas.
- Le fichier est écrit par Claude. Il peut se tromper ou dater déjà : relis-le avant de repartir.
- Il garde ce que contenait ta session, chemins et noms compris. N'y laisse ni mot de passe ni clé.
- Notre test porte sur une seule session, la nôtre, sur Opus 5.5. La reprise a tourné en mode plan, en lecture seule.

1. Le context rot en deux phrases

Plus le contexte d'un modèle s'allonge, moins il retrouve bien ce qu'il contient. Les impasses et les fausses pistes qui s'empilent dans la conversation aggravent la chose.

Le terme apparaît sur Hacker News le 18 juin 2025, dans un commentaire qui décrit déjà la parade : résumer, puis repartir d'une instance neuve qui lit le résumé. Chroma l'a mesuré le 14 juillet 2025 sur 18 modèles : leurs réponses deviennent moins fiables à mesure que l'entrée grandit, même sur des tâches simples. Anthropic reprend le terme le 29 septembre 2025.

> Despite their speed and ability to manage larger and larger volumes of data, we've observed that LLMs, like humans, lose focus or experience confusion at a certain point. Studies on needle-in-a-haystack style benchmarking have uncovered the concept of context rot: as the number of tokens in the context window increases, the model's ability to accurately recall information from that context decreases.
>
> While some models exhibit more gentle degradation than others, this characteristic emerges across all models. Context, therefore, must be treated as a finite resource with diminishing marginal returns. Like humans, who have limited working memory capacity, LLMs have an "attention budget" that they draw on when parsing large volumes of context. Every new token introduced depletes this budget by some amount, increasing the need to carefully curate the tokens available to the LLM.

L'article d'Anthropic du 29 septembre 2025 : le rappel baisse quand le nombre de tokens augmente (capture du 20 novembre 2026)

La baisse est progressive : Anthropic parle d'une pente. La doc de Claude Code rappelle aussi que la plupart des modèles actuels ont une fenêtre d'un million de tokens, compactée vers 967 000 : tu manques rarement de place. Elle conseille quand même de vider le contexte entre deux tâches, parce qu'un contexte concentré donne de meilleures réponses et coûte moins.

---

2. Ce que fait le compactage

Quand tu tapes `/compact`, ou quand la fenêtre se remplit, Claude Code remplace la conversation par un résumé structuré et tu continues dans la même conversation. D'après sa doc, le résumé garde tes demandes et les extraits de code importants, mais des consignes détaillées données au début peuvent se perdre. Anthropic ajoute qu'il garde les décisions, les bugs pas encore réglés et les détails d'implémentation, et qu'il jette les sorties d'outils en double.

Ensuite, Claude Code relit sur le disque le CLAUDE.md à la racine du projet, la mémoire automatique, le plan et jusqu'à cinq fichiers récents.



When a long session compacts, Claude Code summarizes the conversation history to fit the context window.

Mechanism	After compaction
System prompt and output style	Both still apply
Project-root CLAUDE.md and unscoped rules	Re-injected from disk
Auto memory	Re-injected from disk
Skill descriptions	Rebuilt from your current **skills** and re-injected. Requires Claude Code v2.1.304 or later
Git status snapshot	If the session started with one, Claude Code reads a fresh one from your repository
The plan Claude wrote in **plan mode**	Re-injected from disk
Rules with `paths:` frontmatter	Claude Code reloads them **on demand**
Nested CLAUDE.md in subdirectories	Claude Code reloads them **on demand**
Files Claude read or edited	Claude Code re-reads up to five, most recently modified first

La doc de Claude Code : ce qui est rechargé après un compactage (extrait du tableau, capture du 20 novembre 2026)

Le compactage arrive aussi sans que tu le demandes, dans deux cas :

- la fenêtre est presque pleine, vers 967 000 tokens sur Opus 5.5 ;
- tu laisses une session de plus de 200 000 tokens environ une heure sans rien envoyer, sur un abonnement. Claude Code la compacte juste avant que son cache expire, et à ton retour tu repars d'un résumé que tu n'as pas lu.

Il lui manque deux choses. Tu ne choisis ses rubriques qu'à travers une consigne. Et le résumé vit dans la conversation : impossible de l'ouvrir, de le corriger ou de le passer à quelqu'un comme un fichier. Le handoff règle les deux.

---

3. Quand faire un handoff, et quand c'est inutile

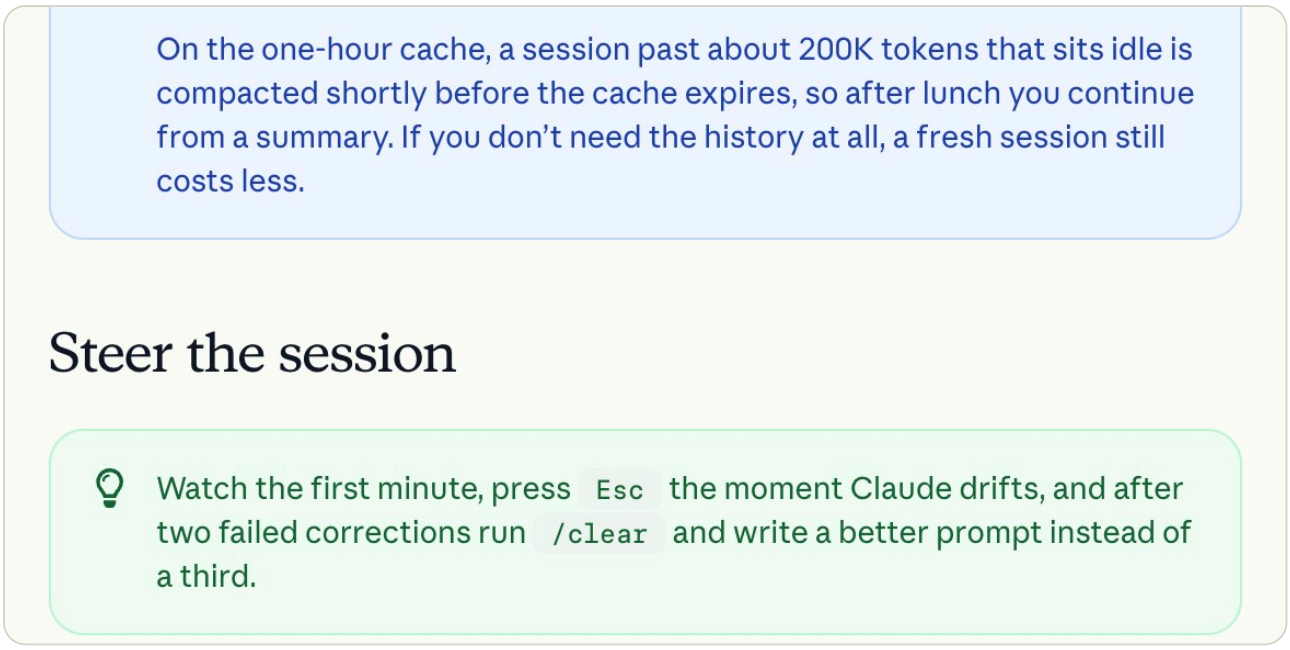
Fais-en un :

- avant de t'absenter plusieurs heures d'une session longue. Au-delà de 200 000 tokens, ton retour partira d'un résumé de toute façon, autant choisir ce qu'il dit ;
- quand Claude rate deux fois la même correction. D'après la doc, une troisième correction s'ajoute à une conversation qui contient déjà deux essais ratés, et `/clear` plus un nouveau prompt qui reprend ce que tu as appris marche mieux presque à chaque fois ;
- quand la tâche continue demain, sur une autre machine ou chez un collègue ;
- quand `/context` montre une conversation remplie de sorties de commandes et d'essais.

Il ne sert à rien :

- entre deux tâches sans rapport : `/clear` seul suffit, il n'y a rien à transmettre ;

- sur une session courte, ou quand tout est déjà dans les fichiers, les commits et le ticket ;
- pour une règle qui doit tenir dans toutes tes sessions, sa place est dans CLAUDE.md ;
- si tu veux retrouver la conversation elle-même : `/resume` la rouvre telle quelle.



La doc de Claude Code : la session compactée pendant la pause, et /clear après deux corrections ratées (capture du 20 novembre 2026)

---

4. La consigne à copier

Envoie-la dans la session à passer, avant de la quitter :

À COLLER

```
Avant qu'on s'arrête, crée un fichier handoff.md à la racine du projet avec six sections : 1) l'objectif, avec les règles que je t'ai données en cours de route, mot pour mot ; 2) l'état actuel : ce qui marche, ce qui est fait sans être vérifié, ce qui n'est pas commencé ; 3) les fichiers sur lesquels tu travailles ; 4) les changements que tu as faits ; 5) la liste des tentatives ratées, avec l'erreur exacte et pourquoi ; 6) les prochaines étapes précises, avec les commandes. Pas de mot de passe, pas de clé, pas de jeton.
```

La fin du point 1 vient de notre test : notre consigne ne l'avait pas, et une règle donnée au début de la conversation s'est perdue en route (partie 7).

---

5. Le modèle de handoff.md

Si tu préfères que Claude remplisse un cadre, colle ce modèle dans ton projet et demande-lui de le compléter.

À COLLER

```
# Handoff · <la tâche> · <date>

## 1. L'objectif
Ce qu'on veut obtenir, en deux phrases.
Les règles données en cours de route, mot pour mot.

## 2. L'état actuel
Ce qui marche, et comment on l'a vérifié.
Ce qui est fait mais pas vérifié.
Ce qui n'est pas commencé.

## 3. Les fichiers sur lesquels je travaille
- chemin/du/fichier : son rôle, ce qui reste à y faire

## 4. Les changements que j'ai faits
- fichier ou commande : ce qui a changé, le commit s'il existe

## 5. Les tentatives ratées
- ce qui a été essayé : l'erreur exacte, pourquoi ça a raté, ce qui a marché à
la place

## 6. Les prochaines étapes précises
1. la première action, avec la commande ou le fichier
2. comment vérifier que c'est bon
```

Ne saute pas la rubrique 5. C'est elle qui évite à la session neuve de refaire les mêmes essais.

---

6. /clear, puis la phrase de reprise

D'abord, relis le fichier. Deux minutes suffisent : corrige ce qui est faux, enlève ce qui ne doit pas circuler.

Ensuite, tape `/clear`. Claude Code ouvre une conversation neuve, au contexte vide. Ton CLAUDE.md et ta mémoire automatique (si elle est allumée, ce qui est le réglage par défaut) sont rechargés, comme au début de toute conversation. L'ancienne est gardée : `/resume` la rouvre. Tu peux lui donner un nom au passage, pour la retrouver dans la liste :

À COLLER

```
/clear facturation-avant-relais
```

Fermer Claude Code et le relancer dans le dossier revient presque au même.

Enfin, envoie la phrase de reprise :

À COLLER

```
Lis handoff.md, vérifie sur le disque que l'état décrit est toujours vrai, puis reprends là où on s'est arrêté. Commence par me dire ce qui a changé et ton plan.
```

Pour une reprise sans risque, lance la session neuve en mode plan : Claude lit et propose, sans rien modifier.

À COLLER

```
claude --permission-mode plan
```

`/clear [name]`	Start a new conversation with empty context. Pass a name to label the previous conversation in the `/resume` picker. To free up context while continuing the same conversation, use `/compact` instead. Resume the previous conversation with `/resume`, or, in the same Claude Code process, restore it from **the rewind menu's previous-session entry**. Aliases: `/reset`, `/new`

La doc de Claude Code : /clear ouvre une conversation neuve, /resume rouvre l'ancienne (capture du 20 novembre 2026)

---

7. Ce que notre test a montré

Le 20 novembre 2026 à 22 h 12, on a copié telle quelle une vraie session de Claude Code sur Opus 5.5, en plein travail, à 279 080 tokens, soit 28 % de sa fenêtre. Deux copies, dans un dossier à part, sans CLAUDE.md, sans hooks ni MCP.

- Copie A, la consigne en six sections : un `handoff.md` de 1 533 mots, écrit en 80 secondes environ. Six tentatives ratées, la plupart avec l'erreur exacte et ce qui a marché à la place, et neuf prochaines étapes avec leurs commandes.
- Copie B, `/compact` : 110 secondes et 0,34 dollar au prix de l'API, pris sur l'abonnement. Un résumé de 3 905 mots en neuf rubriques aux titres anglais, toujours les mêmes : Primary Request and Intent, Key Technical Concepts, Files and Code Sections, Errors and fixes, Problem Solving, All user messages, Pending Tasks, Current Work, Optional Next Step. Le contexte est passé de 279 080 à 9 483 tokens. Le résumé finit par le chemin de l'historique complet, au cas où un détail manquerait.

Deux surprises.

- Les impasses sont des deux côtés. Les erreurs de la soirée (un motif de fichiers mal protégé sous zsh, une boucle qui ne découpait pas ses mots, deux refus HTTP 429 d'une API de voix) figurent dans la rubrique Errors and fixes du compactage et dans les tentatives ratées

du handoff, chaque fois avec leur correction. Le compactage n'a pas traîné le débogage en vrac.
- Le compactage a gardé plus. Une règle de forme donnée dans le premier message, commencer chaque réponse par un mot précis, est dans son résumé et pas dans le handoff. La session neuve ne l'a pas appliquée. Dans notre test, CLAUDE.md était coupé ; chez toi, une règle écrite dans CLAUDE.md revient après `/clear`.

Copie C, la reprise. Une session neuve, sans historique, en mode plan, avec la lecture de fichiers pour seuls outils, a reçu « Lis handoff.md et reprends là où on s'est arrêté ».

- Contexte au premier message : 6 693 tokens. Chez toi il y en aura plus, avec CLAUDE.md, les outils MCP et les skills, et `/context` te dira combien.
- Après lecture du handoff : 12 115 tokens. Le fichier en pèse environ 5 200.
- Elle a relu le dossier et commencé par : « il est plus avancé que ce que décrit le handoff ». La vraie session avait continué pendant le test, et son plan en huit étapes partait de l'état réel.
- 70 secondes, 0,48 dollar au prix de l'API, pris sur l'abonnement.

Notre avis : le handoff est deux fois et demie plus court que le résumé du compactage, ses rubriques sont les tiennes, et tu le relis comme n'importe quel fichier. Le compactage garde davantage, règles comprises. Les deux restent des résumés écrits par Claude, aucun ne te ramène exactement où tu étais. D'où la vérification sur le disque dans la phrase de reprise.

Pas testé : `/clear` dans une session interactive (nos sessions neuves sont parties de `claude -p`), un autre modèle, une reprise plusieurs jours après, un projet d'équipe.

---

8. Les alternatives intégrées

Claude Code fait déjà une partie du travail. Voici quand chaque commande vaut mieux que le handoff.

- `/compact` avec une consigne : tu restes dans la même conversation, allégée. Exemple juste en dessous.
- `/rewind`, puis Summarize from here : seule la fin a mal tourné, un long débogage par exemple. Tes premières consignes restent intactes. Summarize up to here fait l'inverse. Ta consigne se tape sur la ligne « add context (optional) ».
- `/clear` seul : entre deux tâches sans rapport. Il ne coûte rien.
- `/resume`, ou `claude --continue` depuis le terminal : pour retrouver une conversation telle quelle. `/rename` lui donne un nom, `claude --resume` suivi de ce nom la rouvre plus tard.
- CLAUDE.md : pour les règles qui doivent survivre à tout. Il est relu après chaque compactage et chargé dans chaque nouvelle conversation. Une section « Compact instructions » y dit ce que le résumé doit garder.
- `/context` : pour voir ce qui remplit ta fenêtre avant de choisir.

À COLLER

```
/compact garde l'objectif, les fichiers modifiés, les erreurs déjà réglées et ma dernière demande mot pour mot
```

La doc décrit d'ailleurs un geste voisin du handoff : faire écrire la spécification dans `SPEC.md`, puis ouvrir une session neuve qui part de ce fichier.

---

9. Limites

- Un handoff écrit par Claude peut contenir une erreur ou une étape déjà faite. Dans notre test, le dossier avait avancé depuis, et la session neuve ne l'a vu qu'en le relisant. Demande-lui de vérifier, dans la phrase de reprise.
- Notre handoff contenait des chemins avec le nom du compte, des identifiants internes et le solde d'un service. Aucune clé chez nous, mais ce genre de fichier ne se publie pas, et un mot de passe n'a rien à y faire.
- Si ton dépôt est partagé ou public, range le fichier hors du dépôt, ou ajoute `handoff.md` à ton `.gitignore`.
- Écrire le handoff relit toute la conversation, comme `/compact`. Fais-le une fois, au bon moment.
- `/clear` ne touche pas à tes fichiers, mais il ne commite rien. Si le travail en vaut la peine, fais ton commit avant.

Sources

- Claude Code : les bonnes pratiques, les commandes, la fenêtre de contexte, le fonctionnement, rewind et résumé, le compactage à l'arrêt, CLAUDE.md et la mémoire, les sessions.
- Anthropic, Effective context engineering for AI agents, 29 septembre 2025.
- Chroma, Context Rot: How Increasing Input Tokens Impacts LLM Performance, 14 juillet 2025.
- Le premier usage du terme, sur Hacker News, 18 juin 2025.