

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=652d2cc3e10e468d9f11969266c0bacc-VS-V%C3%A9rifier-finale-s

<USER>

[Verify][SPECIF FINALISER LIVESESSION POUR LES SESSIONS PROMPT NORMALES 20260712 204731](#)

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧  scanning recent files

🔧  scanning git: 🗑️ ["État des modifications"](#), 🗑️ ["Inventaire des écarts"](#), 🗑️ ["Volume des écarts"](#)

🔧  reading project: 📁 g ["Docs liveSession proches"](#)

🔧  scanning git: 🗑️ ["Commit courant"](#), 🗑️ ["Détail du dernier commit"](#)

🔧  reading project: 📄 [liveSession.programming-doc-readme.md](#), 📄 [easyAgents.programming-doc-readme.md](#), 📄 [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#), 📄 [thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md](#), 📄 [prompt-stream-dispatch-lock.programming-doc-readme.md](#), 📄 [fallback-persist-and-title.programming-doc-readme.md](#)

J'ai lu les documents de programmation liveSession, EasyAgents, isomorphisme, thinking, Queue B et fallback persist. Ils décrivent bien les invariants de la spécification ; je vérifie maintenant que le code exécutable les matérialise réellement — les docs sont la carte, mais le code reste le territoire 🗺️.

🔧  reading project: 📄 h ["Cœur liveSession"](#), 📄 h ["Transport Prompt exact"](#), 📄 h ["Panneau et activité"](#), 📄 e ["Any explicites ajoutés"](#), 📄 e ["Désactivations ESLint"](#), 📄 e ["Marqueurs TODO"](#),

📄 r ["LiveSessionRegistryService"](#),

📄 r ["useAiStream"](#), 📄 r ["PromptLiveSessionPanelCoordinatorService"](#), 📄 [LiveSessionRegistry.service.ts](#) [L1-

L320], 📄 [LiveSessionRegistry.service.ts](#) [L321-L680], 📄 [LiveSessionRegistry.service.ts](#) [L681-

L1092],

📄 [NormalPromptLiveSessionSource.service.ts](#),

📄 [iaStreamHandlers.ts](#),

📄 [PromptLiveSessionPanelCoordinator.service.ts](#),

📄 [useAiStream.hook.ts](#),

📄 [usePromptLiveSession.hook.ts](#),

📄 [PromptLiveSessionBootstrap.adapter.ts](#),

📄 [PromptLiveSessionPanelQuery.util.ts](#),

📄 [stream.endpoint.ts](#),

📄 [LiveSessionSourceLifecycle.service.ts](#),

📄 [liveSessionStreamHandlers.ts](#),

📄 [PromptLiveSessionDetachedStop.service.ts](#),

📄 [SidebarQueryRunningBridge.ts](#),

📄 [Prompt.tsx](#),

📄 [applyHydratedPromptSessionState.util.ts](#),

📄 [LiveSessionPanel.type.ts](#),

📄 [LiveSessionOpenResult.type.ts](#),

📄 [parseSlimbkStreamBodyError.util.ts](#),

📄 [usePromptSessionLocator.hook.ts](#),

📄 [NormalPromptLiveSession.adapter.ts](#),

📄 [PromptLiveSession.types.ts](#),

📄 [usePromptRunCommands.hook.ts](#),

📄 [usePromptRunController.finalization.hook.ts](#),

📄 [openPanel.util.ts](#),

📄 [delete.endpoint.ts](#),

📄 [EasyAgentsBatchRunRetry.service.ts](#),

📄 [SidebarStore.queryWindow.listeners.manager.ts](#),

📄 [SidebarStore.queryWindow.mutations.manager.ts](#),

📄 r ["preparePromptContinuationStreaming"](#),

🔗 r "[NormalPromptLiveSessionSourceService](#)", 🔗 r "[usePromptRouterQueryStringRawSnapshot](#)", 🔗 e "[Casts double ajoutés](#)"

🔧 ☒ scanning git: 📄 "[Diff commandes Prompt](#)", 📄 "[Diff endpoint source](#)", 📄 "[Diff bootstrap Prompt](#)"

🔧 ☒ reading project: 📄 [LiveSessionRegistry.service.ts](#) [L1090-L1126], 🔗 e "[Échec priming source](#)", 🔗 e "[Service source dans endpoint](#)", 🔗 e "[Helper continuation utilisé](#)", 🔗 e "[Snapshot brut utilisé](#)"

🔧 ☒ scanning git: 📄 "[Fichiers générés du lot](#)", 📄 "[Écart document interdit](#)"

🔧 ☒ reading project: 📄 [delete.endpoint.liveSession.test.ts](#) [L1-L180]

1) La spécification entière est-elle implémentée ?

❌ **Réponse : non — estimation 84 %**

L'implémentation est **très étendue** et couvre la majorité de l'infrastructure demandée :

- contrats `sourceId`, replay et descripteurs stricts ;
- registre avec réservations, époques, retraites, tombstones et claims destructifs ;
- replay et Open exacts ;
- origine webview extension-only ;
- coordination des panneaux ;
- Activité ouverte/fermée et Stop détaché ;
- démarrage frontend transactionnel ;
- `preserveLocal`, handoff direct et protection par `streamInstanceId` ;
- documentation technique et importante couverture de tests.

Cependant, plusieurs éléments fondamentaux de la spécification existent **sans être reliés au chemin de production** :

1. [NormalPromptLiveSessionSource.service.ts](#) n'a **aucune référence de production**. Il est uniquement utilisé par ses tests.
2. [preparePromptContinuationStreaming.util.ts](#) n'est également utilisé que par des tests.
3. `Prompt` n'utilise pas le hook de snapshot brut figé créé dans [usePromptSessionLocator.hook.ts](#).
4. Le commit a modifié un fichier dans la zone documentaire personnelle explicitement interdite et hors spécification.
La présence des classes, helpers et tests demandés ne suffit donc pas : plusieurs pièces sont bien fabriquées, mais certaines sont encore posées à côté du moteur plutôt que branchées dessus. Le classique « excellente pièce détachée, voiture toujours sur trois cylindres » 🚗.

2) L'implémentation fait-elle ce qu'elle doit faire conformément à la spécification ?

⚠️ **Réponse : partiellement — estimation fonctionnelle 78 %**

Fonctionnalité technique demandée	État	Évaluation fondée sur le code
Identité exacte { <code>locator</code> , <code>turnId</code> , <code>sourceId</code> }	☒ Complet	Les contrats, clés de run, endpoints et messages transporter bien <code>sourceId</code> . LiveSessionPanel.type.ts utilise une union stricte, interdit les champs propres au mauvais adaptateur et impose <code>sourceId === turnId</code> pour <code>EasyAgents</code> .
Descripteurs stricts <code>EasyAgents/normalPrompt</code>	☒ Complet	Les branches <code>live</code> , <code>terminal</code> , <code>not_ready</code> et <code>not_found</code> ainsi que les propriétés présentes à <code>undefined</code> , sont contrôlées dans LiveSessionOpenResult.type.ts .
Résultats Open stricts et enveloppe sélective	☒ Complet	iaStreamHandlers.ts réserve avant le premier <code>await</code> .
Réservation avant les opérations asynchrones	☒ Présent	La logique existe dans les services source/lifecycle, mais le chemin normal de production promeut directement le record avant l'appel <code>Slimbk</code> .
Promotion après priming lazy du <code>Readable</code>	❌ Non intégré	<code>EasyAgents</code> utilise le lifecycle partagé, mais <code>Prompt normal</code> ne passe pas par <code>NormalPromptLiveSessionSourceService</code> .
Lifecycle partagé entre <code>EasyAgents</code> et <code>Prompt normal</code>	⚠️ Partiel	Le handler extension capture une baseline avant un appel réseau vers l'endpoint ; l'endpoint appelle ensuite directement <code>askStream</code> . Ce n'est pas la barrière atomique spécifiée.
Capture de baseline immédiatement avant <code>askStream</code>	❌ Non conforme	

Fonctionnalité technique demandée	État	Évaluation fondée sur le code
Validation de l'assistant exact pour <code>continueResponse</code>	✗ Non garantie en production	La validation existe dans le service source inutilisé, mais pas dans le chemin réellement exécuté.
Persistance de chaque <code>meta.error</code> avant <code>append</code>	✓ Présente dans le handler actif	Le handler appelle <code>patchLastAssistantMeta</code> avant <code>appendChunk</code> .
Protocole V1 du premier <code>meta.error</code>	✓ Complet	Le premier terminal d'erreur peut être transmis dans le header, décodé puis remplacé dans la FIFO.
Erreur Slimbk post-header filtrée avant les chunks	✓ Complet	parseSlimbkStreamBodyError.util.ts est utilisé avant le cast vers <code>AiStreamedChunk</code> .
FIFO extension et finalisation après drainage	✓ Globalement complet	Les handlers attendent la FIFO avant la finalisation du registre et du runtime Activité.
Finalisation à cinq résultats exacts	✓ Complet	Le registre implémente les retraites <code>baseline/terminal</code> , <code>handoffReady</code> , annulation et <code>stale</code> .
Buffer complet, ordonné et sans limite ajoutée	✓ Complet	Le registre conserve les chunks dans un tableau complet et sert des suffixes par curseur.
Époques, tombstones et générations destructives	✓ Complet	Les trois concepts sont distincts et les Open revalident époque et génération destructive.
Claims destructifs pour <code>Delete/Retry</code>	✓ Majoritairement complet	Les handles exacts sont acquis, utilisés pour l'éviction, puis libérés en <code>finally</code> .
Exactitude de la webview d'origine	✓ Complet	Le message bus transmet l'origine <code>extension-only</code> et les handlers comparent la webview exacte au panneau courant.
Panneau ouvert/fermé et continuation en extension	✓ Complet sur le chemin principal	Le coordinateur détache le panneau sans supprimer une source active et la projection Activité peut rester fermée/running.
Activité Idle/open et Running/source/closed	✓ Complet	Les unions et transitions principales sont présentes.
Stop détaché uniquement pour source fermée	✓ Complet	Le service vérifie la projection exacte, réclame l'abort, marque <code>stopRequested</code> , puis abort sans <code>await</code> .
Stop unique dans le compositeur reconnecté	✓ Complet	L'adaptateur <code>normalPrompt</code> ne demande aucun Stop dans la bannière.
Son déduplicé par run et tentative	✓ Implémenté	Le manager reçoit <code>runRef</code> et <code>soundAttemptId</code> exacts.
Démarrage frontend synchrone avec refus sans mutation	✓ Complet	useAiStream.hook.ts utilise une ref single-flight synchrone et retourne l'union d'acceptation.
Guards <code>sourceId + streamInstanceId</code>	✓ Complet sur les callbacks principaux	Les callbacks réseau et continuations après drainage revalident l'identité active.
Queue A, Queue B et commit avant outcome	✓ Complet	<code>completed</code> et <code>transportError</code> sont publiés après les deux queues et la barrière de commit.
Terminal data suivi d'une erreur transport	✓ Complet	Aucun second terminal n'est injecté lorsque <code>terminalDataSeenRef</code> est déjà vrai.
<code>replayOpenAttemptId</code> et protection des réponses Open stale	✓ Complet	Chaque Open reçoit une identité indépendante et les réponses tardives sont ignorées.
Retry <code>preserveLocal</code> sans lecture du brouillon	✓ Complet	La branche construit un objet assaini sans lire <code>promptDraft</code> les attachments de brouillon ni <code>nextTurnId</code> .
Handoff direct <code>normalPrompt</code>	✓ Implémenté	Le mode passe directement à <code>normal</code> après terminal, outcome complet et finalisation frontend correspondante.
Snapshot brut figé de query	✗ Non conforme	Le hook dédié existe mais Prompt.tsx reconstruit lui-même le snapshot avec des hooks scalaires et des dépendances

Fonctionnalité technique demandée	État	Évaluation fondée sur le code
Frontière React externe pour query invalide	✅ Complet	variables. Une query invalide ne monte ni contrôleur, ni hydratation, ni auto-run.
Désactivation de l'auto-run pour toute query live	✅ Comportement principal complet	L'effet auto-run est neutralisé, même si les hooks scalaires auto-run restent montés.
Plans purs pour les sept commandes	⚠️ Partiel	Send, SPECIF, Retry et Edit suivent bien l'acceptation avant mutation ; Continue duplique encore manuellement sa préparation au lieu d'utiliser le helper pur prévu.
Continuation/thinking factorisée	❌ Non intégrée	Le helper créé n'est pas appelé par le code de production.
Barrière fallback {turnId, streamInstanceId}	✅ Complet	Le clear intervient sur ack exact ; la finalisation sans ack démasque sans effacer les données locales.
Suppression durable sans disparition invisible sur échec	❌ Bug	L'activité est retirée avant la suppression disque et n'est pas reconstruite si celle-ci échoue.
Mise à jour des programming docs	✅ Complet	Les six documents techniques demandés ont été mis à jour.
Tests de toutes les courses majeures	⚠️ Large mais insuffisant	Beaucoup de tests existent, mais certains valident des helpers/services isolés qui ne sont pas utilisés en production.
Conclusion de la question 2		
Le chemin nominal fermeture → poursuite → Activité → réouverture → replay est largement construit , mais la source Prompt normale n'utilise pas l'architecture source/lifecycle spécifiée et plusieurs garanties testées ne sont donc pas réellement garanties par le chemin exécuté.		

3) Y a-t-il des bugs ?

❌ **Réponse : oui — 4 anomalies principales, dont 2 à risque élevé**

● **Bug 1 — Le service source normalPrompt est du code mort**

Sévérité : élevée

`NormalPromptLiveSessionSource.service.ts` n'est référencé par aucun module de production.

À la place :

- `iaStreamHandlers.ts` réserve **et promeut** directement le run ;
- `stream.endpoint.ts` appelle directement `orchAsk.stream.askStream(...)` ;
- la persistance, l'append, les baselines et les terminaux sont partiellement réimplémentés dans le handler.

Conséquences :

- absence du priming lazy avant promotion ;
- capture de baseline au mauvais niveau ;
- validation `continueResponse` non appliquée en production ;
- divergence possible entre les tests du service et le comportement réel ;
- duplication de logique terminale et de persistance.

C'est le principal écart avec la spécification.

● **Bug 2 — Une suppression disque échouée peut faire disparaître une source active de l'Activité**

Sévérité : élevée

Dans `delete.endpoint.ts`, l'ordre est :

1. fermer le panneau ;
2. retirer l'activité ;
3. tenter la suppression disque ;
4. évincer le registre uniquement en cas de succès.

Si la suppression disque échoue :

- la source n'est correctement **pas évincée** ;
- mais sa projection Activité a déjà été retirée ;
- la libération du claim ne la reconstruit pas ;
- la source peut donc rester active, panneau fermé, sans item Activité.

Le test correspondant vérifie l'absence d'éviction et la libération du claim, mais pas la restauration de l'activité.

Cela viole directement l'invariant « aucune activité invisible ».

🔴 Bug 3 — Le snapshot brut Prompt n'est pas réellement possédé par le hook figé prévu

Sévérité : moyenne à élevée

Le hook `usePromptRouterQueryStringRawSnapshot()` est correctement défini avec un `useMemo(..., [])`, mais il n'est utilisé que par son test.

`Prompt.tsx` :

- appelle d'abord deux hooks scalaires ;
- relit directement la globale ;
- reconstruit une query de secours qui ne contient que le locator ;
- recalcule le snapshot si les scalaires changent.

Lorsque la globale brute n'est pas disponible, cette reconstruction peut perdre :

- les occurrences dupliquées ;
- les paramètres live privés ;
- les paramètres inconnus ;
- l'ordre brut nécessaire à la classification stricte.

Le runtime normal fournit probablement la globale dans le cas habituel, mais le contrat strict prévu par la spécification n'est pas garanti par le composant.

🔴 Bug 4 — La préparation pure de continuation est testée, mais ignorée par la production

Sévérité : moyenne

`preparePromptContinuationStreaming.util.ts` n'est référencé que par des tests.

Le véritable chemin Continue dans `usePromptRunCommands.hook.ts` :

- recherche une première fois l'assistant pour le plan ;
- recommence après acceptation ;
- reconstruit manuellement le placeholder thinking ;
- modifie les refs et messages directement ;
- utilise un double cast `as unknown as PromptMessage`.

Le helper et la production peuvent donc diverger sans qu'un test du helper ne détecte la régression réelle.

🟡 Défaut latent supplémentaire dans le service source inutilisé

Si le priming échoue avant `onSourceStarted`,

le `catch` de `NormalPromptLiveSessionSourceService.execute()` essaie de persister et d'appendre un terminal alors que la réservation n'a pas été promue en record.

L'append peut alors lever « run non enregistré » et masquer l'erreur primaire. Ce défaut deviendra actif dès que le service sera correctement branché.

⚠ Écart de périmètre

Le commit modifie également un fichier personnel dans la zone explicitement interdite, alors que ce fichier n'appartient pas à la spécification. Je n'en ai pas chargé le contenu.

Les fichiers générés présents dans le commit ne sont pas signalés comme bugs : leurs changements semblent cohérents avec une régénération des contrats Slimbk, et l'examen Git seul ne permet pas d'affirmer qu'ils ont été édités manuellement.

4) Quelle est mon opinion sur le code source et la spécification ?

Sur la spécification : 93 % — excellente

La spécification est exceptionnellement rigoureuse sur :

- les identités exactes ;
- les barrières temporelles ;
- les erreurs post-header ;
- les courses ABA ;
- les claims destructifs ;
- le comportement terminal ;
- l'isomorphisme streaming/hydratation ;
- les responsabilités extension/frontend ;
- les scénarios de test.

Son facteur de risque annoncé à **92 %** était réaliste. C'est une très bonne spécification d'architecture concurrente, Joel — rares sont les specs qui identifient aussi précisément les frontières de propriété et les courses avant implémentation. 🙌

Son défaut principal est son volume atomique : **160 fichiers** rendent très difficile la vérification qu'un helper créé est effectivement utilisé par le chemin produit.

Sur le code : 80 % — solide, mais pas finalisé

✅ **Points forts**

- typage généralement strict ;
- contrats Zod bien structurés ;
- registre cohérent et défensif ;
- séparation nette des époques et générations destructives ;
- nombreux commentaires d'invariants ;
- protections exactes par webview, runRef et stream instance ;
- tests nombreux et ciblés ;
- coordination panel/Activity globalement lisible ;
- implémentation `preserveLocal` particulièrement propre.

❌ **Points faibles**

- deux abstractions centrales sont du code mort ;
- duplication importante entre endpoint, handler et service source ;
- plusieurs tests donnent une confiance trompeuse parce qu'ils testent des composants non branchés ;
- la continuation conserve une implémentation manuelle parallèle au helper prévu ;
- certains doubles casts ajoutés sont contraires au niveau de typage demandé ;
- un chemin destructif d'échec laisse une projection incohérente.

Avis global

L'architecture est **bonne et récupérable sans refonte**, mais le commit ressemble davantage à une implémentation très avancée à laquelle il manque une passe finale d'intégration qu'à une tranche totalement terminée.

5) Résumé du rapport

Axe vérifié	Estimation	Statut	Conclusion
Présence des fichiers et contrats prévus	96 %	✅	Presque tous les artefacts demandés existent.
Conformité architecturale	79 %	⚠️	Le chemin source <code>normalPrompt</code> contourne son service dédié.
Registre/replay/claims	91 %	✅	Partie la plus solide de l'implémentation.
Transport et protocoles d'erreur	86 %	✅	V1 et enveloppes post-header sont correctement traités.
Panneaux, Activité et Stop	84 %	⚠️	Chemin nominal bon, mais suppression échouée incohérente.
Frontend transactionnel	87 %	✅	Accept/refuse et barrières stream sont bien réalisés.
Bootstrap query strict	70 %	⚠️	Dispatcher correct, ownership du snapshot brut incomplet.
Continuation/thinking	72 %	⚠️	Helper pur présent, mais production encore dupliquée.
Tests	88 %	⚠️	Très larges, mais quelques tests couvrent du code mort plutôt que le câblage réel.
Respect du périmètre	95 %	⚠️	Une modification hors spécification dans la zone personnelle interdite.
Implémentation globale	84 %	❌	Pas entièrement conforme ni totalement finalisée.
Confiance fonctionnelle actuelle	78 %	⚠️	Le scénario principal peut fonctionner, mais des garanties importantes restent absentes.

6) Conclusion en une phrase

La tranche est très largement construite et possède de solides fondations, mais elle ne peut pas être considérée comme terminée tant que la source `Prompt normale`, le `snapshot brut`, la `continuation pure` et le `rollback d'Activité sur échec de suppression` ne sont pas correctement intégrés.

7) Corrections proposées

Lot A — Corrections backend critiques

- Brancher réellement [NormalPromptLiveSessionSource.service.ts](#) dans l'endpoint source.
- Supprimer la duplication `source/baseline/persistence` dans le handler.

- Promouvoir seulement après priming lazy.
- Corriger l'échec de priming avant promotion.
- Restaurer/reconstruire la projection Activité lorsque la suppression durable échoue.
- Ajouter les tests d'intégration prouvant que le service source est réellement traversé.

Lot B — Corrections frontend contractuelles

- Utiliser le snapshot brut figé de [usePromptSessionLocator.hook.ts](#) dans [Prompt.tsx](#).
- Retirer la reconstruction scalaire de secours du bootstrap strict.
- Brancher le helper pur de continuation dans [usePromptRunCommands.hook.ts](#).
- Supprimer les doubles casts ajoutés dans cette zone.
- Transformer les tests isolés en tests de câblage production.

Lot C — Corrections complètes recommandées

- Appliquer les lots A et B.
- Vérifier les appels de production des nouveaux services/helpers afin qu'aucun artefact central ne reste uniquement référencé par ses tests.
- Renforcer les scénarios suppression échouée, priming échoué, continuation acceptée/refusée et query brute dupliquée.
- Ne toucher à aucun fichier personnel hors périmètre.



Coder le lot A backend critique



Coder le lot B frontend contractuel



Coder toutes les corrections

</ASSISTANT>

<USER>

[Coder toutes les corrections]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>



✓ scanning git: ʘ "État avant corrections"



✓ reading

project: [NormalPromptLiveSessionSource.service.ts](#), [LiveSessionSourceLifecycle.service.ts](#), [stream.endpoint.ts](#),

[iaStreamHandlers.ts](#),

[delete.endpoint.ts](#),

[preparePromptContinuationStreaming.util.ts](#), [usePromptRunCommands.hook.ts](#), [Prompt.tsx](#) [L1-

L260], [usePromptSessionLocator.hook.ts](#) [L250-L330], [LiveSessionService.ts](#), [LiveSessionRecord.type.ts](#) [L1-

L300], [LiveSessionRegistry.service.ts](#) [L40-

L260],

[stream.endpoint.test.ts](#),

[NormalPromptLiveSessionSource.service.test.ts](#),

[iaStreamHandlers.runIdPropagation.test.ts](#),

[iaStreamHandlers.queryDoneSound.test.ts](#), [delete.endpoint.liveSession.test.ts](#), e "Appels endpoint

stream", e "Args client stream", [callLocalApi.ai.orch.query.ask.generated.ts](#) [L1-

L220],

[AiStreamedChunk.type.ts](#),

[promptHistoryTypes.type.ts](#),

[usePromptRunCommands.continueResponse.test.tsx](#),

[usePromptRunCommands.startAcceptance.test.tsx](#),

[Prompt.liveSessionBootstrap.test.tsx](#),

[usePromptSessionLocator.hook.test.tsx](#), [callLocalApi.ai.orch.query.ask.generated.ts](#) [L850-

L985], [callLocalApi.ai.orch.query.ask.generated.ts](#) [L985-

L1115], [callLocalApi.ai.orch.query.ask.generated.ts](#) [L1115-

L1185],

[SidebarStore.rehydrate.manager.ts](#),

[SidebarStore.core.manager.ts](#),

[SidebarStore.queryWindow.mutations.manager.ts](#), [SidebarStore.handlers.manager.ts](#), e "Réhydratation

[activités](#)",  [PromptLiveSessionPanelCoordinator.service.ts](#) [L1-L320],  [SidebarStore.queryWindow.listeners.manager.ts](#) [L1-L260],  [SidebarQueryRunningBridge.ts](#) [L1-L300],  e ["Hooks query Prompt"](#),  e ["Mock query snapshot Prompt"](#),  e ["Router global tests Prompt"](#),  e ["Mock useQueryString"](#),  [Prompt.autoRunPrompt.test.tsx](#) [L1-L220],  [Prompt.fallbackPersistAck.clearInput.test.tsx](#) [L1-L180],  e ["Imports composant Prompt"](#),  e ["Import statique Prompt"](#),  e ["Mocks locator Prompt"](#),  e ["SessionId mock Prompt"](#)

  scanning git:  ["Casts ajoutées finalizer"](#),  ["Casts ajoutées commandes"](#)

  reading project:  e ["Doubles casts"](#)

[liveSession"](#),

 [usePromptRunCommands.compactContextNow.test.tsx](#),

 [usePromptRunCommands.retryQuery.attachmentOnly.test.tsx](#),

 [usePromptRunController.liveSessionObserver.test.tsx](#),

 [LiveSessionSourceLifecycle.service.test.ts](#),  [OrchAsk.ts](#) [L1-L180],  e ["Méthode patch"](#)

[mémoire"](#),  [Prompt.autoRunPrompt.test.tsx](#) [L300-L390],  [Prompt.promptDraft.hydratation.test.tsx](#) [L285-

L330],  [Prompt.editLastPrompt.noThumbTurnIdDrift.test.tsx](#) [L125-

L165],  [Prompt.editLastPrompt.restoreAttachments.test.tsx](#) [L90-

L125],  [Prompt.ensureNextTurnId.stability.test.tsx](#) [L90-L120],  [Prompt.autoRunPrompt.test.tsx](#) [L390-

L681],  [PromptLiveSessionBootstrap.adapter.ts](#) [L1-L260],  e ["RenderHook"](#)

[usage"](#),  [usePromptThinkingLifecycle.continuation.test.tsx](#),  e ["API notices"](#)

[outil"](#),  [usePromptToolCallNotice.hook.ts](#) [L455-L535],  [LiveSessionRun.type.ts](#) [L1-L180],  g ["Tests handler IA stream"](#)

  patching file:  [NormalPromptLiveSessionSource.service.ts](#)

[AICode/src/extension/src/aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.ts](#)

Error

([what is it?](#))

  patching file:  [stream.endpoint.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/stream.endpoint.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { loadAiCatalog } from
'../../../../../../../../lib/catalogUtil/crud/loadAiCatalog.util.js';
- import { AiCatalog } from
'../../../../../../../../sharedlib/types/catalogTypes/AiCatalog.type.js';
- import { AiStreamedChunk } from
'../../../../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
-
+ import { loadAiCatalog } from
'../../../../../../../../lib/catalogUtil/crud/loadAiCatalog.util.js';
+ import type { AiCatalog } from
'../../../../../../../../sharedlib/types/catalogTypes/AiCatalog.type.js';
+ import type { AiStreamedChunk } from
'../../../../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+
- import { ZChatStorageScopeRelPath } from
'../../../../../../../../sharedlib/types/chatTypes/ChatSessionLocator.type.js';
- import { ZLiveSessionSourceId } from
'../../../../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
-
+ import { ZChatStorageScopeRelPath } from
'../../../../../../../../sharedlib/types/chatTypes/ChatSessionLocator.type.js';
+ import {
+   type LiveSessionRunRef,
+   ZLiveSessionSourceId,
+ } from '../../../../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
```


+

```
- import { encodeLiveSessionSourceErrorV1 } from
'../../../../../../../../sharedlib/types/aiTypes/aiStopSuffix.util.js';
-
+ import { encodeLiveSessionSourceErrorV1 } from
'../../../../../../../../sharedlib/types/aiTypes/aiStopSuffix.util.js';
+ import { getLiveSessionServiceForProd } from
'../../../../../../../../aiTools/orch/liveSession/core/LiveSessionService.js';
+ import { LiveSessionSourceLifecycleService } from
'../../../../../../../../aiTools/orch/liveSession/source/LiveSessionSourceLifecycle.service.js'
;
+ import {
+   NormalPromptLiveSessionSourceService,
+   takeNormalPromptLiveSessionReservation,
+ } from
'../../../../../../../../aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.js';
+
+ /** Parse one raw Orch value while preserving the endpoint's historical malformed-JSON
diagnostic. */
+ function parseOrchStreamChunk(raw: string): AiStreamedChunk {
+   try {
+     return JSON.parse(raw) as AiStreamedChunk;
+   } catch {
+     throw new Error(
+       `[LOCAL] Malformed stream chunk (not JSON): ${JSON.stringify(raw)}`,
+     );
+   }
+ }
```

```
-
-   // Invoke the LLM streaming engine
-   const readable = orchAsk.stream.askStream({
-     aiCatalog,
-     turnId: params.args.turnId,
-     threadId: params.args.threadId,
-     storageScopeRelPath: params.args.storageScopeRelPath,
-     aiQuery: params.args.aiQuery,
-     aiQueryOptions: params.args.aiQueryOptions,
-   });
-
-   // Loop through each streamed chunk
-   let index = 0;
-
+   const runRef: LiveSessionRunRef = {
+     locator: {
+       sessionId: params.args.threadId,
+       storageScopeRelPath: params.args.storageScopeRelPath,
+     },
+     turnId: params.args.turnId,
+     sourceId: params.args.sourceId,
+   };
+
+   // Consume the process-local capability only after all pre-source awaits
completed successfully.
+   const reservation = takeNormalPromptLiveSessionReservation(runRef);
+   const source = new NormalPromptLiveSessionSourceService({
+     liveSession: getLiveSessionServiceForProd(),
+     lifecycle: new LiveSessionSourceLifecycleService(),
+     orchStream: orchAsk.stream,
+     parseChunk: parseOrchStreamChunk,
```

```

+         });
+
+         // The endpoint owns only Slimbk framing; the source service owns priming,
durability, and registry append.
+         let headerSent = false;
+
-         let funStreamChunk: FunStreamChunk | undefined = undefined;
-         for await (const chunk of readable) {
-             // First chunk must be parsed to validate it's not an error
-             if (index === 0) {
-                 index++;
-
-                 let parsed: AiStreamedChunk | undefined;
-                 try {
-                     parsed = JSON.parse(chunk);
-                 } catch {
-                     throw new Error(
-                         `[LOCAL] Malformed stream chunk (not JSON): $
{JSON.stringify(chunk)}`,
-                     );
-                 }
-
-                 /**
-                  * If the very first chunk is a meta error coming from the orchestrator
/ LLM
-                  * provider, we must convert it into a SlimbkResult error header
instead of
-                  * throwing here.
-                  *
-                  * Rationale:
-                  * - callSlimbkStream() on the client side expects the first line of
the
-                  * response to be a SlimbkResult header.
-                  * - If we throw before calling funSendStreamResult(), no header is
ever sent,
-                  * the stream ends empty, and the client reports:
-                  * "Error: Stream ended before header"
-                  * which hides the real Bedrock/Anthropic error message.
-                  *
-                  * By sending a 500_SERVER_ERROR header with the provider message, the
client
-                  * receives a proper SlimbkResult, and the error surfaces as:
-                  * "500_SERVER_ERROR <Bedrock Converse API error ...>"
-                  */
-                 if (
-                     parsed?.type === 'meta' &&
-                     parsed?.value?.metaType === 'error'
-                 ) {
-                     // Only the first source error uses the private header protocol.
-                     funSendStreamResult(
-                         encodeLiveSessionSourceErrorV1(parsed.value.value),
-                     );
-                     // Stream handler terminates here: header has been sent and no
-                     // chunk body needs to be forwarded for this failure case.
-                     return;
-                 }
-
-                 // If valid chunk (token or non-error meta), send Slimbk 200_OK
-                 funStreamChunk = funSendStreamResult({
-                     status: '200_OK',
-                     response: 'streaming',
-                 });
-

```

```

-         funStreamChunk(chunk); // forward first chunk
-     } else {
-         if (!funStreamChunk) {
-             throw new Error('Internal error: stream not initialized');
-         }
-         // After first chunk, just forward everything
-         funStreamChunk(chunk);
-     }
- }
-
+ let funStreamChunk: FunStreamChunk | undefined = undefined;
+ await source.execute({
+     reservation,
+     runRef,
+     aiCatalog,
+     aiQuery: params.args.aiQuery,
+     aiQueryOptions: params.args.aiQueryOptions,
+     onChunk: async (chunk) => {
+         const encodedChunk = JSON.stringify(chunk);
+
+         // Only the first durable source error is encoded in the private
non-200 header.
+         if (!headerSent) {
+             headerSent = true;
+             if (
+                 chunk.type === 'meta' &&
+                 chunk.value.metaType === 'error'
+             ) {
+                 funSendStreamResult(
+                     encodeLiveSessionSourceErrorV1(chunk.value.value),
+                 );
+                 return;
+             }
+             funStreamChunk = funSendStreamResult({
+                 status: '200_OK',
+                 response: 'streaming',
+             });
+         }
+
+         // A terminal error header cannot be followed by an unrelated body
chunk.
+         if (!funStreamChunk) {
+             throw new Error(
+                 'Normal Prompt source emitted data after its terminal error
header.',
+             );
+         }
+         funStreamChunk(encodedChunk);
+     },
+ });
+

```

[AI Code/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/stream.endpoint.ts](#)

Copy

Review

Accept

  patching file:  [iaStreamHandlers.ts](#)

[AI Code/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts](#)

Copy

Review

Accept

PatchFull file

```
- import type { AiStreamedChunk } from
'../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
- import {
-   buildInterruptedChunk,
-   buildSystemErrorChunk,
-   decodeLiveSessionSourceErrorV1,
- } from '../../../../../sharedlib/types/aiTypes/aiStopSuffix.util.js';
-
+ import type { AiStreamedChunk } from
'../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import { decodeLiveSessionSourceErrorV1 } from
'../../../../../sharedlib/types/aiTypes/aiStopSuffix.util.js';
+


---


- import { sidebarQueryRunningBridge } from './SidebarQueryRunningBridge.js';
-
+ import { sidebarQueryRunningBridge } from './SidebarQueryRunningBridge.js';
+ import {
+   resolveNormalPromptLiveSessionReplayPolicy,
+   stageNormalPromptLiveSessionReservation,
+   unstageNormalPromptLiveSessionReservation,
+ } from
'../../../../../aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.js';
+


---


-   const sourceStartedAtMs = Date.now();
-
+   const sourceStartedAtMs = Date.now();
+   const replayPolicy = resolveNormalPromptLiveSessionReplayPolicy(
+     args.aiQuery,
+   );
+


---


-   ]),
-   readiness:
-     args.aiQuery.type === 'continueResponse'
-       ? 'baselineOnly'
-       : 'baselineAndObservableChunk',
-
+   ]),
+   readiness: replayPolicy.readiness,
+


---


-   ownership: 'source',
-   replayMode:
-     args.aiQuery.type === 'continueResponse'
-       ? 'continueAssistant'
-       : 'replaceTurnAssistants',
-
+   ownership: 'source',
+   replayMode: replayPolicy.replayMode,
+


---


-   }
-   const baseline = orchAsk.stream.memory.getThreadMessagesSnapshot(
-     runRef.locator.sessionId,
-   );
-   liveSession.registerRun({ reservation });
-   if (args.aiQuery.type === 'continueResponse') {
-     liveSession.markBaselineReady({
-
+   }
+
+ }
```

```

+    // Keep the opaque reservation process-local while allowing the Slimbk endpoint to
+    consume it once.
+    try {
+        stageNormalPromptLiveSessionReservation({
+
+            runRef,
+            baselineMessages: baseline,
+
+            runRef,
+            reservation,
+
+            reservation,
+        });
+
+        reservation,
+    });
+    } catch (error) {
+        liveSession.cancelSourceReservation(reservation);
+        throw error;
+    }
+
+    const activity = sidebarQueryRunningBridge.allocateStart({
+        runRef,
+        ownership: 'source',
+        sourceStartedAtMs,
+        panelState: 'open',
+    });
+
+    let activity: ReturnType<
+        typeof sidebarQueryRunningBridge.allocateStart
+    >;
+    try {
+        activity = sidebarQueryRunningBridge.allocateStart({
+            runRef,
+            ownership: 'source',
+            sourceStartedAtMs,
+            panelState: 'open',
+        });
+    } catch (error) {
+        unstageNormalPromptLiveSessionReservation({ runRef, reservation });
+        liveSession.cancelSourceReservation(reservation);
+        throw error;
+    }
+
+    const chunk = value as AiStreamedChunk;
+    if (chunk.type === 'meta' && chunk.value.metaType === 'error') {
+        await orchAsk.stream.memory.patchLastAssistantMeta({
+            threadId: runRef.locator.sessionId,
+            turnId: runRef.turnId,
+            patch: {
+                errorReason: chunk.value.value.reason,
+                errorMessage: chunk.value.value.message,
+                ...(chunk.value.value.details === undefined
+                    ? {}
+                    : { errorDetails: chunk.value.value.details }),
+            },
+        });
+    }
+    liveSession.appendChunk({ runRef, chunk });

```

```

-
+         const chunk = value as AiStreamedChunk;
+
-
-         hasTerminalData ||= isLiveSessionTerminalMetaChunk(chunk);
-         if (
-             args.aiQuery.type !== 'continueResponse' &&
-             chunk.type === 'meta' &&
-             chunk.value.metaType === 'fallback_persisted'
-         ) {
-             liveSession.markBaselineReady({
-                 runRef,
-                 baselineMessages:
-                     orchAsk.stream.memory.getThreadMessagesSnapshot(
-                         runRef.locator.sessionId,
-                     ),
-             });
-         }
+         hasTerminalData ||= isLiveSessionTerminalMetaChunk(chunk);
+
+         // Registry append and terminal durability already completed inside the
source endpoint.
+
-
-         transportError = normalizeError(error);
-         }
-     }
+         transportError = normalizeError(error);
+     }
+
+     // The endpoint normally consumes this capability; remove it here only when
transport failed earlier.
+     unstageNormalPromptLiveSessionReservation({ runRef, reservation });
+
-
-     }
-     if (!hasTerminalData) {
-         enqueueChunk(
-             transportError
-                 ? buildInterruptedChunk(runRef.turnId, transportError)
-                 : buildSystemErrorChunk(
-                     runRef.turnId,
-                     'Normal Prompt source ended without terminal metadata.',
-                 ),
-             );
-         try {
-             await fifo;
-         } catch (error) {
-             transportError = normalizeError(error);
-             fifo = Promise.resolve();
-         }
+     }
+     if (!hasTerminalData && !transportError) {
+         transportError = new Error(
+             'Normal Prompt source transport completed without terminal metadata.',
+         );
+

```

[AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts](#)

Copy

Review

Accept



patching file: [delete.endpoint.ts](#)

[AI/Code/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { getLiveSessionServiceForProd } from
'../../../../../../aiTools/orch/liveSession/core/LiveSessionService.js';
-
+ import { getLiveSessionServiceForProd } from
'../../../../../../aiTools/orch/liveSession/core/LiveSessionService.js';
+ import { reconcileClosedQueryActivityFromLiveAuthority } from
'../../../../../../extension/panels/sidebar/SidebarStore.rehydrate.manager.js';
+
-
- import { reconcileClosedQueryActivityFromLiveAuthority } from
'../../../../../../extension/panels/sidebar/SidebarStore.rehydrate.manager.js';
-
+ import { reconcileClosedQueryActivityFromLiveAuthority } from
'../../../../../../extension/panels/sidebar/SidebarStore.rehydrate.manager.js';
+
+ /** Normalize deletion and reconciliation failures before deterministic aggregation. */
+ function normalizeError(error: unknown): Error {
+   return error instanceof Error ? error : new Error(String(error));
+ }
+
-
-   );
-   }
-
-
+   );
+   }
+
-
-   */
-
+   */
+   let sourceWasActive = false;
+   let deletionFailure: Error | undefined;
+
-
-   try {
-
+   try {
+     sourceWasActive = liveSession
+       .listActiveSnapshots()
+       .some(
+         (snapshot) =>
+           snapshot.runRef.locator.sessionId ===
+             locator.sessionId &&
+             snapshot.runRef.locator.storageScopeRelPath ===
+               locator.storageScopeRelPath &&
+             (snapshot.kind === 'reservation' ||
+               (snapshot.ownership === 'source' &&
+                 !snapshot.sourceSettled &&
+                 !snapshot.handoffReady)),
+         );
+   }
+
-
-   }
```

```

-         closePromptPanel(locator);
-         await forceRemoveQueryActivity(locator);
-
+         closePromptPanel(locator);
+
+         // Closed idle/observer projections can disappear immediately; active
sources remain recoverable.
+         if (!sourceWasActive) {
+             await forceRemoveQueryActivity(locator);
+         }
+
-         locator,
-     );
-
+         locator,
+     );
+
+     // Durable success retires live authority before any auxiliary Activity
broadcast can fail.
+
-         liveSession.evictLocator(locator, destructionClaim);
-
+         liveSession.evictLocator(locator, destructionClaim);
+         if (sourceWasActive) {
+             await forceRemoveQueryActivity(locator);
+         }
+
-     } catch (err) {
-         const systemMessage: string =
-             err instanceof Error ? err.message : String(err);
-
+     } catch (err) {
+         deletionFailure = normalizeError(err);
+
-         void VsCode.window.showErrorMessage(
-             `This chat could not be deleted due to a system error.\n\n${
{systemMessage}`,
-
+         void VsCode.window.showErrorMessage(
+             `This chat could not be deleted due to a system error.\n\n${
{deletionFailure.message}`,
+
-         );
-         throw err;
-
+         );
+
-
-         // Remove the corresponding chat list item and clear selection if it pointed to
this session.
-
+
+         if (deletionFailure) {
+             try {
+                 // Revalidate after claim release: active sources stay closed/running,
everything else disappears.
+                 await reconcileClosedQueryActivityFromLiveAuthority(locator);
+             } catch (reconciliationError) {
+                 throw new AggregateError(
+                     [

```

```

+         normalizeError(deletionFailure),
+         normalizeError(reconciliationError),
+     ],
+     'Chat deletion failed and Activity reconciliation also failed.',
+ );
+ }
+ throw deletionFailure;
+ }
+
+ // Remove the corresponding chat list item and clear selection if it pointed to
this session.

```

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)

Copy

Review

Accept

  patching file:  [Prompt.tsx](#)

[AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx](#)

Copy

Review

Accept

PatchFull file

```

- } from '../liveSession/adapters/PromptLiveSessionBootstrap.adapter.js';
-
+ } from '../liveSession/adapters/PromptLiveSessionBootstrap.adapter.js';
+ import { usePromptRouterQueryStringRawSnapshot } from
'../lib/usePromptSessionLocator.hook.js';
+
- export function Prompt(): JSX.Element {
-     const fallbackSessionId = useQueryString('sessionId');
-     const fallbackStorageScopeRelPath =
-         useQueryString('storageScopeRelPath') ?? '';
-     const rawQuerySnapshot = useMemo(() => {
-         const descriptor = Object.getOwnPropertyDescriptor(
-             globalThis,
-             '__AICODE_ROUTER_QUERY_STRING__',
-         );
-         if (typeof descriptor?.value === 'string') return descriptor.value;
-         return fallbackSessionId
-             ? `sessionId=${encodeURIComponent(
-                 fallbackSessionId,
-             )}&storageScopeRelPath=${encodeURIComponent(
-                 fallbackStorageScopeRelPath,
-             )}`
-             : undefined;
-     }, [fallbackSessionId, fallbackStorageScopeRelPath]);
-
+ export function Prompt(): JSX.Element {
+     // The duplicate-sensitive bootstrap owns one immutable raw query snapshot for the
panel lifetime.
+     const rawQuerySnapshot = usePromptRouterQueryStringRawSnapshot();
+

```

[AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx](#)

Copy

Review

Accept

  patching file:  [usePromptRunCommands.hook.ts](#)

[AICode/src/frontend/src/app/prompt/Prompt/usePromptRunCommands.hook.ts](#)

Copy

Review

Accept

PatchFull file

```
- import type { AiStreamStartResult } from
- '../../../../../messageBus/AiStreamStartArgs.type.js';
-
+ import type { AiStreamStartResult } from
+ '../../../../../messageBus/AiStreamStartArgs.type.js';
+ import { preparePromptContinuationStreaming } from
+ './preparePromptContinuationStreaming.util.js';
+
-
-     const handleContinueResponse = useCallback(): AiStreamStartResult => {
-         // Resolve continuation identity purely before acquiring the transport single-
- flight.
-         let plannedAssistantTurnId: string | undefined;
-         for (let i = messages.length - 1; i >= 0; i -= 1) {
-             if (
-                 messages[i]?.role === 'assistant' &&
-                 messages[i]?.assistantBubbleKind !== 'think'
-             ) {
-                 plannedAssistantTurnId = messages[i]?.turnId;
-                 break;
-             }
-         }
-         const plannedTurnId =
-             plannedAssistantTurnId ?? `turn:${uidGenerator.newUniqueId}`;
-
+     const handleContinueResponse = useCallback(): AiStreamStartResult => {
+         // Build the complete continuation scaffold without mutating React state or
+ streaming refs.
+         const plan = preparePromptContinuationStreaming(messages);
+
-
-         kind: 'promptQuery',
-         turnId: plannedTurnId,
-
+         kind: 'promptQuery',
+         turnId: plan.turnId,
+
-
-         // Reuse last assistant as target
-         let lastAssistantUid: string | undefined = undefined;
-         let lastAssistantText = '';
-         let lastAssistantTurnId: string | undefined = undefined;
-
-         // Walk messages backward to find the latest assistant bubble
-         for (let i = messages.length - 1; i >= 0; i--) {
-             const m = messages[i];
-             if (m?.role === 'assistant' && m.assistantBubbleKind !== 'think') {
-                 lastAssistantUid = m.uid;
-                 lastAssistantText = m.text ?? '';
-                 lastAssistantTurnId = m.turnId;
-                 break;
-             }
-         }
-
-         // Capture base text for append mode
-         baseAssistantTextRef.current = lastAssistantText;
-
-         // Fresh thinking UID for continuation
```

```

-     const newThinkingUid = uidGenerator.newUniqueId;
-     streamingThinkingUidRef.current = newThinkingUid;
-
-     // Keep the same turnId as the assistant being continued
-     const continueTurnId = lastAssistantTurnId ?? plannedTurnId;
-     currentTurnIdRef.current = continueTurnId;
-
-     // Insert thinking bubble before targeted assistant bubble
-     setMessages((prev) => {
-         const out = [...prev];
-         if (lastAssistantUid) {
-             // Find the target assistant index to insert a thinking bubble right
before it
-             const aiIdx = out.findIndex((m) => m.uid === lastAssistantUid);
-             const thinkMsg: PromptMessage = {
-                 turnId: continueTurnId,
-                 uid: newThinkingUid,
-                 role: 'assistant',
-                 text: '',
-                 assistantBubbleKind: 'think',
-                 thinkingInitialOpen: false,
-             };
-             // Insert before existing assistant or push if not found defensively
-             if (aiIdx >= 0) {
-                 out.splice(aiIdx, 0, thinkMsg);
-             } else {
-                 out.push(thinkMsg);
-             }
-         }
-         return out;
-     });
-
+
+     // Commit the exact plan only after the transport single-flight accepted this
attempt.
+     baseAssistantTextRef.current = plan.baseAssistantText;
+     streamingThinkingUidRef.current = plan.thinkingUid;
+     streamingMessageUidRef.current = plan.assistantUid;
+     currentTurnIdRef.current = plan.turnId;
+     setMessages(plan.messages);
+
-
-     resetLastReceivedStreamKind();
-
-     // Set target assistant UID BEFORE starting the stream
-     streamingMessageUidRef.current = lastAssistantUid;
-
-     // Continuation is a fresh streaming segment: lock file-edit actions until the
new end-of-turn finalization.
-     if (lastAssistantUid) {
-         setMessages((prev) => {
-             const idx = prev.findIndex((m) => m.uid === lastAssistantUid);
-             if (idx < 0) return prev;
-             const existing = prev[idx];
-             if (existing.role !== 'assistant') return prev;
-             if (existing.fileEditsUiFinalized === false) return prev;
-             const out = [...prev];
-             out[idx] = {
-                 ...(existing as object),
-                 fileEditsUiFinalized: false,
-             } as unknown as PromptMessage;
-             return out;
-         });
-     }

```

```
-
+         resetLastReceivedStreamKind();
+
+ AICode/src/frontend/src/app/prompt/Prompt/usePromptRunCommands.hook.ts
```

Copy

Review

Accept

  **reading project:**  [NormalPromptLiveSessionSource.service.ts](#) [L1-L90],  [NormalPromptLiveSessionSource.service.ts](#) [L90-L220],  [stream.endpoint.ts](#) [L1-L220],  [iaStreamHandlers.ts](#) [L1-L320],  [usePromptRunCommands.hook.ts](#) [L320-L460],  [Prompt.tsx](#) [L55-L125]

  **patching file:**  [NormalPromptLiveSessionSource.service.ts](#)
[AICode/src/extension/src/aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { isLiveSessionTerminalMetaChunk } from
'../../../../../sharedlib/types/liveSessionTypes/LiveSessionStream.util.js';
- import type { LiveSessionRunRef } from
'../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
-
+ import { isLiveSessionTerminalMetaChunk } from
'../../../../../sharedlib/types/liveSessionTypes/LiveSessionStream.util.js';
+ import {
+   buildLiveSessionRunKey,
+   type LiveSessionRunRef,
+ } from ' ../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
+
-
+
+   });
- }
-
+   });
+ }
+
+ /**
+  * Process-local ownership bridge between the exact Prompt webview handler and the
+  * local Slimbk endpoint.
+  * The opaque reservation handle never crosses the serialized transport boundary: only
+  * the handler that
+  * successfully reserved the run can stage it, and the endpoint consumes that exact
+  * handle once.
+  */
+ const stagedReservationByRunKey = new Map<
+   string,
+   LiveSessionSourceReservationHandle
+ >();
+
+ /** Stage one exact opaque reservation before the handler starts the local endpoint
+  * request. */
+ export function stageNormalPromptLiveSessionReservation(args: {
+   runRef: LiveSessionRunRef;
+   reservation: LiveSessionSourceReservationHandle;
+ }): void {
+   const runKey = buildLiveSessionRunKey(args.runRef);
+
+   // A second staged owner would let a duplicate transport consume another handler's
+   reservation.
+   if (stagedReservationByRunKey.has(runKey)) {
```

```
+         throw new Error(
+             'Normal Prompt live-session reservation is already staged.',
+         );
+     }
+     stagedReservationByRunKey.set(runKey, args.reservation);
+ }
+
+ /** Consume the exact staged reservation once immediately before source execution. */
+ export function takeNormalPromptLiveSessionReservation(
+     runRef: LiveSessionRunRef,
+ ): LiveSessionSourceReservationHandle {
+     const runKey = buildLiveSessionRunKey(runRef);
+     const reservation = stagedReservationByRunKey.get(runKey);
+     if (!reservation) {
+         throw new Error(
+             'Normal Prompt live-session source has no staged reservation.',
+         );
+     }
+
+     // Delete before execution so retries or duplicate endpoint calls cannot reuse the
+     // capability.
+     stagedReservationByRunKey.delete(runKey);
+     return reservation;
+ }
+
+ /** Remove only the still-staged reservation owned by the exact handler during
+  * transport cleanup. */
+ export function unstageNormalPromptLiveSessionReservation(args: {
+     runRef: LiveSessionRunRef;
+     reservation: LiveSessionSourceReservationHandle;
+ }): boolean {
+     const runKey = buildLiveSessionRunKey(args.runRef);
+     if (stagedReservationByRunKey.get(runKey) !== args.reservation) {
+         return false;
+     }
+     stagedReservationByRunKey.delete(runKey);
+     return true;
+ }
+ }
```

```
-      let hasTerminalMeta = false;
-
+      let hasTerminalMeta = false;
+      let sourceStarted = false;
+
```

```
-      onSourceStarted: () => {  
-        this.deps.liveSession.registerRun({  
-  
+      onSourceStarted: () => {  
+        const registration = this.deps.liveSession.registerRun({  
+
```

```
-         reservation: args.reservation,
-     });
-
+         reservation: args.reservation,
+     });
+     sourceStarted = true;
+
```

-	}
-	},
-	
+	}

```

+
+          // A Stop accepted while the source was only reserved must also
reach the now-primed stream.
+          if (registration.abortRequested) {
+              this.deps.orchStream.abortStream(
+                  args.runRef.locator.sessionId,
+                  'normal_prompt_live_session_source_stop',
+              );
+          }
+      },
+
-      } catch (error) {
-          if (!hasTerminalMeta) {
-
+      } catch (error) {
+          // A synchronous iterator-priming failure leaves only a reservation, so no
registry append is legal.
+          if (sourceStarted && !hasTerminalMeta) {
+

```

[AI Code/src/extension/src/aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.ts](#)

Copy

Review

Accept



patching file: [NormalPromptLiveSessionSource.service.test.ts](#)

[AI Code/src/extension/src/aiTools/orch/liveSession/source/ test /NormalPromptLiveSessionSource.service.test.ts](#)

Copy

Review

Accept

PatchFull file

```

-      resolveNormalPromptLiveSessionReplayPolicy,
-
+      resolveNormalPromptLiveSessionReplayPolicy,
+      stageNormalPromptLiveSessionReservation,
+      takeNormalPromptLiveSessionReservation,
+      unstageNormalPromptLiveSessionReservation,
+
- describe('NormalPromptLiveSessionSourceService', () => {
-
+ describe('NormalPromptLiveSessionSourceService', () => {
+     it('hands one opaque reservation from the exact handler to the endpoint exactly
once', () => {
+         const otherReservation: LiveSessionSourceReservationHandle =
+             Object.freeze({ token: Symbol('other-reservation') });
+
+         stageNormalPromptLiveSessionReservation({
+             runRef: RUN,
+             reservation: RESERVATION,
+         });
+         expect(() =>
+             stageNormalPromptLiveSessionReservation({
+                 runRef: RUN,
+                 reservation: otherReservation,
+             }),
+         ).toThrow('already staged');
+         expect(takeNormalPromptLiveSessionReservation(RUN)).toBe(RESERVATION);
+         expect(() => takeNormalPromptLiveSessionReservation(RUN)).toThrow(
+             'no staged reservation',
+         );
+     });
+

```



```
+ stageNormalPromptLiveSessionReservation({
+   runRef: RUN,
+   reservation: RESERVATION,
+ });
+ expect(
+   unstageNormalPromptLiveSessionReservation({
+     runRef: RUN,
+     reservation: otherReservation,
+   }),
+ ).toBe(false);
+ expect(
+   unstageNormalPromptLiveSessionReservation({
+     runRef: RUN,
+     reservation: RESERVATION,
+   }),
+ ).toBe(true);
+ });
+
- expect(order).toEqual(['register', 'baseline', 'append', 'forward']);
- });
-
+ expect(order).toEqual(['register', 'baseline', 'append', 'forward']);
+ });
+
+ it('reapplies an abort accepted during reservation immediately after promotion',
async () => {
+   const abortStream = jest.fn<(threadId: string, reason: string) => void>();
+   const terminal: AiStreamedChunk = {
+     turnId: RUN.turnId,
+     type: 'meta',
+     value: {
+       metaType: 'stop_reason',
+       value: { reason: AiStopReason.END_TURN },
+     },
+   };
+ };
+ const service = new NormalPromptLiveSessionSourceService({
+   liveSession: {
+     registerRun: () => ({ abortRequested: true }),
+     markBaselineReady: () => ({ becameReplayReady: false }),
+     appendChunk: () => ({ becameReplayReady: false }),
+   },
+   lifecycle: new LiveSessionSourceLifecycleService(),
+   orchStream: {
+     askStream: () => Readable.from([encode(terminal)]),
+     abortStream,
+     memory: {
+       getThreadMessagesSnapshot: () => [],
+       patchLastAssistantMeta: async () => {},
+     },
+   },
+   parseChunk: (raw) => JSON.parse(raw) as AiStreamedChunk,
+ });
+
+ await service.execute({
+   reservation: RESERVATION,
+   runRef: RUN,
+   aiCatalog: { engines: [] },
+   aiQuery: { type: 'prompt', promptValue: 'hello' },
+   aiQueryOptions: {},
+   onChunk: async () => {},
+ });
```

```

+         expect(abortStream).toHaveBeenCalledWith(
+             RUN.locator.sessionId,
+             'normal_prompt_live_session_source_stop',
+         );
+     });
+
+     it('does not append when synchronous iterator priming fails before reservation
promotion', async () => {
+         const readable = new Readable({ read: () => {} });
+         Object.defineProperty(readable, Symbol.asyncIterator, {
+             configurable: true,
+             value: () => {
+                 throw new Error('priming failed');
+             },
+         });
+         const registerRun = jest.fn(() => ({ abortRequested: false }));
+         const appendChunk = jest.fn<
+             (args: { runRef: LiveSessionRunRef; chunk: AiStreamedChunk }) => {
+                 becameReplayReady: boolean;
+             }
+         >(() => ({ becameReplayReady: false }));
+         const service = new NormalPromptLiveSessionSourceService({
+             liveSession: {
+                 registerRun,
+                 markBaselineReady: () => ({ becameReplayReady: false }),
+                 appendChunk,
+             },
+             lifecycle: new LiveSessionSourceLifecycleService(),
+             orchStream: {
+                 askStream: () => readable,
+                 abortStream: () => {},
+                 memory: {
+                     getThreadMessagesSnapshot: () => [],
+                     patchLastAssistantMeta: async () => {},
+                 },
+             },
+             parseChunk: (raw) => JSON.parse(raw) as AiStreamedChunk,
+         });
+
+         await expect(
+             service.execute({
+                 reservation: RESERVATION,
+                 runRef: RUN,
+                 aiCatalog: { engines: [] },
+                 aiQuery: { type: 'prompt', promptValue: 'hello' },
+                 aiQueryOptions: {},
+                 onChunk: async () => {},
+             }),
+         ).rejects.toThrow('priming failed');
+         expect(registerRun).not.toHaveBeenCalled();
+         expect(appendChunk).not.toHaveBeenCalled();
+     });
+
+

```

[AICode/src/extension/src/aiTools/orch/liveSession/source/ test /NormalPromptLiveSessionSource.service.test.ts](#)

Copy

Review

Accept

  patching file:  [stream.endpoint.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Error

([what is it?](#))

  patching file:  [stream.endpoint.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Error

([what is it?](#))

  patching file:  [stream.endpoint.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Error

([what is it?](#))

  reading project:  [stream.endpoint.test.ts](#)

  patching file:  [stream.endpoint.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { beforeEach, describe, expect, it, jest } from '@jest/globals';
-
+ import { beforeEach, describe, expect, it, jest } from '@jest/globals';
+ import { Readable } from 'node:stream';
+
+
- import type { SlimbkResult } from
'../../../../../../../../sharedlib/slimbk/framework/SlimbkResult.type.js';
-
+ import type { SlimbkResult } from
'../../../../../../../../sharedlib/slimbk/framework/SlimbkResult.type.js';
+ import type { LiveSessionService } from
'../../../../../../../../aiTools/orch/liveSession/core/LiveSessionService.js';
+ import type { LiveSessionSourceReservationHandle } from
'../../../../../../../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
+
- const patchProjectSettingsMock = jest.fn<(patch: unknown) => Promise<void>>();
- const askStreamMock = jest.fn<(args: unknown) => AsyncIterable<string>>();
-
+ const patchProjectSettingsMock = jest.fn<(patch: unknown) => Promise<void>>();
+ const askStreamMock = jest.fn<(args: unknown) => Readable>();
+ const abortStreamMock = jest.fn<(threadId: string, reason: string) => void>();
+ const getThreadMessagesSnapshotMock = jest.fn<(threadId: string) => []>();
+ const patchLastAssistantMetaMock = jest.fn<(args: unknown) => Promise<void>>();
+ const registerRunMock = jest.fn<LiveSessionService['registerRun']>();
+ const markBaselineReadyMock = jest.fn<LiveSessionService['markBaselineReady']>();
+ const appendChunkMock = jest.fn<LiveSessionService['appendChunk']>();
+ const reservation: LiveSessionSourceReservationHandle = Object.freeze({
+   token: Symbol('endpoint-reservation'),
+ });
+
-
+   askStream: askStreamMock,
-
+   askStream: askStreamMock,
+   abortStream: abortStreamMock,
+   memory: {
+     getThreadMessagesSnapshot: getThreadMessagesSnapshotMock,
```

```

+         patchLastAssistantMeta: patchLastAssistantMetaMock,
+     },
+
-
- const endpoint = (await import('../stream.endpoint.js')).default;
-
+
+ jest.unstable_mockModule(
+     '../../../../../aiTools/orch/liveSession/core/LiveSessionService.js',
+     () => ({
+         getLiveSessionServiceForProd: () => ({
+             registerRun: registerRunMock,
+             markBaselineReady: markBaselineReadyMock,
+             appendChunk: appendChunkMock,
+         }),
+     }),
+ );
+
+ const { stageNormalPromptLiveSessionReservation } = await import(
+     '../../../../../aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.js'
+ );
+ const endpoint = (await import('../stream.endpoint.js')).default;
+
-
- async function* createChunkStream(
-     chunks: readonly string[],
- ): AsyncGenerator<string, void, void> {
-     for (const chunk of chunks) {
-         yield chunk;
-     }
- }
-
+
+ /** Build the lazy Node source consumed by the real normal-Prompt source service. */
+ function createChunkStream(chunks: readonly string[]): Readable {
+     return Readable.from(chunks);
+
-     };
- }
-
+     };
+ }
+
+ /** Stage the exact process-local capability consumed by the endpoint before source
+ execution. */
+ function stageReservation(args: ReturnType<typeof buildArgs>): void {
+     stageNormalPromptLiveSessionReservation({
+         runRef: {
+             locator: {
+                 sessionId: args.threadId,
+                 storageScopeRelPath: args.storageScopeRelPath,
+             },
+             turnId: args.turnId,
+             sourceId: args.sourceId,
+         },
+         reservation,
+     });
+ }
+
-
- askStreamMock.mockReset();

```

```

-
+ askStreamMock.mockReset();
+ abortStreamMock.mockReset();
+ getThreadMessagesSnapshotMock.mockReset();
+ patchLastAssistantMetaMock.mockReset();
+ registerRunMock.mockReset();
+ markBaselineReadyMock.mockReset();
+ appendChunkMock.mockReset();
+
+ getThreadMessagesSnapshotMock.mockReturnValue([]);
+ patchLastAssistantMetaMock.mockResolvedValue(undefined);
+ registerRunMock.mockReturnValue({ abortRequested: false });
+ markBaselineReadyMock.mockReturnValue({ becameReplayReady: false });
+ appendChunkMock.mockReturnValue({ becameReplayReady: false });
+
-
- it('persists the SPECIF Code programming doc preference before loading the catalog
and starting the stream', async () => {
-   patchProjectSettingsMock.mockResolvedValue(undefined);
-   loadAiCatalogMock.mockResolvedValue({ catalog: 'ok' });
-   askStreamMock.mockReturnValue(
-     createChunkStream([
-       JSON.stringify({
-         type: 'meta',
-         value: {
-           metaType: 'stop_reason',
-           value: { reason: 'end_turn' },
-         },
-       )],
-     ),
-   );
-
-   const funStreamChunkMock = jest.fn<(chunk: string) => void>(() => {});
-   const funSendStreamResultMock = jest.fn<
-     (result: SlimbkResult) => (chunk: string) => void
-   >(() => funStreamChunkMock);
-
-   const parsedArgs = endpoint.argsSchema.parse(
-     buildArgs({
-       type: 'specifAction',
-       action: 'code',
-       specifName: 'SPECIF_X',
-       specifMarkdown: '## Specif',
-       strictMode: false,
-       programmingDocEnabled: false,
-       additionalInstructions: '',
-     }),
-   );
-
-   await endpoint.streamedHandler(
-     { headers: {}, args: parsedArgs },
-     funSendStreamResultMock,
-   );
-
-   expect(patchProjectSettingsMock).toHaveBeenCalledWith({
-     specifCodeProgrammingDocEnabled: false,
-   });
-   expect(loadAiCatalogMock).toHaveBeenCalledTimes(1);
-   expect(askStreamMock).toHaveBeenCalledTimes(1);
-
-   const patchCallOrder =
-     patchProjectSettingsMock.mock.invocationCallOrder[0];
-   const loadCatalogCallOrder =

```

```

-         loadAiCatalogMock.mock.invocationCallOrder[0];
-         const askStreamCallOrder = askStreamMock.mock.invocationCallOrder[0];
-
-         expect(patchCallOrder).toBeLessThan(loadCatalogCallOrder);
-         expect(loadCatalogCallOrder).toBeLessThan(askStreamCallOrder);
-     });
-
+
+     it('persists the SPECIF Code programming doc preference before loading the catalog
and starting the stream', async () => {
+         patchProjectSettingsMock.mockResolvedValue(undefined);
+         loadAiCatalogMock.mockResolvedValue({ engines: [] });
+         askStreamMock.mockReturnValue(
+             createChunkStream([
+                 JSON.stringify({
+                     turnId: 'turn-1',
+                     type: 'meta',
+                     value: {
+                         metaType: 'stop_reason',
+                         value: { reason: 'end_turn' },
+                     },
+                 }),
+             ]),
+         );
+
+         const funStreamChunkMock = jest.fn<(chunk: string) => void>(() => {});
+         const funSendStreamResultMock = jest.fn<
+             (result: SlimbkResult) => (chunk: string) => void
+         >(() => funStreamChunkMock);
+
+         const parsedArgs = endpoint.argsSchema.parse(
+             buildArgs({
+                 type: 'specifAction',
+                 action: 'code',
+                 specifName: 'SPECIF_X',
+                 specifMarkdown: '## Specif',
+                 strictMode: false,
+                 programmingDocEnabled: false,
+                 additionalInstructions: '',
+             })
+         );
+         stageReservation(parsedArgs);
+
+         await endpoint.streamedHandler(
+             { headers: {}, args: parsedArgs },
+             funSendStreamResultMock,
+         );
+
+         expect(patchProjectSettingsMock).toHaveBeenCalledWith({
+             specifCodeProgrammingDocEnabled: false,
+         });
+         expect(loadAiCatalogMock).toHaveBeenCalledTimes(1);
+         expect(askStreamMock).toHaveBeenCalledTimes(1);
+         expect(registerRunMock).toHaveBeenCalledWith({ reservation });
+         expect(appendChunkMock).toHaveBeenCalledTimes(1);
+
+         const patchCallOrder =
+             patchProjectSettingsMock.mock.invocationCallOrder[0];
+         const loadCatalogCallOrder =
+             loadAiCatalogMock.mock.invocationCallOrder[0];
+         const askStreamCallOrder = askStreamMock.mock.invocationCallOrder[0];
+
+         expect(patchCallOrder).toBeLessThan(loadCatalogCallOrder);

```

```

+         expect(loadCatalogCallOrder).toBeLessThan(askStreamCallOrder);
+     });
+
-
-
-     it('does not persist the preference for non-code queries', async () => {
-         loadAiCatalogMock.mockResolvedValue({ catalog: 'ok' });
-         askStreamMock.mockReturnValue(
-             createChunkStream([
-                 JSON.stringify({
-                     type: 'meta',
-                     value: {
-                         metaType: 'stop_reason',
-                         value: { reason: 'end_turn' },
-                     },
-                 }),
-             ]),
-         );
-
-         const funStreamChunkMock = jest.fn<(chunk: string) => void>(() => {});
-         const funSendStreamResultMock = jest.fn<
-             (result: SlimbkResult) => (chunk: string) => void
-         >(() => funStreamChunkMock);
-
-         const parsedArgs = endpoint.argsSchema.parse(
-             buildArgs({
-                 type: 'prompt',
-                 promptValue: 'Hello',
-             }),
-         );
-
-         await endpoint.streamedHandler(
-             { headers: {}, args: parsedArgs },
-             funSendStreamResultMock,
-         );
-
-         expect(patchProjectSettingsMock).not.toHaveBeenCalled();
-         expect(loadAiCatalogMock).toHaveBeenCalledTimes(1);
-         expect(askStreamMock).toHaveBeenCalledTimes(1);
-     });
-
+
+     it('does not persist the preference for non-code queries', async () => {
+         loadAiCatalogMock.mockResolvedValue({ engines: [] });
+         askStreamMock.mockReturnValue(
+             createChunkStream([
+                 JSON.stringify({
+                     turnId: 'turn-1',
+                     type: 'meta',
+                     value: {
+                         metaType: 'stop_reason',
+                         value: { reason: 'end_turn' },
+                     },
+                 }),
+             ]),
+         );
+
+         const funStreamChunkMock = jest.fn<(chunk: string) => void>(() => {});
+         const funSendStreamResultMock = jest.fn<
+             (result: SlimbkResult) => (chunk: string) => void
+         >(() => funStreamChunkMock);
+
+         const parsedArgs = endpoint.argsSchema.parse(
+             buildArgs({

```

```

+         type: 'prompt',
+         promptValue: 'Hello',
+     })),
+ );
+ stageReservation(parsedArgs);
+
+ await endpoint.streamedHandler(
+     { headers: {}, args: parsedArgs },
+     funSendStreamResultMock,
+ );
+
+ expect(patchProjectSettingsMock).not.toHaveBeenCalled();
+ expect(loadAiCatalogMock).toHaveBeenCalledTimes(1);
+ expect(askStreamMock).toHaveBeenCalledTimes(1);
+ });
+

```

```

-
- it('encodes only a first meta.error as the private V1 header', async () => {
-     loadAiCatalogMock.mockResolvedValue({ catalog: 'ok' });
-
+
+ it('encodes only a first meta.error as the private V1 header', async () => {
+     loadAiCatalogMock.mockResolvedValue({ engines: [] });
+

```

```

-         () => jest.fn(),
-     );
-
+         () => jest.fn(),
+     );
+ const parsedArgs = endpoint.argsSchema.parse(
+     buildArgs({ type: 'prompt', promptValue: 'Hello' }),
+ );
+ stageReservation(parsedArgs);
+

```

```

- stageReservation(parsedArgs);
- await endpoint.streamedHandler(
-     {
-         headers: {},
-         args: endpoint.argsSchema.parse(
-             buildArgs({ type: 'prompt', promptValue: 'Hello' }),
-         ),
-     },
+
+ stageReservation(parsedArgs);
+ await endpoint.streamedHandler(
+     {
+         headers: {},
+         args: parsedArgs,
+     },

```

```

-     );
- expect(send).toHaveBeenCalledWith(
-     expect.objectContaining({
-
+     );
+ expect(patchLastAssistantMetaMock).toHaveBeenCalledTimes(1);
+ expect(appendChunkMock).toHaveBeenCalledTimes(1);
+ expect(send).toHaveBeenCalledWith(
+     expect.objectContaining({
+

```

```

-     it('keeps a later meta.error as an ordinary body chunk after the 200 header', async
() => {
-         loadAiCatalogMock.mockResolvedValue({ catalog: 'ok' });
-
+
+     it('keeps a later meta.error as an ordinary body chunk after the 200 header', async
() => {
+         loadAiCatalogMock.mockResolvedValue({ engines: [] });
+


---


-         },
-         );
-
+         },
+         );
+         const parsedArgs = endpoint.argsSchema.parse(
+             buildArgs({ type: 'prompt', promptValue: 'Hello' }),
+         );
+         stageReservation(parsedArgs);
+


---


-             headers: {},
-             args: endpoint.argsSchema.parse(
-                 buildArgs({ type: 'prompt', promptValue: 'Hello' }),
-             ),
-
+             headers: {},
+             args: parsedArgs,
+


---


-         expect(chunks).toEqual([first, lateError]);
-
+         expect(chunks).toEqual([first, lateError]);
+         expect(patchLastAssistantMetaMock).toHaveBeenCalledTimes(1);
+         expect(appendChunkMock).toHaveBeenCalledTimes(2);
+


---


- /** Stage the exact process-local capability consumed by the endpoint before source
execution. */
- function stageReservation(args: ReturnType<typeof buildArgs>): void {
-
+ /** Stage the exact process-local capability consumed by the endpoint before source
execution. */
+ function stageReservation(args: {
+     turnId: string;
+     sourceId: string;
+     threadId: string;
+     storageScopeRelPath: string;
+ }): void {
+


---


-         const askStreamCallOrder = askStreamMock.mock.invocationCallOrder[0];
-
+         const askStreamCallOrder = askStreamMock.mock.invocationCallOrder[0];
+         const registerCallOrder = registerRunMock.mock.invocationCallOrder[0];
+         const appendCallOrder = appendChunkMock.mock.invocationCallOrder[0];
+         const bodyCallOrder = funStreamChunkMock.mock.invocationCallOrder[0];
+


---


-         expect(loadCatalogCallOrder).toBeLessThan(askStreamCallOrder);
-
+         expect(loadCatalogCallOrder).toBeLessThan(askStreamCallOrder);
+         expect(askStreamCallOrder).toBeLessThan(registerCallOrder);
+         expect(registerCallOrder).toBeLessThan(appendCallOrder);
+         expect(appendCallOrder).toBeLessThan(bodyCallOrder);
+

```

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Copy

Review

Accept

  patching file:  [iaStreamHandlers.queryDoneSound.test.ts](#)

[AICode/src/extension/src/messageBus/handlers/ia/ test /iaStreamHandlers.queryDoneSound.test.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { beforeEach, describe, expect, it, jest } from '@jest/globals';
-
+ import { beforeEach, describe, expect, it, jest } from '@jest/globals';
+ import type { LiveSessionSourceReservationHandle } from
+ '../../../../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
+
- const webview = {};
- const appendChunkMock = jest.fn(() => {
-   order.push('append');
-   return { becameReplayReady: false };
-
+ const webview = {};
+ const reservation: LiveSessionSourceReservationHandle = Object.freeze({
+   token: Symbol('handler-reservation'),
+ });
+ const reserveSourceMock = jest.fn(() => reservation);
+ const cancelSourceReservationMock = jest.fn(() => true);
+ const finalizeSourceAfterForwardingMock = jest.fn(() => 'handoffReady');
+ const stageReservationMock = jest.fn(() => {
+   order.push('stage');
+ });
+ const unstageReservationMock = jest.fn(() => {
+   order.push('unstage');
+   return false;
+
-   getLiveSessionServiceForProd: () => ({
-     reserveSource: () => Object.freeze({ token: Symbol('r') }),
-     registerRun: () => ({ abortRequested: false }),
-     markBaselineReady: () => ({ becameReplayReady: false }),
-     appendChunk: appendChunkMock,
-     finalizeSourceAfterForwarding: () => 'handoffReady',
-
+     getLiveSessionServiceForProd: () => ({
+       reserveSource: reserveSourceMock,
+       cancelSourceReservation: cancelSourceReservationMock,
+       finalizeSourceAfterForwarding: finalizeSourceAfterForwardingMock,
+     }),
+   }),
+ );
+ jest.unstable_mockModule(
+   '../../../../../aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.js',
+   () => ({
+     resolveNormalPromptLiveSessionReplayPolicy: () => ({
+       replayMode: 'replaceTurnAssistants',
+       readiness: 'baselineAndObservableChunk',
+
-     readiness: 'baselineAndObservableChunk',
```

```

-    )),
-
+     readiness: 'baselineAndObservableChunk',
+   )),
+   stageNormalPromptLiveSessionReservation: stageReservationMock,
+   unstageNormalPromptLiveSessionReservation: unstageReservationMock,
+
-       abortStream: () => {},
-       memory: {
-         getThreadMessagesSnapshot: () => [],
-         patchLastAssistantMeta: async () => {
-           order.push('persist');
-         },
-       },
-
+       abortStream: () => {},
+
-   ) => {
-
+   ) => {
+     order.push('transport');
+
-   sidebarQueryRunningBridge: {
-     allocateStart: () => ({ runId: 1, publication: Promise.resolve() }),
-     onChunk: async () => {},
-
+   sidebarQueryRunningBridge: {
+     allocateStart: () => {
+       order.push('activityStart');
+       return { runId: 1, publication: Promise.resolve() };
+     },
+     onChunk: async () => {
+       order.push('activityChunk');
+     },
+
-   () => ({
-     onAiStreamStart: () => 1,
-     onAiStreamChunk: () => order.push('sound'),
-
+   () => ({
+     onAiStreamStart: () => {
+       order.push('soundStart');
+       return 1;
+     },
+     onAiStreamChunk: () => order.push('soundChunk'),
+
-   order.length = 0;
-
-   // Restore the registry append implementation removed by Jest resetMocks so the
order
-   // assertion observes durable error persistence before the exact buffer append.
-   appendChunkMock.mockImplementation(() => {
-     order.push('append');
-     return { becameReplayReady: false };
-
+   order.length = 0;
+   reserveSourceMock.mockImplementation(() => {
+     order.push('reserve');
+     return reservation;
+   });

```


Review

Accept

PatchFull file

```
-     jest.fn<LiveSessionService['releaseLocatorDestructionClaim']>());
-
+     jest.fn<LiveSessionService['releaseLocatorDestructionClaim']>());
+ const listActiveSnapshotsMock =
+     jest.fn<LiveSessionService['listActiveSnapshots']>());
+


---


- const deleteMock = jest.fn<(locator: ChatSessionLocator) => Promise<void>>();
-
+ const deleteMock = jest.fn<(locator: ChatSessionLocator) => Promise<void>>();
+ const forceRemoveActivityMock =
+     jest.fn<(locator: ChatSessionLocator) => Promise<void>>();
+ const reconcileActivityMock =
+     jest.fn<(locator: ChatSessionLocator) => Promise<void>>();
+


---


- const showErrorMessageMock = jest.fn<(message: string) => Promise<undefined>>();
-
+ const showErrorMessageMock = jest.fn<(message: string) => Promise<undefined>>();
+ const runRef = {
+     locator: {
+         sessionId: 'a'.repeat(32),
+         storageScopeRelPath: '',
+     },
+     turnId: 'turn',
+     sourceId: 'source',
+ } as const;
+


---


-         releaseLocatorDestructionClaim: releaseMock,
-
+         releaseLocatorDestructionClaim: releaseMock,
+         listActiveSnapshots: listActiveSnapshotsMock,
+


---


-     () => ({
-         forceRemoveQueryActivity: async () => {
-             order.push('removeActivity');
-         },
-     });
+     () => ({
+         forceRemoveQueryActivity: forceRemoveActivityMock,
+     }),
+ );
+ jest.unstable_mockModule(
+     '../../../../../extension/panels/sidebar/SidebarStore.rehydrate.manager.js',
+     () => ({
+         reconcileClosedQueryActivityFromLiveAuthority: reconcileActivityMock,
+


---


-         return true;
-     });
-
+         return true;
+     });
+     listActiveSnapshotsMock.mockReturnValue([
+         {
+             kind: 'reservation',
+             runRef,
+             viewerGroupId: 'normalPrompt',
+             replayMode: 'replaceTurnAssistants',
+             sourceStartedAtMs: 1,
```

```

+         abortRequested: false,
+         locatorEpoch: 1,
+     },
+ ]);
+
-     order.push('delete');
- });
-
+     order.push('delete');
+ });
+ forceRemoveActivityMock.mockImplementation(async () => {
+     order.push('removeActivity');
+ });
+ reconcileActivityMock.mockImplementation(async () => {
+     order.push('reconcile');
+ });
+
-     'close',
-     'removeActivity',
-
+     'close',
+
-     'evict',
-
+     'evict',
+     'removeActivity',
+
-     expect(evictMock).not.toHaveBeenCalled();
-
+     expect(evictMock).not.toHaveBeenCalled();
+     expect(forceRemoveActivityMock).not.toHaveBeenCalled();
+
-     expect(releaseMock).toHaveBeenCalledWith(claim);
-
+     expect(releaseMock).toHaveBeenCalledWith(claim);
+     expect(reconcileActivityMock).toHaveBeenCalledWith(runRef.locator);
+     expect(order).toEqual(['close', 'delete', 'release', 'reconcile']);
+
-     expect(order).toEqual(['close', 'delete', 'release', 'reconcile']);
- });
-
+     expect(order).toEqual(['close', 'delete', 'release', 'reconcile']);
+ });
+
+ it('removes a closed idle projection before durable deletion while holding the
claim', async () => {
+     listActiveSnapshotsMock.mockReturnValue([]);
+
+     await endpoint.syncHandler({
+         args: {
+             sessionId: runRef.locator.sessionId,
+             storageScopeRelPath: runRef.locator.storageScopeRelPath,
+         },
+     } as Parameters<typeof endpoint.syncHandler>[0]);
+
+     expect(order).toEqual([
+         'close',
+         'removeActivity',
+         'delete',
+         'evict',

```

```
+         'release',
+     ]);
+     expect(reconcileActivityMock).not.toHaveBeenCalled();
+ });
+
```

[AI Code/src/extension/src/slimbk-local/endpoints/ai/orch/chat/_test_/delete.endpoint.liveSession.test.ts](#)

Copy

Review

Accept

  **patching file:**  [usePromptRunCommands.continueResponse.test.tsx](#)

[AI Code/src/frontend/src/app/prompt/Prompt/_test_/usePromptRunCommands.continueResponse.test.tsx](#)

Copy

Review

Accept

PatchFull file

```
-
- import { describe, it } from '@jest/globals';
-
+
+ import { beforeEach, describe, expect, it, jest } from '@jest/globals';
+ import { act, renderHook } from '@testing-library/react';
+ import { useRef, useState } from 'react';
+
+ import type { AiStreamStartArgs, AiStreamStartResult } from
+ '../../../../../messageBus/AiStreamStartArgs.type.js';
+ import type { AiStreamedChunk } from
+ '../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import type { PromptMessage } from '../../PromptHistory/promptHistoryTypes.type.js';
+ import type { ToolCallNoticeApi } from '../usePromptToolCallNotice.hook.js';
+
+ const startMock = jest.fn<
+   (
+     args: Extract<AiStreamStartArgs, { kind: 'promptQuery' }>,
+   ) => AiStreamStartResult
+ >();
+ const resetLastReceivedStreamKindMock = jest.fn<() => void>();
+ const toolCallNoticeMock = {
+   ingestToolCalls: jest.fn(),
+   ensureGapAfterNotice: jest.fn(),
+   finalizeActiveNoticeOnFirstMessageToken: jest.fn(),
+   finalizeActiveNoticeOnFirstThinkingToken: jest.fn(),
+   finalizeActiveNoticeAtTurnEnd: jest.fn(),
+   reset: jest.fn(),
+   webSearchOpenIfNeeded: jest.fn(),
+   webSearchAddItem: jest.fn(),
+   webSearchSetResults: jest.fn(),
+   webSearchMarkError: jest.fn(),
+   markWriteOrPatchErrorByPath: jest.fn(),
+   ciOpenItem: jest.fn(),
+   ciAppendCode: jest.fn(),
+   ciAppendOutputs: jest.fn(),
+   ciAppendFile: jest.fn(),
+   ciFinalize: jest.fn(),
+   applyToolCallsUpdatePayload: jest.fn(),
+ } satisfies ToolCallNoticeApi;
+
+ jest.unstable_mockModule('../usePromptAiQueryOptionsBase.hook.js', () => ({
+   usePromptAiQueryOptionsBase: () => ({
+     reasoning: 'medium',
+     verbosity: 'low',
```

```

+     })),
+   }));
+
+   const { usePromptRunCommands } = await import(
+     '../usePromptRunCommands.hook.js'
+   );
+
+   const sessionLocator = {
+     sessionId: 'a'.repeat(32),
+     storageScopeRelPath: '',
+   } as const;
+
+   /** Create an isolated hook runtime exposing both command results and every mutated
+   continuation surface. */
+   function useContinuationHarness() {
+     const [messages, setMessages] = useState<PromptMessage[]>([
+       { turnId: 'turn', uid: 'user', role: 'user', text: 'Question' },
+       {
+         turnId: 'turn',
+         uid: 'assistant',
+         role: 'assistant',
+         text: 'Partial answer',
+         fileEditsUiFinalized: true,
+       },
+     ]);
+     const [uiPending, setUiPending] = useState(false);
+     const [streamingThinkingTokens, setStreamingThinkingTokens] = useState([
+       'stale-thinking',
+     ]);
+     const [streamingTokens, setStreamingTokens] = useState(['stale-text']);
+     const streamingMessageUidRef = useRef<string | undefined>(undefined);
+     const streamingThinkingUidRef = useRef<string | undefined>(undefined);
+     const currentTurnIdRef = useRef<string | undefined>(undefined);
+     const baseAssistantTextRef = useRef('');
+     const lastConsumedRowIndexRef = useRef(7);
+     const metaOnlyFinalizedTurnIdRef = useRef<string | undefined>('old-turn');
+     const lastMetaChunkFinalizedRef = useRef<AiStreamedChunk | undefined>(
+       undefined,
+     );
+
+     const commands = usePromptRunCommands({
+       sessionLocator,
+       dbSettingsExtUiPrompt: {
+         autoScroll: true,
+         paginationEnabled: false,
+       },
+       start: startMock,
+       messages,
+       setMessages,
+       setUiPending,
+       setStreamingThinkingTokens,
+       setStreamingTokens,
+       streamingMessageUidRef,
+       streamingThinkingUidRef,
+       currentTurnIdRef,
+       baseAssistantTextRef,
+       lastConsumedRowIndexRef,
+       toolCallNotice: toolCallNoticeMock,
+       metaOnlyFinalizedTurnIdRef,
+       lastMetaChunkFinalizedRef,
+       resetLastReceivedStreamKind: resetLastReceivedStreamKindMock,
+     });
+
+     return {

```



```

+         ...commands,
+         messages,
+         uiPending,
+         streamingThinkingTokens,
+         streamingTokens,
+         streamingMessageUidRef,
+         streamingThinkingUidRef,
+         currentTurnIdRef,
+         baseAssistantTextRef,
+         lastConsumedRawIndexRef,
+         metaOnlyFinalizedTurnIdRef,
+     };
+ }
+
+ describe('Continue response command integration', () => {
+     beforeEach(() => {
+         jest.clearAllMocks();
+         startMock.mockReturnValue({
+             accepted: true,
+             runRef: {
+                 locator: sessionLocator,
+                 turnId: 'turn',
+                 sourceId: 'source',
+             },
+             streamInstanceId: 'stream',
+         });
+     });
+
+     it('commits the pure continuation plan exactly once after synchronous start acceptance', () => {
+         const { result } = renderHook(() => useContinuationHarness());
+         let startResult: AiStreamStartResult | undefined;
+
+         -
+         - import { preparePromptContinuationStreaming } from
+         - '../preparePromptContinuationStreaming.util.js';
+         -
+         +         act(() => {
+         +             startResult = result.current.handleContinueResponse();
+         +         });
+         +
+         -
+         - describe('Continue response plan', () => {
+         -     it('reuses logical turn with a fresh thinking attempt and no duplicate assistant',
+         - ) => {
+         -         const plan = preparePromptContinuationStreaming([
+         -             { turnId: 'turn', uid: 'u', role: 'user', text: 'Q' },
+         -             { turnId: 'turn', uid: 'a', role: 'assistant', text: 'A' },
+         -         ]);
+         -         expect(plan.turnId).toBe('turn');
+         -
+         +         expect(startResult).toEqual(
+         +             expect.objectContaining({
+         +                 accepted: true,
+         +                 streamInstanceId: 'stream',
+         +             }),
+         +         );
+         +         expect(startMock).toHaveBeenCalledWith({
+         +             kind: 'promptQuery',
+         +             turnId: 'turn',

```

```

+         storageScopeRelPath: '',
+         aiQuery: { type: 'continueResponse' },
+         aiQueryOptions: { reasoning: 'medium', verbosity: 'low' },
+     });
+     expect(result.current.uiPending).toBe(true);
+     expect(result.current.streamingThinkingTokens).toEqual([]);
+     expect(result.current.streamingTokens).toEqual([]);
+     expect(result.current.lastConsumedRawIndexRef.current).toBe(0);
+     expect(result.current.baseAssistantTextRef.current).toBe(
+         'Partial answer',
+     );
+     expect(result.current.currentTurnIdRef.current).toBe('turn');
+     expect(result.current.streamingMessageUidRef.current).toBe(
+         'assistant',
+     );
+     expect(result.current.streamingThinkingUidRef.current).toBeDefined();
+
+
+     expect(
+         plan.messages.filter((message) => message.uid === 'a'),
+     );
+
+     expect(
+         result.current.messages.filter(
+             (message) => message.uid === 'assistant',
+         ),
+     );
+
+
+     ).toHaveLength(1);
+     expect(plan.thinkingUid).not.toBe('a');
+
+
+     ).toHaveLength(1);
+     expect(result.current.messages[1]).toEqual(
+         expect.objectContaining({
+             role: 'assistant',
+             assistantBubbleKind: 'think',
+         }),
+     );
+     expect(result.current.messages[2]).toEqual(
+         expect.objectContaining({
+             uid: 'assistant',
+             fileEditsUiFinalized: false,
+         }),
+     );
+     expect(result.current.metaOnlyFinalizedTurnIdRef.current).toBeUndefined();
+     expect(toolCallNoticeMock.reset).toHaveBeenCalledTimes(1);
+     expect(resetLastReceivedStreamKindMock).toHaveBeenCalledTimes(1);
+ });
+
+ it('keeps every visible state and streaming ref untouched when start is refused',
+ () => {
+     startMock.mockReturnValue({ accepted: false });
+     const { result } = renderHook(() => useContinuationHarness());
+     const initialMessages = result.current.messages;
+     let startResult: AiStreamStartResult | undefined;
+
+     act(() => {
+         startResult = result.current.handleContinueResponse();
+     });
+
+     expect(startResult).toEqual({ accepted: false });
+     expect(result.current.messages).toBe(initialMessages);
+     expect(result.current.uiPending).toBe(false);
+     expect(result.current.streamingThinkingTokens).toEqual([
+         'stale-thinking',

```

```

+     });
+     expect(result.current.streamingTokens).toEqual(['stale-text']);
+     expect(result.current.streamingMessageUidRef.current).toBeUndefined();
+     expect(result.current.streamingThinkingUidRef.current).toBeUndefined();
+     expect(result.current.currentTurnIdRef.current).toBeUndefined();
+     expect(result.current.baseAssistantTextRef.current).toBe('');
+     expect(result.current.lastConsumedRawIndexRef.current).toBe(7);
+     expect(result.current.metaOnlyFinalizedTurnIdRef.current).toBe(
+       'old-turn',
+     );
+     expect(toolCallNoticeMock.reset).not.toHaveBeenCalled();
+     expect(resetLastReceivedStreamKindMock).not.toHaveBeenCalled();
+

```

[AI Code/src/frontend/src/app/prompt/Prompt/ test /usePromptRunCommands.continueResponse.test.tsx](#)

Copy

Review

Accept

  reading project:  e ["Satisfies API mocks"](#)

  patching file:  [Prompt.autoRunPrompt.test.tsx](#)

[AI Code/src/frontend/src/app/prompt/Prompt/ test /Prompt.autoRunPrompt.test.tsx](#)

Copy

Review

Accept

PatchFull file

```

- jest.unstable_mockModule('../..lib/usePromptSessionLocator.hook.js', () => ({
-
+ jest.unstable_mockModule('../..lib/usePromptSessionLocator.hook.js', () => ({
+   usePromptRouterQueryStringRawSnapshot: () => {
+     const params = new URLSearchParams();
+     for (const [name, value] of queryValues) {
+       params.append(name, value);
+     }
+     if (!params.has('storageScopeRelPath')) {
+       params.append(
+         'storageScopeRelPath',
+         sessionLocatorMock.storageScopeRelPath,
+       );
+     }
+     return params.toString();
+   },
+

```

```

-     queryValues.clear();
-     queryValues.set('sessionId', 'session-123');
-
+     queryValues.clear();
+     queryValues.set('sessionId', sessionLocatorMock.sessionId);
+     queryValues.set(
+       'storageScopeRelPath',
+       sessionLocatorMock.storageScopeRelPath,
+     );
+

```

[AI Code/src/frontend/src/app/prompt/Prompt/ test /Prompt.autoRunPrompt.test.tsx](#)

Copy

Review

Accept

  patching file:  [Prompt.promptDraft.hydration.test.tsx](#)

Review

Accept

PatchFull file

```
- jest.unstable_mockModule('../..../lib/usePromptSessionLocator.hook.js', () => ({  
-  
+ jest.unstable_mockModule('../..../lib/usePromptSessionLocator.hook.js', () => ({  
+   usePromptRouterQueryStringRawSnapshot: () =>  
+     `sessionId=${sessionLocatorMock.sessionId}&storageScopeRelPath=${  
{encodeURIComponent(sessionLocatorMock.storageScopeRelPath)}`,  
+  
+  
AICode/src/frontend/src/app/prompt/Prompt/ test /Prompt.ensureNextTurnId.stability.test.tsx
```

Copy

Review

Accept

  reading project:  e ["Mocks locator incomplets"](#),
 [stream.endpoint.test.ts](#),
 [iaStreamHandlers.queryDoneSound.test.ts](#),
 [usePromptRunCommands.continueResponse.test.tsx](#),  [NormalPromptLiveSessionSource.service.ts](#) [L1-L300],  [NormalPromptLiveSessionSource.service.test.ts](#) [L1-L260]
  scanning git:  "Écarts correction actuels"
  reading project:  d ["OrchAskStream"](#),  [OrchAskStream.ts](#) [L1-L220],  [OrchMemory.ts](#) [L640-L700]
  patching file:  [iaStreamHandlers.queryDoneSound.test.ts](#)
[AICode/src/extension/src/messageBus/handlers/ia/ test /iaStreamHandlers.queryDoneSound.test.ts](#)
Grouped
([what is it?](#))

  reading project:  e ["Service source branché"](#),  e ["Helper continuation branché"](#),  e ["Snapshot brut branché"](#),  e ["Append handler dupliqué"](#),  e ["Baseline handler dupliquée"](#),  e ["Double cast commandes"](#),  e ["Fichiers générés modifiés"](#)
  patching file:  [stream.endpoint.test.ts](#)
[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)
Grouped
([what is it?](#))

  patching file:  [usePromptRunCommands.continueResponse.test.tsx](#)
[AICode/src/frontend/src/app/prompt/Prompt/ test /usePromptRunCommands.continueResponse.test.tsx](#)
Grouped
([what is it?](#))

  scanning recent files
  reading project:  e ["Démarrage backend local"](#),  [callLocalApi.generated.ts](#) [L1-L80],  [usePromptSessionLocator.hook.ts](#) [L1-L80],  [delete.endpoint.ts](#) [L45-L125]
  patching file:  [delete.endpoint.ts](#)
[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)
Grouped
([what is it?](#))

  patching file:  [delete.endpoint.liveSession.test.ts](#)
[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/ test /delete.endpoint.liveSession.test.ts](#)
Grouped

([what is it?](#))



patching file: [stream.endpoint.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/ test /stream.endpoint.test.ts](#)

Grouped

([what is it?](#))



reading project: [Prompt.autoRunPrompt.test.tsx](#) [L355-L395], [Prompt.autoRunPrompt.test.tsx](#) [L455-

L485], [Prompt.promptDraft.hydratation.test.tsx](#) [L305-

L322], [Prompt.editLastPrompt.noThumbTurnIdDrift.test.tsx](#) [L140-

L155], [Prompt.editLastPrompt.restoreAttachments.test.tsx](#) [L103-

L118], [Prompt.ensureNextTurnId.stability.test.tsx](#) [L100-L115], e ["Règle longueur lignes"](#), [Prompt.autoRunPrompt.test.tsx](#) [L485-L510]



patching file: [SidebarStore.rehydrate.manager.ts](#)

[AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts](#)

Copy

Review

Accept

PatchFull file

```
- } from './SidebarStore.chatTitles.manager.js';
- import {
-
+ } from './SidebarStore.chatTitles.manager.js';
+ import {
+   isSameQueryActivityLocator,
+
- } from './SidebarStore.queryWindow.listeners.manager.js';
-
+ } from './SidebarStore.queryWindow.listeners.manager.js';
+ import {
+   forceRemoveQueryActivity,
+   panelClosed,
+ } from './SidebarStore.queryWindow.mutations.manager.js';
+

-
- /** Rebuild Prompt activities from open panels plus active records/reservations. */
-
+ /**
+  * Reconcile one now-closed Prompt activity after a failed destructive operation
+  * releases its claim.
+  * Active source authority remains visible as Running/source/closed; every non-source
+  * projection is removed.
+  */
+ export async function reconcileClosedQueryActivityFromLiveAuthority(
+   locator: ChatSessionLocator,
+ ): Promise<void> {
+   const source = getLiveSessionServiceForProd()
+     .listActiveSnapshots()
+     .find(
+       (snapshot) =>
+         isSameQueryActivityLocator(snapshot.runRef.locator, locator) &&
+         (snapshot.kind === 'reservation' ||
+           (snapshot.ownership === 'source' &&
+             !snapshot.sourceSettled &&
+             !snapshot.handoffReady)),
+     );
+   if (!source) {
```

```

+      // Closed idle/observer activities are invalid and must not survive a failed
delete attempt.
+      await forceRemoveQueryActivity(locator);
+      return;
+    }
+
+    const runtime = sidebarQueryRunningBridge.getSnapshot(locator);
+    const runtimeStopRequested =
+      runtime !== undefined &&
+      runtime.runRef.turnId === source.runRef.turnId &&
+      runtime.runRef.sourceId === source.runRef.sourceId &&
+      runtime.stopRequested;
+    const stopRequested =
+      source.abortRequested || runtimeStopRequested;
+    const current =
+      sidebarStore.sidebarContentState.activityState.queryState.items.find(
+        (item) => isSameQueryActivityLocator(item, locator),
+      );
+
+    if (
+      current?.status === 'running' &&
+      current.ownership === 'source' &&
+      current.runRef.turnId === source.runRef.turnId &&
+      current.runRef.sourceId === source.runRef.sourceId
+    ) {
+      // Preserve elapsed time, usage, title, and openedAt when the exact projection
still exists.
+      await panelClosed({
+        locator,
+        runRef: source.runRef,
+        stopRequested,
+      });
+      return;
+    }
+
+    // Rebuild a missing or stale projection from the registry/runtime authorities
only.
+    await sourceStarted({
+      runRef: source.runRef,
+      ownership: 'source',
+      panelState: 'closed',
+      sourceStartedAtMs: source.sourceStartedAtMs,
+      stopRequested,
+    });
+  }
+
+  /** Rebuild Prompt activities from open panels plus active records/reservations. */
+

```

[AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts](#)

Copy

Review

Accept

  *reading project:*  [delete.endpoint.ts](#) [L1-L125],  [delete.endpoint.liveSession.test.ts](#) [L1-L180]

  *patching file:*  [delete.endpoint.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)

Grouped

([what is it?](#))

  *patching file:*  [delete.endpoint.liveSession.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/ test /delete.endpoint.liveSession.test.ts](#)

Error

([what is it?](#))

  reading project:  [delete.endpoint.liveSession.test.ts](#)

  patching file:  [delete.endpoint.liveSession.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/ test /delete.endpoint.liveSession.test.ts](#)

Grouped

([what is it?](#))

  reading project:  [SidebarStore.rehydrate.manager.test.ts](#)

  patching file:  [SidebarStore.rehydrate.manager.test.ts](#)

[AICode/src/extension/src/extension/panels/sidebar/ test /SidebarStore.rehydrate.manager.test.ts](#)

Copy

Review

Accept

PatchFull file

```
- const { sidebarStore } = await import('../SidebarStore.core.manager.js');
- const { rehydrateQueryActivities } =
-   await import('../SidebarStore.rehydrate.manager.js');
-
+ const { sidebarStore } = await import('../SidebarStore.core.manager.js');
+ const {
+   reconcileClosedQueryActivityFromLiveAuthority,
+   rehydrateQueryActivities,
+ } = await import('../SidebarStore.rehydrate.manager.js');
+
-   );
-   });
-
+   );
+   });
+
+   it('rebuilds a missing closed source projection from live authority after delete
failure', async () => {
+     const locator = {
+       sessionId: 'c'.repeat(32),
+       storageScopeRelPath: '',
+     };
+     listActiveSnapshotsMock.mockReturnValue([
+       {
+         kind: 'reservation',
+         runRef: {
+           locator,
+           turnId: 'turn-c',
+           sourceId: 'source-c',
+         },
+         sourceStartedAtMs: 10,
+         abortRequested: true,
+       },
+     ]);
+
+     await reconcileClosedQueryActivityFromLiveAuthority(locator);
+
+     expect(
+       sidebarStore.sidebarContentState.activityState.queryState.items,
+     ).toEqual([
+       expect.objectContaining({
```



```

+         sessionId: locator.sessionId,
+         status: 'running',
+         ownership: 'source',
+         panelState: 'closed',
+         stopRequested: true,
+     })),
+ ]);
+ });
+
+ it('closes an existing exact source projection without losing its usage or
openedAt', async () => {
+     const locator = {
+         sessionId: 'e'.repeat(32),
+         storageScopeRelPath: '',
+     };
+     const runRef = {
+         locator,
+         turnId: 'turn-e',
+         sourceId: 'source-e',
+     };
+     const usage = {
+         query_time: {
+             start_ts_utc: 1,
+             end_ts_utc: 2,
+             duration_ms: 1,
+         },
+     };
+     sidebarStore.sidebarContentState.activityState.queryState.items = [
+         {
+             status: 'running',
+             ownership: 'source',
+             panelState: 'open',
+             sessionId: locator.sessionId,
+             storageScopeRelPath: locator.storageScopeRelPath,
+             title: 'Running chat',
+             runRef,
+             liveQueryElapsedMs: 12,
+             usage,
+             stopRequested: false,
+             updatedAt: 3,
+             openedAt: 4,
+         },
+     ];
+     listActiveSnapshotsMock.mockReturnValue([
+         {
+             kind: 'record',
+             runRef,
+             ownership: 'source',
+             sourceStartedAtMs: 1,
+             abortRequested: false,
+             sourceSettled: false,
+             handoffReady: false,
+         },
+     ]);
+
+     await reconcileClosedQueryActivityFromLiveAuthority(locator);
+
+     expect(
+         sidebarStore.sidebarContentState.activityState.queryState.items[0],
+     ).toEqual(
+         expect.objectContaining({
+             panelState: 'closed',
+             usage,

```

```

+         openedAt: 4,
+       })),
+     );
+   });
+
+   it('removes a closed non-source projection when no live authority remains', async
+ ) => {
+     const locator = {
+       sessionId: 'd'.repeat(32),
+       storageScopeRelPath: '',
+     };
+     sidebarStore.sidebarContentState.activityState.queryState.items = [
+       {
+         status: 'idle',
+         panelState: 'open',
+         sessionId: locator.sessionId,
+         storageScopeRelPath: locator.storageScopeRelPath,
+         title: 'Idle chat',
+         updatedAt: 1,
+         openedAt: 1,
+       },
+     ];
+
+     await reconcileClosedQueryActivityFromLiveAuthority(locator);
+
+     expect(
+       sidebarStore.sidebarContentState.activityState.queryState.items,
+     ).toEqual([]);
+   });
+
+

```

[AICode/src/extension/src/extension/panels/sidebar/ test /SidebarStore.rehydrate.manager.test.ts](#)

Copy

Review

Accept

  reading

project:

 [delete.endpoint.ts](#),

 [SidebarStore.rehydrate.manager.ts](#),  [delete.endpoint.liveSession.test.ts](#),  [SidebarStore.rehydrate.manager.test.ts](#)

  patching file:  [SidebarStore.rehydrate.manager.ts](#)

[AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts](#)

Grouped

([what is it?](#))

  reading project:  e ["Acquisition claim destructif"](#),  [LiveSessionRegistry.service.ts](#) [L820-L875]

  patching file:  [SidebarStore.rehydrate.manager.test.ts](#)

[AICode/src/extension/src/extension/panels/sidebar/ test /SidebarStore.rehydrate.manager.test.ts](#)

Grouped

([what is it?](#))

  reading project:  e ["Tests endpoint suppression"](#),  e ["Mocks liveSession suppression"](#),  e ["Appels delete chat API"](#),  [diskAliases.lastWins.endpoints.test.ts](#) [L1-L240],  e ["Règle throw Error"](#)

  patching file:  [delete.endpoint.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)

Grouped

([what is it?](#))

  reading project:  [Prompt.tsx](#) [L720-L820]

  patching file:  [delete.endpoint.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts](#)

Grouped

([what is it?](#))

  reading project:  e ["Définition FunStreamChunk"](#),  e ["Verbatim module syntax"](#)

Corrections complètes appliquées

Les **quatre anomalies identifiées pendant la vérification** sont maintenant corrigées. L'implémentation utilise désormais réellement les abstractions prévues par la spécification au lieu de seulement les exposer à travers des tests isolés.

Zone	Corrections réalisées
Source Prompt normale	Le service source est désormais traversé par le chemin de production : capture de baseline, priming du <code>Readable</code> , promotion de réservation, persistance, append et forwarding ordonné. NormalPromptLiveSessionSource.service.ts , stream.e
Réservation exacte	Ajout d'un pont process-local à consommation unique. Le <code>handle opaque</code> n'est jamais sérialisé et seul le handler ayant obtenu la réservation peut alimenter l'endpoint source. NormalPromptLiveSessionSource.service.ts
Priming et Stop précoce	Une erreur synchrone de priming ne tente plus d'appendre sur une simple réservation. Un Stop accepté pendant la phase réservée est réappliqué après la promotion du stream primé. NormalPromptLiveSessionSource.service.ts
Handler extension	Suppression de la duplication de baseline, persistance et append. Le handler se concentre maintenant sur la iaStreamHandlers.ts FIFO auxiliaire, Activité, son, émission frontend et finalisation post-forwarding.
Protocole d'erreur	Le premier <code>meta.error</code> durable utilise toujours le header privé <code>V1</code> ; les erreurs tardives restent des chunks du body après le header <code>200_OK</code> . stream.endpoint.ts
Suppression destructive	Une suppression disque échouée ne peut plus faire disparaître une source active de l'Activité. La projection est réconciliée après libération du delete.endpoint.ts , SidebarStore.rehydrate.manager

Zone	Corrections réalisées	
	claim à partir des autorités registre/runtime. Une source encore active redevient exactement <code>Running/source</code>	
États Activité	<code>/closed</code> . Un état Idle ou observer fermé est supprimé. Une projection exacte existante conserve usage, elapsed, titre et <code>openedAt</code> . Le bootstrap Prompt consomme maintenant	SidebarStore.rehydrate.manager.ts
Snapshot brut Prompt	directement le snapshot brut figé fourni par le hook dédié. La reconstruction scalaire partielle a été supprimée. Le chemin Continue utilise réellement le plan pur dédié. Le scaffold n'est committé	Prompt.tsx , usePromptSessionLocator.hook.ts
Continuation frontend	qu'après acceptation synchrone du transport et le double cast de <code>PromptMessage</code> a disparu de ce chemin. Un refus de <code>start()</code> ne	usePromptRunCommands.hook.ts , preparePromptContinu
Refus de démarrage	modifie ni messages, ni buffers, ni refs, ni <code>uiPending</code> , ni état de finalisation. Les tests vérifient maintenant le passage réel endpoint → service source, l'ordre priming	usePromptRunCommands.continueResponse.test.tsx
Tests de câblage	→ promotion → append → forwarding, la réservation opaque, l'échec de priming, le Stop précoce et la récupération Activité. Aucun	stream.endpoint.test.ts , NormalPromptLiveSessionSc
Périmètre protégé	fichier <code>*.generated.*</code> et aucun document personnel interdit n'a été modifié par ces corrections.	—

Résultat

- **18 fichiers** ont été ajustés.
- Les abstractions centrales précédemment mortes sont maintenant **branchées au runtime réel**.
- La source Prompt normale respecte désormais le cycle **réservation** → **priming** → **promotion** → **durabilité** → **append** → **forwarding** → **finalisation**.
- Les suppressions échouées reconstruisent une projection Activité cohérente au lieu de produire une source invisible.
- Le bootstrap Prompt utilise une unique query brute figée.
- Continue suit le même contrat transactionnel strict que les autres commandes.

Nom de commit

text

Copy

fix(live-session): wire prompt source lifecycle and recover activity state

🎵 La source reste en vie 🚫 Le replay avance sans copie 🎵

</ASSISTANT>

<USER>

Summary of all failing tests FAIL src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts • chat delete live-session atomicity › does not evict on durable delete failure and still releases the claim expect(received).toEqual(expected) // deep equality - Expected - 1 + Received + 0 Array ["close", - "delete", "release", "reconcile",] 162 | expect(releaseMock).toHaveBeenCalledWith(claim); 163 | expect(reconcileActivityMock).toHaveBeenCalledWith(runRef.locator); > 164 | expect(order).toEqual(['close', 'delete', 'release', 'reconcile']); | ^ 165 | }); 166 | 167 | it('removes a closed idle projection before durable deletion while holding the claim', async () => { at Object.<anonymous> (src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts:164:23) FAIL src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts • Test suite failed to run SyntaxError: The requested module './SidebarStore.queryWindow.mutations.manager.js' does not provide an export named 'forceRemoveQueryActivity' 53 | 54 | const { SidebarQueryActivityRuntimeService } => 55 | await import('./SidebarQueryRunningBridge.js'); | ^ 56 | 57 | const LOCATOR = { 58 | sessionId: 'a.repeat(32), at EsmLoader.tryLoadGraphSync (node_modules/jest-runtime/build/index.js:725:18) at src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts:55:5 Test Suites: 2 failed, 822 passed, 824 total Tests: 1 failed, 2646 passed, 2647 total Snapshots: 1 passed, 1 total Time: 66.562 s Ran all test suites. Command failed: C:\Program Files\nodejs\node.exe --expose-gc --max-old-space-size=8192 --experimental-vm-modules C:\dev\AICode\src\extension\node_modules\jest\bin\jest.js --config C:\dev\AICode\src\extension\jest.config.ts --maxWorkers 10 (exit 1)

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

LLM-DEBUG-3 — Analyse des deux échecs

Les deux traces pointent vers des **tests devenus incohérents avec leurs propres mocks**, pas vers une régression fonctionnelle de l'application. Je vérifie les implémentations exactes avant de corriger uniquement les tests concernés — le mock a glissé sur une peau de banane, le runtime n'y est pour rien 🍌.

🔧  **reading project:**  [delete.endpoint.liveSession.test.ts](#) [L1-L220],  [delete.endpoint.ts](#) [L1-L180],  [SidebarQueryRunningBridge.race.test.ts](#) [L1-L130],  [SidebarStore.rehydrate.manager.ts](#) [L1-L150],  [SidebarStore.queryWindow.mutations.manager.ts](#) [L1-L180],  [SidebarQueryRunningBridge.ts](#) [L1-L100]

🔧  **patching file:**  [delete.endpoint.liveSession.test.ts](#)

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts](#)

Copy

Review

Accept

PatchFull file

```
-      it('does not evict on durable delete failure and still releases the claim', async
() => {
-        deleteMock.mockRejectedValueOnce(new Error('disk failed'));
-
+      it('does not evict on durable delete failure and still releases the claim', async
() => {
+        deleteMock.mockImplementationOnce(async () => {
+          // Preserve the observable invocation marker while making this exact
durable write fail.
+          order.push('delete');
+          throw new Error('disk failed');
```

+ });

+

[AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/ test /delete.endpoint.liveSession.test.ts](#)

Copy

Review

Accept



patching file: [SidebarQueryRunningBridge.race.test.ts](#)

[AICode/src/extension/src/messageBus/handlers/ia/ test /SidebarQueryRunningBridge.race.test.ts](#)

Copy

Review

Accept

PatchFull file

```
-  
'../../../../extension/panels/sidebar/SidebarStore.queryWindow.mutations.manager.js',  
-    () => {  
-  
+  
'../../../../extension/panels/sidebar/SidebarStore.queryWindow.mutations.manager.js',  
+    () => {  
+        // SidebarStore core also loads the reconciliation module, which statically  
+        // imports this export.  
+        forceRemoveQueryActivity: async () => {},  
+  
+  

```

-

+

```
'../../../../extension/panels/sidebar/SidebarStore.queryWindow.mutations.manager.js',  
+    () => {  
+        // SidebarStore core also loads the reconciliation module, which statically  
+        // imports this export.  
+        forceRemoveQueryActivity: async () => {},  
+  
+  

```

```
+        forceRemoveQueryActivity: async () => {},  
+  
+  

```

+

[AICode/src/extension/src/messageBus/handlers/ia/ test /SidebarQueryRunningBridge.race.test.ts](#)

Copy

Review

Accept



reading project: [delete.endpoint.liveSession.test.ts](#) [L135-

L175], [SidebarQueryRunningBridge.race.test.ts](#) [L35-L65], [package.json](#) [L1-L100]

LLM-DEBUG-3 — Les deux échecs sont corrigés

Les traces confirmaient **deux défauts dans les tests**, pas dans le code applicatif.

1. Échec sur l'ordre de suppression durable

Dans [delete.endpoint.liveSession.test.ts](#), le test remplaçait l'implémentation du mock par `mockRejectedValueOnce(...)`.

Cette substitution supprimait involontairement l'instruction du mock de base qui enregistrait "delete" dans le tableau `order`. L'endpoint appelait bien la suppression durable, mais **le test supprimait lui-même son marqueur d'observation** avant de vérifier sa présence.

La branche d'échec utilise maintenant une implémentation qui :

1. enregistre "delete" ;
 2. lève ensuite l'erreur disque attendue.
- L'ordre vérifié reste donc correctement :

text

Copy

close → delete → release → reconcile

2. Namespace ESM incomplet dans le test Activité

Dans [SidebarQueryRunningBridge.race.test.ts](#), le mock ESM

de [SidebarStore.queryWindow.mutations.manager.ts](#) n'exposait pas `forceRemoveQueryActivity`.

Or [SidebarStore.rehydrate.manager.ts](#) importe désormais statiquement cette fonction. Avec les modules ESM, Jest valide le namespace complet au chargement du graphe, même si ce chemin n'est pas invoqué par le scénario testé.

Le mock expose maintenant une fonction asynchrone neutre pour satisfaire le véritable contrat du module, sans modifier le comportement du test de course.

Résumé	Test concerné	Cause racine	Origine	Corrigé
Suppression	<code>liveSession</code>	<code>mockRejectedValueOnce</code> écrasait le marqueur "delete"	Test	Mock rejetant après l'appel
Course	<code>SidebarQueryRunningBridge</code>	Namespace ESM incomplet	Test	Ajout de <code>forceRemove</code> mock
Code applicatif		Aucun défaut révélé par ces deux traces	Application	Aucun changement
Portée des changements		Deux fichiers de test uniquement	Tests	Modifications mineures
Commande de vérification ciblée				
Depuis src/extension/ :				

```
powershell
Copy
node --expose-gc --max-old-space-size=8192 --experimental-vm-modules
./node_modules/jest/bin/jest.js --config ./jest.config.ts --runInBand
src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts
src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts
```

Relancez ensuite la suite complète et transmettez-moi tout nouvel échec éventuel afin de poursuivre la boucle **LLM-DEBUG-3**.

Nom de commit

```
text
Copy
test(live-session): restore delete ordering and complete activity mocks
```

🎵 Les mocks suivent enfin le contrat 🦋 Deux tests réparés, le runtime ne bouge pas 🎵

</ASSISTANT>

SHOW ALLPage 1 / 1