

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=78ce8dd6f8a546c38dc64ffa16d2bce7-VS-V%C3%A9rifier-session-<USER>

[Verify]SPECIF FINALISER LIVESESSION POUR LES SESSIONS PROMPT NORMALES 20260712 204731</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧  reading project: 📁 g "[Docs liveSession](#)", 📁 g "[Docs streaming Prompt](#)", 📁 g "[Docs Prompt ciblées](#)"

🔧  scanning recent files

🔧  scanning git: 📁 "État Git courant", 📁 "Révision Git courante"

🔧  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](#)

🔧  scanning git: 📁 "Périmètre du commit", 📁 "Volume du commit", 📁 "Diff source global"

🔧  reading project: 📄 h "[Contrats liveSession](#)", 📄 h "[Registre liveSession](#)", 📄 h "[Origine et transport](#)", 📄 h "[Identités frontend](#)", 📄 h "[Activité et panneaux](#)"

🔧  scanning git: 📁 "Périmètre fonctionnel", 📁 "Statistique fonctionnelle", 📁 "Résumé fonctionnel"

🔧  reading

project:

📄 [LiveSessionRun.type.ts](#),
📄 [LiveSessionPanel.type.ts](#),
📄 [LiveSessionOpenResult.type.ts](#),
📄 [LiveSessionReplay.type.ts](#),
📄 [parseSlimbkStreamBodyError.util.ts](#),
📄 [LiveSessionSourceLifecycle.service.ts](#),
📄 [NormalPromptLiveSessionSource.service.ts](#),
📄 [PromptLiveSessionBootstrap.adapter.ts](#),
📄 [NormalPromptLiveSession.adapter.ts](#), 📄 [AiStreamStartArgs.type.ts](#), 📄 [LiveSessionRecord.type.ts](#), 📁 "Cœur du registre", 📄 e "[Claim destructif](#)", 📄 e "[Réservation source](#)", 📄 e "[Éviction destructive](#)", 📄 [LiveSessionRegistry.service.ts](#) [L1-L430], 📄 [LiveSessionRegistry.service.ts](#) [L780-L1050], 📄 [LiveSessionOpen.service.ts](#), 📄 [LiveSessionReplay.service.ts](#), 📄 [LiveSessionService.ts](#) [L1-L260], 📄 [LiveSessionRegistry.service.ts](#) [L431-L779], 📄 [LiveSessionRegistry.service.ts](#) [L1050-L1206], 📄 [aiStopSuffix.util.ts](#), 📄 [OrchMessagesMemoryAssistantPatch.ts](#) [L1-L260], 📄 [SlimbkResult.type.ts](#), 📄 [Slimbk.ts](#) [L1-L260], 📄 [slimbkBuildHeader.ts](#), 📄 e "[Suffixe assistant](#)", 📄 e "[Header Slimbk stream](#)", 📄 e "[Erreur body Slimbk](#)", 📁 "[Suffixe et flux Slimbk](#)", 📄 [Slimbk.ts](#) [L320-L421], 📄 [extTypedMessageBus.util.ts](#), 📄 [extMessageBus.util.ts](#) [L260-L330],
📄 [iaStreamHandlers.ts](#),
📄 [liveSessionStreamHandlers.ts](#),
📄 [stream.endpoint.ts](#),
📄 [abortSource.endpoint.ts](#),
📄 [abort.endpoint.ts](#),
📄 [open.endpoint.ts](#),
📄 [stream.endpoint.ts](#),
📄 [PromptLiveSessionPanelQuery.util.ts](#),
📄 [PromptLiveSessionPanelCoordinator.service.ts](#),
📄 [PromptLiveSessionDetachedStop.service.ts](#),
📄 [SidebarQueryRunningBridge.ts](#), 📄 [SidebarStore.ts](#), 📄 [SidebarActivityQuery.tsx](#), 📄 [openPanel.util.ts](#) [L1-L360], 📄 [PromptThreadCleanup.manager.ts](#) [L1-L360], 📄 [openPanel.util.ts](#) [L361-L684],
📄 [SidebarStore.queryWindow.mutations.manager.ts](#),
📄 [SidebarStore.queryWindow.listeners.manager.ts](#),
📄 [SidebarStore.rehydrate.manager.ts](#), 📄 [SidebarStore.handlers.manager.ts](#), 📄 [SidebarUiMessages.ts](#), 📄 e "[Option](#)

[bouton désactiver](#)",  [VscButton.tsx](#) [L1-L260],  [VscButton.module.css](#) [L1-L220],  [useAiStream.hook.ts](#) [L1-L260],  [useAiStream.hook.ts](#) [L600-L900],  e ["Identité Queue A"](#),  e ["Identité Queue B"](#),  e ["Refus start sans mutation"](#),  e ["Finalisation des queues"](#),  [useAiStream.hook.ts](#) [L901-L1172],  e ["Retrait Stop transport"](#),  e ["Terminal consommé"](#),  e ["Traitement meta erreur"](#),  e ["Traitement stop reason"](#),  ["Traitement terminal"](#),
 [PromptLiveSession.types.ts](#),
 [usePromptLiveSession.hook.ts](#),
 [applyHydratedPromptSessionState.util.ts](#),
 [preparePromptContinuationStreaming.util.ts](#),
 [preparePromptStreamingTurn.util.ts](#),
 [Prompt.tsx](#),
 [usePromptRunCommands.hook.ts](#),
 [usePromptRunController.finalization.hook.ts](#),  [usePromptThinkingLifecycle.hook.ts](#),  [usePromptToolUse.hook.ts](#),
 [usePromptCancelController.hook.ts](#),  [usePromptRunController.hook.ts](#) [L1-L720],
 [delete.endpoint.ts](#),
 [EasyAgentsBatchRunRetry.service.ts](#),
 [EasyAgentsBatchLiveSessionAdapter.service.ts](#),
 [EasyAgentsBatchRunChildExecution.service.ts](#),
 [EasyAgentsBatchRunManager.service.ts](#),  [MessageRegistry.ts](#) [L1-L220],  [PromptQueryDoneSound.manager.ts](#),  [PromptInput.module.css](#),  e ["Bannière composer"](#),  e ["Stop unique composer"](#),  ["Intégration PromptInput"](#),
 [PromptLiveSessionBanner.tsx](#),
 [PromptLiveSessionBanner.module.css](#),
 [useAiStreamQueue.hook.ts](#),  [usePromptStreamDispatchGateQueue.hook.ts](#)
  scanning git:  ["Diff Queue A"](#),  ["Diff Queue B"](#),  ["Diff finalisation Prompt"](#)
  reading project:  h ["Tests finalisation stale"](#),  e ["Finalized ID tests"](#),  [usePromptIndexPending.hook.tsx](#)
  scanning git:  ["Diff garde index"](#),  ["Diff arrêt Slimbk"](#)
  reading project:  e ["Flag enregistrement source"](#),  e ["SourceId EasyAgents"](#),  e ["Handoff EasyAgents"](#),  ["Adaptation EasyAgents"](#),  e ["Transmission origine stream"](#),  [extMessageBus.util.ts](#) [L1-L150],  h ["Fallback et brouillon"](#),  e ["Ack fallback"](#),  [usePromptSessionLocator.hook.ts](#) [L1-L220],  [usePromptSessionLocator.hook.ts](#) [L220-L303]
  scanning git:  ["Tests ignorés liveSession"](#),  ["Marqueurs qualité ciblés"](#),  ["Intégrité du diff"](#),  ["Tests exclus exacts"](#),  ["Imports dynamiques ciblés"](#),  ["Require ciblé"](#)
  reading project:  [SidebarStore.core.manager.ts](#) [L1-L220],  h ["Tests fermeture replay"](#),  h ["Tests courses exactes"](#),  h ["Tests destruction"](#),  h ["Tests preserveLocal"](#),
 [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#),
 [useAiStream.hook.promptSourceIdentity.test.ts](#),
 [extMessageBus.streamOriginContext.test.ts](#),
 [parseSlimbkStreamBodyError.util.test.ts](#),  [usePromptRunController.liveSessionObserver.test.tsx](#) [L180-L271],  r ["openCentralPanel"](#),  [registerDeepLinkUriHandler.ts](#) [L125-L155],  e ["Ancien champ windows"](#),  e ["Ancien type activité"](#),  e ["Ancien streamKey"](#),  e ["Ancien canAbortSource"](#),  e ["Nouvel replay Open ID"](#),  e ["Textarea running disabled"](#),  [VscTextarea.tsx](#) [L1-L300],  [VscTextarea.tsx](#) [L300-L620],  [VscTextarea.tsx](#) [L620-L820],  [frontSessionHydration.type.ts](#) [L1-L180],  h ["Stop demandé réouverture"](#),  e ["Abort demandé dans Open"](#),  e ["Stop demandé bootstrap"](#)
  scanning git:  ["Imports expect frontend"](#)
  reading project:  [usePromptRunCommands.continueResponse.test.tsx](#)
  scanning git:  ["Expect dans nouveaux tests"](#)
  reading project:  [usePromptRunCommands.startAcceptance.test.tsx](#) [L1-L20]

Rapport de vérification

Joel, cette tranche est impressionnante : les courses asynchrones doivent désormais présenter trois pièces d'identité et un permis de séjour avant d'entrer 😊.

L’audit porte sur le **HEAD propre 2f65fc21fb88d87c4e323a63d4eef720342bb7a5**, dont le message correspond exactement au nom de commit demandé. J’ai réinspecté les contrats, les chemins source/replay, la coordination des panneaux, le transport Slimbk, l’Activité, les claims destructifs, le frontend Prompt et les tests associés. J’ai également relu les documents techniques concernés, notamment :

- [Architecture liveSession](#)
- [Architecture EasyAgents](#)
- [Invariant d’isomorphisme](#)
- [Invariant thinking/hourglass](#)
- [Queue B et verrou de dispatch](#)
- [Fallback persist et titre](#)

1) L’intégralité de la spécification est-elle implémentée ?

Question

L’intégralité de la spécification est-elle implémentée ?

Réponse

❌ Pas littéralement à 100 %.

L’implémentation principale est très largement présente et cohérente. Mon estimation est la suivante :

Axe vérifié	Estimation	Verdict
Comportement fonctionnel principal	98 %	✓ Presque complet
Architecture et contrats partagés	99 %	✓ Complet
Conformité stricte aux détails de la spécification	96 %	⚠ Quelques écarts
Couverture de tests explicitement exigée	89 %	⚠ Incomplète sur plusieurs scénarios contractuels
Conformité aux règles locales de test	98 %	⚠ Une violation confirmée
Ce qui empêche le verdict « 100 % »		

1. La finalisation Prompt n’est pas entièrement gardée par `streamInstanceId`.

Le hook de finalisation utilise bien l’identité pour publier `finalizedStreamInstanceId`, mais ses autres mutations — attachement de la meta, vidage des buffers et remise à zéro des références de cible — reposent encore sur les références mutables courantes.

2. Le test d’isomorphisme normal Prompt n’implémente pas toute la matrice demandée.

Il couvre bien replace, continuation, thinking, usage, `TOOL_USE` et erreurs canoniques, mais pas explicitement tous les types de requête, les edits/outils complets et le cas terminal suivi d’une erreur de transport dans cette même matrice d’isomorphisme.

3. Le test du contexte d’origine du message bus est trop local.

Il vérifie correctement le contrat du bus typé, mais injecte lui-même l’origine. Il ne traverse pas la construction réelle de l’origine par le wrapper de webview.

4. Un nouveau test frontend importe directement `expect` depuis `@jest/globals`.

Cela contrevient à la convention frontend stricte du dépôt.

Conclusion de la question 1

La fonctionnalité est livrée et largement fonctionnelle, mais la spécification stricte — tests et garde de finalisation compris — est implémentée à environ 96 %, pas à 100 %.

2) L’implémentation actuelle fait-elle ce qu’elle doit faire selon la spécification ?

Question

L’implémentation actuelle fait-elle ce qu’elle doit faire dans son état actuel, selon la spécification ? Pour chaque fonctionnalité technique demandée, est-elle complète ?

Réponse détaillée

Fonctionnalité technique demandée	État	Pré
Identité exacte { <code>locator</code> , <code>turnId</code> , <code>sourceId</code> }	✓ 100 %	Le contrat, la validation stricte et la run key incluant <code>sourceId</code> reste indépendante du <code>sourceId</code> .
Replay modes stricts	✓ 100 %	<code>replaceTurnAssistants</code> et <code>continueAssistant</code> sont stricts.
Descripteurs live complets et spécifiques aux adaptateurs	✓ 100 %	L’union stricte impose les politiques, la ligne EasyAgents est respectée.

Fonctionnalité technique demandée	État	Pre
Branches Open strictes, propriétés présentes à <code>undefined</code> rejetées	✓ 100 %	Le transport, le <code>superRefine</code> et l'enveloppe frontend à se
Parser brut des queries, doublons et paramètres inconnus	✓ 100 %	Les parsers extension et frontend utilisent les occurrences b
Query live explicite prioritaire et query invalide rejetée avant ouverture classique	✓ 100 %	Le coordinateur refuse l'invalidé avant le port d'ouverture ; run dans Prompt.tsx .
Snapshot brut figé et dispatcher pur	✓ 100 %	Le snapshot est lu une seule fois via usePromptSession
Démarrage synchrone transactionnel et refus sans mutation	✓ 100 %	Le single-flight est acquis avant génération de <code>sourceId</code> e commandes préparent, démarrent, puis committent dans us
Origine exacte par webview non sérialisée	✓ 100 % côté code	Le contexte extension-only est défini dans extTypedMess handlers comparent la webview exacte avant réservation.
Réservation avant tout <code>await</code> , promotion sans nouveau bump d'époque	✓ 100 %	La réservation, la promotion opaque, les tombstones et les l dans LiveSessionRegistry.service.ts .
Lifecycle lazy <code>Readable</code> partagé	✓ 100 %	Priming avant registration, consommation ordonnée, destru dans LiveSessionSourceLifecycle.service.ts et r
Baseline immédiate pour continuation et différée pour replace	✓ 100 %	La politique <code>baselineOnly</code> , la capture synchrone de conti réalisées dans NormalPromptLiveSessionSource.ser
Persistance de tout <code>meta.error</code> avant append et forwarding	✓ 100 %	<code>patchLastAssistantMeta</code> précède systématiquement l' suffixes <code>SYSTEM_ERROR/INTERRUPTED</code> sont idempot
Terminal synthétique uniquement sans terminal Orch antérieur	✓ 100 %	Un throw produit <code>INTERRUPTED</code> seulement sans terminal échec de persistance d'un terminal Orch ne provoque pas un
Protocole V1 uniquement pour le premier <code>meta.error</code>	✓ 100 %	L'encodage/décodage strict est dans aiStopSuffix.util
Erreur Slimbk post-header filtrée avant <code>AiStreamedChunk</code>	✓ 100 %	Le parseur dédié est parseSlimbkStreamBodyError.u chunk.
FIFO extension unique et erreurs auxiliaires non court-circuitantes	✓ 100 %	Le source handler et le replay handler sérialisent Activity, s dans iaStreamHandlers.ts et liveSessionStreamH
Finalisation post-forwarding avec cinq résultats exacts	✓ 100 %	Les résultats <code>reservationCancelled</code> , <code>handoffReady</code> , <code>ret</code> effectivement implémentés. L'absence de baseline reste prio
Buffer backlog + tail complet, ordonné et sans limite ajoutée	✓ 100 %	Le registre conserve le tableau complet et le replay utilise d
Claims destructifs, génération destructive et invalidation	✓ 100 %	Acquisition/release bumpent une génération indépendante ;
Open		
Coordination panneau par locator	✓ 100 %	Les queues par locator, les descripteurs provisionnels, revea dans PromptLiveSessionPanelCoordinator.service
Suppression synchrone du panneau avant dispose	✓ 100 %	Le registre de panneaux est retiré avant délégation au coord

Fonctionnalité technique demandée	État	Preuve
asynchrone		
Nettoyage différé sous claim et revalidation après chaque attente	✓ 100 %	Le mapping détaché est conservé dans PromptThreadCleaner .
Activité Idle/open, Running/observer/open, Running/source/open closed	✓ 100 %	L'union stricte et la migration <code>windows</code> → <code>items</code> sont préservées. L'accès <code>queryState.windows</code> ne subsiste.
Activité conservée si source fermée et supprimée à la fin	✓ 100 %	<code>panelClosed</code> conserve exclusivement les sources actives.
Stop détaché uniquement pour source fermée	✓ 100 % comportemental	L'UI dans SidebarActivityQuery.tsx impose les trois états. Elle est réactive uniquement sur refus : c'est plus robuste que <code>dispose</code> .
Propagation synchrone de <code>stopRequested</code>	✓ 100 %	Le runtime Activité est mis à jour avant l'abort dans SidebarActivityQuery.tsx .
Déduplication du son par <code>runRef</code> et tentative	✓ 100 %	<code>soundAttemptId</code> , <code>run</code> key exacte, terminal unique et nettoyage.
Suppression durable sous claim	✓ 100 %	Le chemin ferme le panneau, stoppe et attend les sources, sur Delete .
Retry EasyAgents destructif sous claim exact	✓ 100 %	Le claim est acquis après les préchecks, tenu durant <code>close/emit</code> en <code>finally</code> dans EasyAgentsBatchRunRetry.service.ts .
Préservation EasyAgents	✓ 100 %	<code>sourceId=turnId</code> , ownership observer, readiness historique dans backend EasyAgents .
Composer normal Prompt armé tôt, Send verrouillé, Stop unique	✓ 100 %	L'adaptateur dans NormalPromptLiveSession.adaptation.ts .
Retry <code>preserveLocal</code> sans lire les champs du brouillon	✓ 100 %	applyHydratedPromptSessionState.util.ts construit <code>promptDraft</code> , <code>promptDraftAttachments</code> ou <code>nextTurn</code> .
Handoff direct sans second Open	✓ 100 %	Le handoff exige terminal, outcome completed, instance créée dans usePromptLiveSession.hook.ts .
Terminal puis erreur de transport sans second terminal	✓ 100 %	<code>useAiStream</code> ne synthétise une erreur que si aucun terminus en <code>liveSessionReplayFailed</code> .
Fallback ack corrélé et démasquage sans effacement	✓ 100 %	La barrière { <code>turnId</code> , <code>streamInstanceId</code> } est armée pour que le masque.
Continuation et thinking par instance	✓ 100 %	Le plan de continuation réutilise l'assistant et crée une nouvelle dans preparePromptContinuationStreaming.util.ts .
Queue A/Queue B gardées par identité exacte	✓ 100 % sur l'algorithme de dispatch	Les algorithmes historiques de merge/gating restent en place, désormais l'identité exacte dans Queue A et Queue B .
Finalisation UI entièrement gardée par instance	△ 85 %	La publication de l'instance finalisée est exacte, mais toutes les cibles associées à la tentative dans hook de finalisation .
Tests exhaustifs exigés par la spécification	△ 89 %	La couverture est très large, mais certaines preuves contractuelles manquent.
Programming docs mises à jour	✓ 100 %	Les six documents ciblés décrivent les nouvelles identités, brouillon, etc.
Conclusion de la question 2		
✓ Oui pour le comportement produit principal : fermeture, continuation extension, Activité, réouverture, replay, Stop, handoff, persistance et nettoyage sont effectivement câblés de bout en bout.		
△ Non pour la conformité absolue : une garde frontend reste incomplète et certaines obligations de tests ne sont pas totalement réalisées.		

3) Y a-t-il des bugs ?

Question

Y a-t-il des bugs ?

Réponse

⚠ J'ai identifié un défaut de robustesse fonctionnel probable, une violation de règle de test confirmée et deux lacunes de protection contre les régressions.

Problème	Type	Gravité	Confiance	Analyse
Mutations de finalisation non corrélées à l'instance cible	Race frontend potentielle	Élevée	80 %	Dans usePromptRunController.finalization.ho la publication de <code>finalizedStreamInstanceId</code> , mais des buffers ni la remise à zéro des UIDs courants. Une fin théoriquement agir sur les refs déjà préparées pour B. Cela « toutes mutations gardées par <code>streamInstanceId</code> ».
Import direct de <code>expect</code> dans un nouveau test frontend	Violation confirmée des conventions	Faible à modérée	100 %	usePromptRunCommands.continueResponse.test globals, contrairement à la règle frontend du dépôt qui e de <code>@testing-library/jest-dom</code> .
Matrice d'isomorphisme moins complète que demandée	Couverture de tests	Modérée	100 %	test d'isomorphisme normal Prompt couvre replace, contin erreurs. Il ne compare pas explicitement toute la matrice d prompt/retry/edit/compact/SPECIF, notices et stores d'out suivi de <code>transportError</code> . Certaines briques sont testées to-end exigé.
Test d'origine exacte non end-to-end	Couverture de tests	Modérée	95 %	extMessageBus.streamOriginContext.test.ts p à <code>dispatchStreamedCall</code> . Il ne prouve donc pas qu' initializeViewWithMessageBus fabrique et trans webviews réelles.

Écart acceptable qui n'est pas un bug

Le bouton Stop détaché n'utilise pas la propriété `disableAfterClick`. Il utilise une ref synchrone et un état local, puis se réactive si le backend refuse la demande ou si le transport échoue.

C'est une amélioration, car :

- le double clic dans la même pile est bloqué ;
- l'acceptation désactive effectivement le bouton ;
- un Stop non consommé reste retentable ;
- aucun contrat strict de sécurité ou de typage n'est relâché.

Conclusion de la question 3

Je n'ai pas trouvé de rupture certaine dans le scénario principal fermeture → replay → handoff, mais la finalisation frontend n'applique pas encore totalement la protection d'identité exigée et mérite une correction prioritaire.

4) Quel est mon avis sur le code source et la spécification ?

Question

Quel est mon avis sur le code source et la spécification ?

Réponse

Élément	Note	Avis
Spécification	9,3 / 10	Excellente maîtrise des identités, courses, barrières durables, ABA, claims et UX. Elle est exceptionnellement précise. Son principal défaut est sa taille : certaines obligations sont assez dispersées pour qu'une implémentation puisse être fonctionnelle tout en oubliant une exigence de test ou une garde secondaire.
Architecture produite	9,1 / 10	Très bonne séparation entre registre générique, source Prompt, adaptateurs produit, coordinateur de panneau, runtime Activité et frontend. Les handles opaques et les identités structurelles sont de très bons choix.
Robustesse asynchrone	8,8 / 10	Les réservations synchrones, générations destructives, FIFO, wake-ups et guards des queues sont solides. La garde partielle du hook de finalisation reste le point faible principal.
Qualité du typage	9,4 / 10	Schémas Zod stricts, unions discriminées, absence de <code>any</code> dans le cœur ajouté, identité complète transportée proprement et contexte webview

Élément	Note	Avis
Maintenabilité		gardé hors sérialisation.
	8,5 / 10	Les commentaires et nouveaux services sont très bons. Quelques state machines centrales restent volumineuses, notamment le registre, le transport frontend et la page Prompt, ce qui augmente le coût des futures évolutions.
Protection par les tests	8,6 / 10	Le volume et la diversité des tests sont remarquables. Les scénarios principaux et de nombreuses courses sont couverts. Les lacunes concernent surtout la portée exacte de quelques tests promis par la spécification.

Le périmètre réel est correctement devenu plus large que la liste initiale afin de traiter proprement :

- le framing Slimbk dans [Slimbk.ts](#) ;
- la génération du client dans [slimbkBuildHeader.ts](#) ;
- les identités et erreurs retenues des queues ;
- le retrait du pré-verrouillage prématuré dans [usePromptIndexPending.hook.tsx](#) ;
- le harness partagé de fallback dans [harness de fallback persist](#).

Ces extensions ne constituent pas du hors-périmètre nuisible : elles ferment des races que la liste exacte des fichiers avait sous-estimées.

Conclusion de la question 4

Le code est globalement de niveau production et très bien architecturé ; il lui manque surtout une dernière passe de verrouillage par tentative et de complétude des tests pour correspondre réellement à l'ambition de la spécification.

5) Résumé du rapport — TL-DR

Question

Peux-tu résumer le rapport dans un tableau ?

Réponse

Domaine	Résultat	Estimation	Action
Fonction utilisateur fermeture/réouverture	✓	98 %	Aucune refonte nécessaire
Contrats <code>sourceId</code> , replay et descripteurs	✓	100 %	Aucun correctif
Backend registre/source/replay	✓	99 %	Aucun défaut principal identifié
Transport V1 et erreurs Slimbk post-header	✓	100 %	Aucun correctif
Activité, Stop, son et coordination des panneaux	✓	98 %	Aucun défaut principal identifié
Suppression et Retry destructifs	✓	98 %	Aucun défaut principal identifié
Frontend transactionnel, composer et handoff	⚠	96 %	Ajouter la garde exacte de finalisation
Isomorphisme et tests de courses	⚠	89 %	Compléter la matrice promise
Conventions des tests frontend	✗	98 %	Retirer un import direct de <code>expect</code>
Spécification complète	⚠ Presque	96 %	Une passe corrective ciblée suffit

6) Conclusion en une phrase

Question

Quelle est la conclusion en une phrase ?

Réponse

La tranche `liveSession` pour Prompt normal est fonctionnellement livrée et très solide, mais elle ne doit pas encore être déclarée conforme à 100 % tant que la finalisation n'est pas entièrement gardée par l'identité de tentative et que les trois lacunes de tests identifiées ne sont pas corrigées.

7) Lots de corrections proposés

Question

Si des corrections sont nécessaires, quels lots de corrections proposes-tu ?

Lot A — Robustesse runtime prioritaire 🟡

Objectif : fermer la dernière course frontend liée à la finalisation.

- Introduire une identité de tentative attachée aux refs de cible Prompt.

- L'armer uniquement après `start().accepted === true`, dans la même pile que le commit du scaffold.
- Dans [hook de finalisation](#), autoriser les mutations de messages, buffers et refs uniquement si cette identité correspond au `streamInstanceId` finalisé.
- Laisser une finalisation stale publier son propre signal si nécessaire, mais l'empêcher de toucher les cibles du successeur.
- Ajouter une régression « finalisation A après préparation B » dans [tests du contrôleur liveSession](#).

Lot B — Conformité et complétude des tests 🧪

Objectif : respecter exactement les obligations de test et les conventions du dépôt.

- Retirer l'import direct de `expect` dans [test Continue](#).
- Compléter [test d'isomorphisme normal Prompt](#) avec :
 - `prompt`, `retry`, `edit`, `compact context` et `SPECIF` ;
 - outils/notices et regroupements d'édition ;
 - EOF sans terminal ;
 - terminal suivi de `transportError` ;
 - comparaison streaming/replay/hydratation après continuation.
- Renforcer [test du contexte d'origine](#) en passant par [wrapper réel du bus webview](#), avec deux webviews Prompt distinctes.

Lot C — Tous les correctifs 🛠️

Ce lot applique **le Lot A et le Lot B ensemble** :

- garde exacte de toutes les mutations de finalisation ;
- test de course prédécesseur/successeur ;
- matrice d'isomorphisme complète ;
- test d'origine webview end-to-end ;
- suppression de l'import frontend interdit de `expect`.

🔵 Coder directement le lot A

🧪 Coder directement le lot B

🛠️ Coder directement tous les correctifs

</ASSISTANT>

<USER>

[Coder directement tous les correctifs]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧 ✅ reading project: [liveSession.programming-doc-readme.md](#), [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#), [prompt-stream-dispatch-lock.programming-doc-readme.md](#), [usePromptRunController.finalization.hook.ts](#), [usePromptRunController.hook.ts](#), [usePromptRunCommands.hook.ts](#), [usePromptRunController.liveSessionObserver.test.tsx](#), [usePromptRunCommands.continueResponse.test.tsx](#), [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#), [usePromptLiveSession.normalPrompt.test.tsx](#), [usePromptLiveSession.handoff.test.tsx](#), [extMessageBus.streamOriginContext.test.ts](#), [extMessageBus.util.ts](#), [extTypedMessageBus.util.ts](#), [extTypedMessageBus.broadcastMultiPrompt.test.ts](#), [extMessageBus.streamOriginContext.test.ts](#), 🧠 e "Publication outcome stream", 🧠 e "Libération stream actif", 🧠 e "État instance stream", [useAiStream.hook.ts](#) [L560-L880], [useAiStream.hook.ts](#) [L880-L1172], 🧠 e "Stubs onDidReceiveMessage", 🧠 e "Initialisation bus en test", [extTypedMessageBus.broadcastMultiPrompt.test.ts](#), [FrontChatSession.type.ts](#), 🧠 h "Hydratation outils Prompt", 🧠 h "Isomorphisme outils tests", [applyHydratedPromptSessionState.util.test.ts](#), [promptHistoryTypes.type.ts](#), [frontSessionToPromptMessages.gitGroup.hydratation.test.ts](#), [frontSessionToPromptMessages.specif.hydratation.test.ts](#),

[frontSessionToPromptMessages.delete.hydratation.test.ts](#), [frontSessionToPromptMessages.util.ts](#), ["Événement file delete"](#), ["Rendu file delete"](#), ["Type événements notices"](#), [frontSessionHydration.notices.util.ts](#) [L1-L260], [frontSessionHydration.pairing.util.ts](#) [L560-L760], [frontSessionHydration.notices.util.ts](#) [L260-L520], [frontSessionHydration.notices.util.ts](#) [L520-L639], [h "Terminal puis transport test"](#), ["Test terminal transport"](#), [fileWriteGrouping.store.ts](#), ["Hydratation grouping files"](#), ["Streaming grouping files"](#), ["Set grouping usage"](#), ["Select grouping tests"](#),

[frontSessionHydration.fileWriteGrouping.util.ts](#), [frontSessionHydration.fileWriteGrouping.util.test.ts](#), ["Seed grouping hydratation"](#), [frontSessionHydration.type.ts](#) [L340-L470], [frontSessionHydration.type.ts](#) [L470-L529], [AiQuery.type.ts](#) [L1-L260], [r "usePromptRunCommands"](#), [usePromptRunCommands.startAcceptance.test.tsx](#), ["Hydratation file write"](#), ["Mapping file write"](#), [frontSessionHydration.pairing.util.ts](#) [L525-L585]

  patching file: [usePromptRunController.finalization.hook.ts](#)
[AI/Code/src/frontend/src/app/prompt/Prompt/usePromptRunController.finalization.hook.ts](#)

Copy

Review

Accept

PatchFull file

```
-      streamingThinkingUidRef: RefObject<string | undefined>;
-
+      streamingThinkingUidRef: RefObject<string | undefined>;
+      streamingTargetStreamInstanceIdRef: RefObject<string | undefined>;
+
-
-      lastConsumedRawIndexRef,
-      streamingThinkingUidRef,
-
+      lastConsumedRawIndexRef,
+      streamingThinkingUidRef,
+      streamingTargetStreamInstanceIdRef,
+
-
-      useEffect(() => {
-
+      useEffect(() => {
+        /**
+         * Freeze the target identity and mutable Prompt refs for this render before
+         * scheduling any React updater.
+         * A successor stream may synchronously commit its own scaffold after the
+         * transport owner released its
+         * single-flight but before this passive effect runs. Only the exact attempt
+         * that installed the current
+         * target refs is allowed to finalize notices, attach terminal metadata, or
+         * clear those refs.
+         */
+        const finalizedStreamInstanceId = streamInstanceId;
+        const ownsCurrentStreamingTargets =
+          !isStreaming &&
+          finalizedStreamInstanceId !== undefined &&
+          streamingTargetStreamInstanceIdRef.current ===
+            finalizedStreamInstanceId;
+        const finalizedMessageUid = ownsCurrentStreamingTargets
+          ? streamingMessageUidRef.current
+          : undefined;
+        const finalizedTurnId = ownsCurrentStreamingTargets
+          ? currentTurnIdRef.current
+          : undefined;
+
-
-        !isStreaming &&
-        (streamingTokens.length > 0 || streamingThinkingTokens.length > 0)
```

```

-
+         ownsCurrentStreamingTargets &&
+         (streamingTokens.length > 0 || streamingThinkingTokens.length > 0)
+


---


-
-         // Freeze assistant bubble: attach meta
-         const targetUid = streamingMessageUidRef.current;
-
-         if (targetUid) {
-
+         // Freeze the exact assistant bubble owned by this attempt; never consult
+         successor refs in the updater.
+         if (finalizedMessageUid) {
+


---


-                 setMessages((prev) => {
-                     const idx = prev.findIndex((m) => m.uid === targetUid);
-
+                 setMessages((prev) => {
+                     const idx = prev.findIndex(
+                         (message) => message.uid === finalizedMessageUid,
+                     );
+


---


-         // Mark meta as attached for this turn to prevent a subsequent meta-only
pass
-         lastMetaChunkFinalizedRef.current = lastMetaChunk;
-         metaOnlyFinalizedTurnIdRef.current = currentTurnIdRef.current;
-
+         // Mark meta as attached for this turn to prevent a subsequent meta-only
pass
+         lastMetaChunkFinalizedRef.current = lastMetaChunk;
+         metaOnlyFinalizedTurnIdRef.current = finalizedTurnId;
+


---


-
-         // Clear identities
-         streamingMessageUidRef.current = undefined;
-         streamingThinkingUidRef.current = undefined;
-
+         // Clear identities only while this finalized attempt still owns the
+         mutable target refs.
+         if (
+             streamingTargetStreamInstanceIdRef.current ===
+             finalizedStreamInstanceId
+         ) {
+             streamingMessageUidRef.current = undefined;
+             streamingThinkingUidRef.current = undefined;
+         }
+


---


-         if (
-             !isStreaming &&
-
+         if (
+             ownsCurrentStreamingTargets &&
+


---


-         ) {
-             // Guard 1: skip if we already finalized this turn under meta-only branch
-             if (
-
+         ) {

```

```

+         const wasAlreadyFinalizedForTurn =
+
-         metaOnlyFinalizedTurnIdRef.current !== undefined &&
-         metaOnlyFinalizedTurnIdRef.current === currentTurnIdRef.current
-     ) {
-         publishFinalizedStreamInstance();
-         return;
-     }
+
+     metaOnlyFinalizedTurnIdRef.current !== undefined &&
+     metaOnlyFinalizedTurnIdRef.current === finalizedTurnId;
+     const wasExactMetaAlreadyAttached =
+         lastMetaChunkFinalizedRef.current === lastMetaChunk;
+
-
-     // Guard 2: skip if we already attached this exact meta chunk object
-     if (lastMetaChunkFinalizedRef.current === lastMetaChunk) {
-         publishFinalizedStreamInstance();
-         return;
-     }
+
+     if (!wasAlreadyFinalizedForTurn && !wasExactMetaAlreadyAttached) {
+         // Finalize the last active notice at turn end as well (no message ever
arrived).
+         toolCallNotice.finalizeActiveNoticeAtTurnEnd();
+
-
-         // Finalize the last active notice at turn end as well (no message ever
arrived)
-         toolCallNotice.finalizeActiveNoticeAtTurnEnd();
-
+         setMessages((prev) => {
+             const out = [...prev];
+
-
-         setMessages((prev) => {
-             const out = [...prev];
-
+
+             // Transactional starts always install an exact assistant target
before callbacks can finalize.
+             // If that target is absent, do not guess from the last assistant
because it may belong to a
+             // successor attempt reusing the same logical turn.
+             const idx = finalizedMessageUid
+                 ? out.findIndex(
+                     (message) =>
+                         message.uid === finalizedMessageUid,
+                 )
+                 : -1;
+
-
-             // Prefer current streaming placeholder if any
-             const smUid = streamingMessageUidRef.current;
-             let idx = smUid ? out.findIndex((m) => m.uid === smUid) : -1;
-
-             // Otherwise fallback to the last assistant message
-             if (idx < 0) {

```

```

-         for (let i = out.length - 1; i >= 0; i--) {
-             if (out[i]?.role === 'assistant') {
-                 idx = i;
-                 break;
-             }
-         }
+
+         if (idx >= 0) {
+             // Do not rewrite message if the same meta is already there.
+             type MsgWithMeta = PromptMessage & {
+                 lastMetaChunk?: unknown;
+             };
+             const existing = out[idx] as MsgWithMeta;
+             if (existing.lastMetaChunk === lastMetaChunk) {
+                 return prev;
+             }
+         }
-     }
- }
+
+     }
+ }

```

```

-         if (idx >= 0) {
-             // Do not rewrite message if the same meta is already there
-             type MsgWithMeta = PromptMessage & {
-                 lastMetaChunk?: unknown;
-             };
-             const existing = out[idx] as MsgWithMeta;
-             if (existing.lastMetaChunk === lastMetaChunk) {
-                 return prev;
-             }
+
+             const updated = {
+                 ...(out[idx] as object),
+                 lastMetaChunk,
+                 fileEditsUiFinalized: true,
+             } as unknown as PromptMessage;
+             out[idx] = updated;
+             return out;
-         }
-     }
+
+     }
+
+     // No structural addition in meta-only termination.
+     return prev;
+ });

```

```

-         // No structural addition in meta-only termination
-         return prev;
-     });
+
+     // Remember that this turn/meta has been finalized in the meta-only
+     path.

```

```

+         lastMetaChunkFinalizedRef.current = lastMetaChunk;
+         metaOnlyFinalizedTurnIdRef.current = finalizedTurnId;
+
-
-         // Remember that this turn/meta has been finalized in meta-only path
-         lastMetaChunkFinalizedRef.current = lastMetaChunk;
-         metaOnlyFinalizedTurnIdRef.current = currentTurnIdRef.current;
-
+         // Also reset base text captured for Continue.
+         baseAssistantTextRef.current = '';
+     }
+ }
+
-
-         // Also reset base text captured for Continue
-         baseAssistantTextRef.current = '';
-
+
+         // Retire only the exact target ownership finalized by this effect. A successor
identity is never cleared.
+         if (
+             ownsCurrentStreamingTargets &&
+             streamingTargetStreamInstanceIdRef.current ===
+                 finalizedStreamInstanceId
+         ) {
+             streamingTargetStreamInstanceIdRef.current = undefined;
+
-         streamingThinkingUidRef,
-
+         streamingThinkingUidRef,
+         streamingTargetStreamInstanceIdRef,
+

```

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

Copy

Review

Accept

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

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

Error

([what is it?](#))

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

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

Error

([what is it?](#))

  reading project:  [usePromptRunCommands.hook.ts](#) [L1-L390],  [usePromptRunCommands.hook.ts](#) [L390-L580]

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

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

Error

([what is it?](#))

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

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

Error

([what is it?](#))

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

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

Copy

Review

Accept

PatchFull file

```
- } from './preparePromptContinuationStreaming.util.js';
-
+ } from './preparePromptContinuationStreaming.util.js';
+
+ type AcceptedAiStreamStartResult = Extract<
+   AiStreamStartResult,
+   { accepted: true }
+ >;
+
- /**
-  * Stable re-export of the compact-context sentinel.
-
+ /**
+  * Bind mutable Prompt targets to one accepted transport before any scaffold mutation
+  * becomes visible.
+  * The finalization hook uses this exact identity to reject stale predecessor effects.
+  */
+ function armAcceptedStreamingTargetIdentity(
+   streamingTargetStreamInstanceIdRef: React.RefObject<string | undefined>,
+   startResult: AcceptedAiStreamStartResult,
+ ): void {
+   streamingTargetStreamInstanceIdRef.current = startResult.streamInstanceId;
+ }
+
+ /**
+  * Stable re-export of the compact-context sentinel.
+
-   streamingMessageUidRef,
-   streamingThinkingUidRef,
-
+   streamingMessageUidRef,
+   streamingThinkingUidRef,
+   streamingTargetStreamInstanceIdRef,
+
-   streamingThinkingUidRef: React.RefObject<string | undefined>;
-
+   streamingThinkingUidRef: React.RefObject<string | undefined>;
+   streamingTargetStreamInstanceIdRef: React.RefObject<string | undefined>;
+
-   if (!startResult.accepted) return startResult;
-   setUiPending(true);
-
-   // Commit the prebuilt scaffold in the same stack as the accepted start
result.
-
+   if (!startResult.accepted) return startResult;
+   armAcceptedStreamingTargetIdentity(
```



```

+         streamingTargetStreamInstanceIdRef,
+         startResult,
+     );
+     setUiPending(true);
+
+     // Commit the prebuilt scaffold in the same stack as the accepted start
result.
+

```

```

-         setUiPending,
-         start,
-     ],
- );
-
- /**
+         setUiPending,
+         start,
+         streamingTargetStreamInstanceIdRef,
+     ],
+ );
+
+ /**
+

```

```

-         if (!startResult.accepted) return startResult;
-         setUiPending(true);
-
+         if (!startResult.accepted) return startResult;
+         armAcceptedStreamingTargetIdentity(
+             streamingTargetStreamInstanceIdRef,
+             startResult,
+         );
+         setUiPending(true);
+

```

```

-         sessionLocator.storageScopeRelPath,
-         start,
-
+         sessionLocator.storageScopeRelPath,
+         start,
+         streamingTargetStreamInstanceIdRef,
+

```

```

-         aiQueryOptions: aiQueryOptionsBase,
-     });
-     if (!startResult.accepted) return startResult;
-
+         aiQueryOptions: aiQueryOptionsBase,
+     });
+     if (!startResult.accepted) return startResult;
+     armAcceptedStreamingTargetIdentity(
+         streamingTargetStreamInstanceIdRef,
+         startResult,
+     );
+

```

```

-         start,
-         aiQueryOptionsBase,
-
+         start,
+         aiQueryOptionsBase,
+         streamingTargetStreamInstanceIdRef,
+

```

```

-         aiQueryOptions: plannedOptions,

```

```

-     });
-     if (!startResult.accepted) return startResult;
-
+     aiQueryOptions: plannedOptions,
+   });
+   if (!startResult.accepted) return startResult;
+   armAcceptedStreamingTargetIdentity(
+     streamingTargetStreamInstanceIdRef,
+     startResult,
+   );
+
-   setUiPending,
-   start,
-
+   setUiPending,
+   start,
+   streamingTargetStreamInstanceIdRef,
+
-   if (!startResult.accepted) return startResult;
-   // Engage pending latch
-
+   if (!startResult.accepted) return startResult;
+   armAcceptedStreamingTargetIdentity(
+     streamingTargetStreamInstanceIdRef,
+     startResult,
+   );
+   // Engage pending latch
+
-   start,
-   ],
-
+   start,
+   streamingTargetStreamInstanceIdRef,
+   ],
+

```

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

Copy

Review

Accept



🟢 patching file: [usePromptRunController.hook.ts](#)

[AI Code/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts](#)

Copy

Review

Accept

PatchFull file

```

-   const streamingThinkingUidRef = useRef<string | undefined>(undefined);
-
+   const streamingThinkingUidRef = useRef<string | undefined>(undefined);
+   /** Exact accepted stream instance that currently owns the mutable Prompt target
+   refs. */
+   const streamingTargetStreamInstanceIdRef = useRef<string | undefined>(
+     undefined,
+   );
+
-   streamingMessageUidRef,
-   streamingThinkingUidRef,
-
+   streamingMessageUidRef,

```

```

+      streamingThinkingUidRef,
+      streamingTargetStreamInstanceIdRef,
+
-      lastConsumedRowIndexRef,
-      streamingThinkingUidRef,
-
+      lastConsumedRowIndexRef,
+      streamingThinkingUidRef,
+      streamingTargetStreamInstanceIdRef,
+
-      if (!startResult.accepted) return startResult;
-
+      if (!startResult.accepted) return startResult;
+
+      // Arm exact target ownership before either replay mode mutates messages or
streaming refs.
+      streamingTargetStreamInstanceIdRef.current =
+        startResult.streamInstanceId;
+
-      streamingMessageUidRef,
-      streamingThinkingUidRef,
-
+      streamingMessageUidRef,
+      streamingThinkingUidRef,
+      streamingTargetStreamInstanceIdRef,
+
-      streamingThinkingUidRef.current = undefined;
-
+      streamingThinkingUidRef.current = undefined;
+      streamingTargetStreamInstanceIdRef.current = undefined;
+
-      setStreamingTokens,
-      setUiPending,
-
+      setStreamingTokens,
+      setUiPending,
+      streamingTargetStreamInstanceIdRef,
+

```

[AI Code/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts](#)

Copy

Review

Accept

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

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

Error

([what is it?](#))

  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 { beforeEach, describe, expect, it, jest } from '@jest/globals';

```

```

-
+
+ import { beforeEach, describe, it, jest } from '@jest/globals';
+
-   const streamingThinkingUidRef = useRef<string | undefined>(undefined);
-
+   const streamingThinkingUidRef = useRef<string | undefined>(undefined);
+   const streamingTargetStreamInstanceIdRef = useRef<string | undefined>(
+     undefined,
+   );
+
-   streamingThinkingUidRef,
-   streamingMessageUidRef,
-   streamingThinkingUidRef,
-
+   streamingThinkingUidRef,
+   streamingMessageUidRef,
+   streamingThinkingUidRef,
+   streamingTargetStreamInstanceIdRef,
+
-   streamingThinkingUidRef,
-   currentTurnIdRef,
-
+   streamingThinkingUidRef,
+   streamingTargetStreamInstanceIdRef,
+   currentTurnIdRef,
+
-   expect(result.current.streamingThinkingUidRef.current).toBeDefined();
-
+   expect(result.current.streamingThinkingUidRef.current).toBeDefined();
+   expect(
+     result.current.streamingTargetStreamInstanceIdRef.current,
+   ).toBe('stream');
+
-   expect(result.current.streamingThinkingUidRef.current).toBeUndefined();
-
+   expect(result.current.streamingThinkingUidRef.current).toBeUndefined();
+   expect(
+     result.current.streamingTargetStreamInstanceIdRef.current,
+   ).toBeUndefined();
+

```

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

Copy

Review

Accept



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

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

Copy

Review

Accept

PatchFull file

```

-   const [, setStreamingTokens] = useState<string[]>([]);
-
+   const [, setStreamingTokens] = useState<string[]>([]);
+   const streamingTargetStreamInstanceIdRef = useRef<string | undefined>(
+     undefined,
+   );
+

```

```

+
-      streamingThinkingUidRef: useRef<string | undefined>(undefined),
-
+      streamingThinkingUidRef: useRef<string | undefined>(undefined),
+      streamingTargetStreamInstanceIdRef,
+
-    return useMemo(
-      () => ({ ...commands, messages, uiPending }),
-      [commands, messages, uiPending],
-
+    return useMemo(
+      () => ({
+        ...commands,
+        messages,
+        streamingTargetStreamInstanceIdRef,
+        uiPending,
+      }),
+      [commands, messages, streamingTargetStreamInstanceIdRef, uiPending],
+
-    expect(result.current.uiPending).toBe(false);
-
+    expect(result.current.uiPending).toBe(false);
+    expect(
+      result.current.streamingTargetStreamInstanceIdRef.current,
+    ).toBeUndefined();
+
-  );
-  expect(result.current.uiPending).toBe(true);
-
+  );
+  expect(result.current.uiPending).toBe(true);
+  expect(
+    result.current.streamingTargetStreamInstanceIdRef.current,
+  ).toBe('stream');
+
-  });
-  expect(result.current.uiPending).toBe(true);
-
+  });
+  expect(result.current.uiPending).toBe(true);
+  expect(
+    result.current.streamingTargetStreamInstanceIdRef.current,
+  ).toBe('retry-stream');
+

```

[AICode/src/frontend/src/app/prompt/Prompt/ _test_ /usePromptRunCommands.startAcceptance.test.tsx](#)

Copy

Review

Accept



patching file: [usePromptRunController.liveSessionObserver.test.tsx](#)

[AICode/src/frontend/src/app/prompt/Prompt/ _test_ /usePromptRunController.liveSessionObserver.test.tsx](#)

Copy

Review

Accept

PatchFull file

```

- import type { AiStreamStartResult } from
-   '.../.../.../messageBus/AiStreamStartArgs.type.js';
-

```

```

+ import type { AiStreamStartResult } from
+ '../../../../../messageBus/AiStreamStartArgs.type.js';
+ import { AiStopReason } from
+ '../../../../../sharedlib/types/aiTypes/AiStopReason.type.js';
+ import type { AiStreamedChunk } from
+ '../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+
- let mockedStreamInstanceId: string | undefined;
-
+ let mockedStreamInstanceId: string | undefined;
+ let mockedLastMetaChunk: AiStreamedChunk | undefined;
+
-
-     start: startMock,
-     lastMetaChunk: undefined,
-
+     start: startMock,
+     lastMetaChunk: mockedLastMetaChunk,
+
-
-     mockedStreamInstanceId = undefined;
-
+     mockedStreamInstanceId = undefined;
+     mockedLastMetaChunk = undefined;
+
-
-     it('publishes finalization for an accepted stream that completed without visible
chunks or meta', async () => {
-
+
+     it('keeps a stale predecessor meta-only finalization away from the successor
scaffold', async () => {
+         const successorRunRef: LiveSessionRunRef = {
+             locator: RUN_REF.locator,
+             turnId: 'turn-successor',
+             sourceId: 'source-successor',
+         };
+         const predecessorTerminal: AiStreamedChunk = {
+             turnId: RUN_REF.turnId,
+             type: 'meta',
+             value: {
+                 metaType: 'stop_reason',
+                 value: { reason: AiStopReason.END_TURN },
+             },
+         };
+         startMock
+             .mockReturnValueOnce({
+                 accepted: true,
+                 runRef: RUN_REF,
+                 streamInstanceId: 'stream-predecessor',
+             })
+             .mockReturnValueOnce({
+                 accepted: true,
+                 runRef: successorRunRef,
+                 streamInstanceId: 'stream-successor',
+             });
+         await act(async () => {
+             root.render(React.createElement(Harness));
+         });
+         const controller = controllerApiRef.current;
+         if (!controller)
+             throw new Error('Controller harness was not initialized.');
```

```

+      // Commit both accepted scaffolds before React observes the predecessor
transport finalization.
+      act(() => {
+        controller.startLiveObserver({
+          runRef: RUN_REF,
+          viewerLeaseId: 'lease-predecessor',
+          replayMode: 'replaceTurnAssistants',
+          stopPolicy: 'source',
+        });
+      });
+      act(() => {
+        controllerApiRef.current?.startLiveObserver({
+          runRef: successorRunRef,
+          viewerLeaseId: 'lease-successor',
+          replayMode: 'replaceTurnAssistants',
+          stopPolicy: 'source',
+        });
+      });
+
+      // Publish the delayed predecessor terminal after the successor owns all
mutable target refs.
+      mockedStreamInstanceId = 'stream-predecessor';
+      mockedLastMetaChunk = predecessorTerminal;
+      await act(async () => {
+        root.render(React.createElement(Harness));
+      });
+
+      const successorAssistant = controllerApiRef.current?.messages.find(
+        (message) =>
+          message.turnId === successorRunRef.turnId &&
+          message.role === 'assistant' &&
+          message.assistantBubbleKind !== 'think',
+      );
+      expect(successorAssistant).toEqual(
+        expect.objectContaining({
+          fileEditsUiFinalized: false,
+        }),
+      );
+      expect(successorAssistant?.lastMetaChunk).toBeUndefined();
+      expect(controllerApiRef.current?.finalizedStreamInstanceId).toBe(
+        'stream-predecessor',
+      );
+
+      // The stale predecessor must also leave successor ownership intact for its own
exact finalization.
+      const successorTerminal: AiStreamedChunk = {
+        turnId: successorRunRef.turnId,
+        type: 'meta',
+        value: {
+          metaType: 'stop_reason',
+          value: { reason: AiStopReason.END_TURN },
+        },
+      };
+      mockedStreamInstanceId = 'stream-successor';
+      mockedLastMetaChunk = successorTerminal;
+      await act(async () => {
+        root.render(React.createElement(Harness));
+      });
+
+      const finalizedSuccessorAssistant =
+        controllerApiRef.current?.messages.find(
+          (message) =>
+            message.turnId === successorRunRef.turnId &&

```

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

Accept

[AICode/src/extension/src/messageBus/core/ test /extMessageBus.streamOriginContext.test.ts](#)

```
- import { describe, expect, it, jest } from '@jest/globals';
- import type * as vscode from 'vscode';
-
+ import { afterEach, describe, expect, it, jest } from '@jest/globals';
+ import type * as vscode from 'vscode';
+
```

```

import type { AiStreamedChunk } from
'../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
- import { TypedMessageBus } from '../extTypedMessageBus.util.js';
-
+ import type { AiStreamedChunk } from
'../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import { bus, initializeViewWithMessageBus } from '../extMessageBus.util.js';
+ import type { StreamHandlerOriginContext } from '../extTypedMessageBus.util.js';
+
-
- /** Build the minimal exact Webview identity required by the extension-only origin
contract. */
- function createWebviewIdentity(): vscode.Webview {
-
+
+ type ReceiveMessageListener = Parameters<
+   vscode.Webview['onDidReceiveMessage']
+ >[0];
+
+ type WebviewHarness = {
+   readonly webview: vscode.Webview;
+   readonly postMessage: jest.MockedFunction<vscode.Webview['postMessage']>;
+   emitMessage(event: unknown): Promise<void>;
+ };
+
+ const registeredWebviews: vscode.Webview[] = [];

```



```
+ /** Build one exact webview harness that exposes the listener installed by the real
message-bus wrapper. */
+ function createWebviewHarness(): WebviewHarness {
+   let receiveMessage: ReceiveMessageListener | undefined;
+   const postMessage: jest.MockedFunction<vscode.Webview['postMessage']> =
+     jest.fn(async () => true);
+   const onDidReceiveMessage: vscode.Webview['onDidReceiveMessage'] = (
+     listener,
+   ) => {
+     receiveMessage = listener;
+     return { dispose: jest.fn() };
+   };
+   const webview = {
+     onDidReceiveMessage,
+     postMessage,
+   } as unknown as vscode.Webview;
+   registeredWebviews.push(webview);
+
-   return {
-     postMessage: jest.fn(async () => true),
-   } as unknown as vscode.Webview;
-
+   return {
+     webview,
+     postMessage,
+     emitMessage: async (event: unknown): Promise<void> => {
+       if (!receiveMessage) {
+         throw new Error('Webview message listener was not installed.');
+       }
+       await Promise.resolve(receiveMessage(event));
+     },
+   };
+
- describe('stream handler origin context', () => {
-   it('keeps exact webviews distinct without adding origin fields to serialized args',
async () => {
-     const bus = new TypedMessageBus<Record<string, never>>();
-     const firstWebview = createWebviewIdentity();
-     const secondWebview = createWebviewIdentity();
-     const origins: vscode.Webview[] = [];
-
+ describe('stream handler origin context', () => {
+   afterEach(() => {
+     for (const webview of registeredWebviews.splice(0)) {
+       bus.unregisterWebview('prompt', webview);
+     }
+   });
+
+   it('builds distinct exact origins through the real webview wrapper without mutating
serialized args', async () => {
+     const firstHarness = createWebviewHarness();
+     const secondHarness = createWebviewHarness();
+     const origins: StreamHandlerOriginContext[] = [];
+
-     bus.onStreamed('proxylocalapi:aiStream', (args, emit, origin) => {
-       origins.push(origin.webview);
-
+     bus.onStreamed('proxylocalapi:aiStream', (args, emit, origin) => {
+       origins.push(origin);
```

```

-
-     await bus.dispatchStreamedCall(
-         'proxylocalapi:aiStream',
-         STREAM_ARGS,
-         { htmlPage: 'prompt', webview: firstWebview },
-         { onData: () => {} },
-     );
-     await bus.dispatchStreamedCall(
-         'proxylocalapi:aiStream',
-         STREAM_ARGS,
-         { htmlPage: 'prompt', webview: secondWebview },
-         { onData: () => {} },
-     );
-
+
+     // Install the actual wrapper twice so each listener closes over its own
immutable origin.
+     initializeViewWithMessageBus('prompt', firstHarness.webview);
+     initializeViewWithMessageBus('prompt', secondHarness.webview);
+
+     await firstHarness.emitMessage({
+         type: 'stream:proxylocalapi:aiStream',
+         channel: 'channel-first',
+         payload: STREAM_ARGS,
+     });
+     await secondHarness.emitMessage({
+         type: 'stream:proxylocalapi:aiStream',
+         channel: 'channel-second',
+         payload: STREAM_ARGS,
+     });
+
-
-     expect(origins).toEqual([firstWebview, secondWebview]);
-
+
+     expect(origins).toEqual([
+         { htmlPage: 'prompt', webview: firstHarness.webview },
+         { htmlPage: 'prompt', webview: secondHarness.webview },
+     ]);
+
-
-     );
-
+
+     );
+     expect(payloads).not.toEqual(
+         expect.arrayContaining([
+             expect.objectContaining({ htmlPage: expect.anything() }),
+         ]),
+     );
+     expect(firstHarness.postMessage).toHaveBeenCalledWith(
+         expect.objectContaining({
+             channel: 'channel-first',
+             kind: 'data',
+         }),
+     );
+     expect(secondHarness.postMessage).toHaveBeenCalledWith(
+         expect.objectContaining({
+             channel: 'channel-second',
+             kind: 'data',
+         }),
+     );
+
+

```

Copy
Review
Accept



patching file: [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#)

[AICode/src/frontend/src/app/prompt/liveSession/core/ test /usePromptLiveSession.normalPromptIsomorphism.test.tsx](#)

Copy
Review
Accept

PatchFull file

```
-
- import { describe, it } from '@jest/globals';
-
+
+ import { afterEach, beforeEach, describe, it } from '@jest/globals';
+
- import { afterEach, beforeEach, describe, it } from '@jest/globals';
-
-
+ import { afterEach, beforeEach, describe, it } from '@jest/globals';
+
+ import { specifUiLabelFromAction } from
+ '..../../../../sharedlib/lib/specif/specifUiCodes.util.js';
+ import type { AiQuery } from '..../../../../sharedlib/types/aiTypes/AiQuery.type.js';
+
- import type { AiStreamedChunk } from
+ '..../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
-
+ import type { AiStreamedChunk } from
+ '..../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import { AiToolName } from '..../../../../sharedlib/types/aiTypes/AiToolName.type.js';
+
- import type { FrontChatSession } from '..../../../../hydration/FrontChatSession.type.js';
-
+ import type { FrontChatSession } from '..../../../../hydration/FrontChatSession.type.js';
+ import { eventsToNoticeTextParts } from
+ '..../../../../hydration/frontSessionHydration.notices.util.js';
+ import type { ToolUseEvent } from '..../../../../hydration/frontSessionHydration.type.js';
+
- import { applyHydratedPromptSessionState } from
+ '..../../../../Prompt/applyHydratedPromptSessionState.util.js';
-
+ import { applyHydratedPromptSessionState } from
+ '..../../../../Prompt/applyHydratedPromptSessionState.util.js';
+ import { COMPACT_CONTEXT_NOW_USER_PROMPT } from
+ '..../../../../Prompt/CompactContextNowUserPrompt.const.js';
+
- import { preparePromptContinuationStreaming } from
+ '..../../../../Prompt/preparePromptContinuationStreaming.util.js';
-
+ import { preparePromptContinuationStreaming } from
+ '..../../../../Prompt/preparePromptContinuationStreaming.util.js';
+ import {
+   clearFileWriteGroupingStore,
+   selectFileWriteGrouping,
+   setFileWriteGrouping,
+ } from '..../../../../PromptCodeFrame/fileWriteGrouping.store.js';
+
```

```

-      | 'lastMetaChunk'
-
+      | 'lastMetaChunk'
+      | 'specifAction'
+

```

```

-      lastMetaChunk: message.lastMetaChunk,
-
+      lastMetaChunk: message.lastMetaChunk,
+      specifAction: message.specifAction,
+

```

```

-    };
-  }
-
+    };
+  }
+
+ type ReplaceStyleQuery = Exclude<AiQuery, { type: 'continueResponse' }>;
+
+ type ReplaceStyleQueryScenario = {
+   readonly name: string;
+   readonly query: ReplaceStyleQuery;
+ };
+
+ const REPLACE_STYLE_QUERY_SCENARIOS = [
+   {
+     name: 'prompt',
+     query: { type: 'prompt', promptValue: 'Prompt request' },
+   },
+   {
+     name: 'retryQuery',
+     query: { type: 'retryQuery', promptValue: 'Retried request' },
+   },
+   {
+     name: 'editLastPrompt',
+     query: { type: 'editLastPrompt', promptValue: 'Edited request' },
+   },
+   {
+     name: 'compactContextNow',
+     query: {
+       type: 'compactContextNow',
+       promptValue: COMPACT_CONTEXT_NOW_USER_PROMPT,
+     },
+   },
+   {
+     name: 'specifAction',
+     query: {
+       type: 'specifAction',
+       action: 'verify',
+       specifName: 'SPECIF_LIVE_SESSION_TEST',
+       specifMarkdown: '## Specification',
+       strictMode: true,
+       programmingDocEnabled: true,
+       additionalInstructions: 'Keep exact identities.',
+     },
+   },
+ ] satisfies readonly ReplaceStyleQueryScenario[];
+
+ /** Build the exact user bubble persisted by each replace-style Prompt query. */
+ function buildUserBubbleFromQuery(query: ReplaceStyleQuery): Pick<
+   PromptMessage,
+   'text' | 'specifAction'
+ > {

```

```

+   if (query.type === 'specifAction') {
+     return {
+       text: `${specifUiLabelFromAction(query.action)} ${query.specifName}`,
+       specifAction: {
+         action: query.action,
+         specifName: query.specifName,
+         specifMarkdown: query.specifMarkdown,
+         strictMode: query.strictMode,
+         programmingDocEnabled: query.programmingDocEnabled,
+         additionalInstructions: query.additionalInstructions,
+       },
+     };
+   }
+   return { text: query.promptValue };
+ }
+
+ /** Build the final visible streaming projection after an empty thinking placeholder
+ was retired. */
+ function buildCompletedReplaceStream(query: ReplaceStyleQuery): PromptMessage[] {
+   const userBubble = buildUserBubbleFromQuery(query);
+   const plan = createPromptStreamingTurnPlan({
+     turnId: TURN_ID,
+     buildMessages: (targets) => [
+       {
+         turnId: TURN_ID,
+         uid: 'user',
+         role: 'user',
+         text: userBubble.text,
+         ...(userBubble.specifAction
+           ? { specifAction: userBubble.specifAction }
+           : {}),
+       },
+       ...buildPromptStreamingAssistantMessages(targets),
+     ],
+   });
+
+   return plan.messages
+     .filter((message) => message.assistantBubbleKind !== 'think')
+     .map((message): PromptMessage => {
+       if (message.role !== 'assistant') return message;
+       return {
+         ...message,
+         text: 'Answer',
+         lastMetaChunk: END_TURN_META,
+         fileEditsUiFinalized: true,
+       };
+     });
+ }
+
+

```

```

- describe('normal Prompt live-session isomorphism contracts', () => {
-
+ describe('normal Prompt live-session isomorphism contracts', () => {
+   beforeEach(() => {
+     clearFileWriteGroupingStore();
+   });
+
+   afterEach(() => {
+     clearFileWriteGroupingStore();
+   });
+
+   it.each(REPLACE_STYLE_QUERY_SCENARIOS)(

```

```

+         'rebuilds the $name final message projection identically after live replay
hydration',
+         ({ query }) => {
+             const userBubble = buildUserBubbleFromQuery(query);
+             const streamedMessages = buildCompletedReplaceStream(query);
+             const hydratedMessages = hydrateLiveSession(
+                 buildSession([
+                     {
+                         role: 'user',
+                         turnId: TURN_ID,
+                         content: [{ text: userBubble.text }],
+                         ...(userBubble.specifAction
+                             ? { specifAction: userBubble.specifAction }
+                             : {}),
+                     },
+                     {
+                         role: 'assistant',
+                         turnId: TURN_ID,
+                         content: [{ text: 'Answer' }],
+                         stopReason: AiStopReason.END_TURN,
+                     },
+                 ]),
+                 'replaceTurnAssistants',
+             );
+
+             expect(toComparableMessages(hydratedMessages)).toEqual(
+                 toComparableMessages(streamedMessages),
+             );
+         },
+     );
+
+
+
+
+

```

```

-         ).toBe('Preparing the tool call.');
```

```

-     });
-
+         ).toBe('Preparing the tool call.');
```

```

+     });
+
+     it('rebuilds one file-write notice, code frame, and grouping entry exactly like
streaming', () => {
+         const toolUseId = 'write-primary';
+         const path = 'AICode/src/example.ts';
+         const content = 'export const value = 1;';
+         const writeEvent: ToolUseEvent = {
+             kind: 'write',
+             file: {
+                 type: 'file',
+                 path,
+                 basename: 'example.ts',
+             },
+             propose: {
+                 uid: toolUseId,
+                 path,
+                 content,
+                 hasError: false,
+             },
+         };
+
+         const streamedGrouping = {
+             fileWriteUnicityClass: 'primary',
+             fileWritePrimaryToolUseId: toolUseId,
+         } as const;
+         setFileWriteGrouping(toolUseId, streamedGrouping);
+         const streamedGroupingSnapshot = selectFileWriteGrouping(toolUseId);

```

```

+     const streamedMessages: PromptMessage[] = [
+       {
+         turnId: TURN_ID,
+         uid: 'user-write',
+         role: 'user',
+         text: 'Write the file',
+       },
+       {
+         turnId: TURN_ID,
+         uid: 'assistant-write',
+         role: 'assistant',
+         text: `${eventsToNoticeTextParts([writeEvent]).join('')}Done.`,
+         assistantBubbleKind: 'msg',
+         lastMetaChunk: END_TURN_META,
+         fileEditsUiFinalized: true,
+       },
+     ];
+     clearFileWriteGroupingStore();
+
+     const hydratedMessages = hydrateLiveSession(
+       buildSession([
+         {
+           role: 'user',
+           turnId: TURN_ID,
+           content: [{ text: 'Write the file' }],
+         },
+         {
+           role: 'assistant',
+           turnId: TURN_ID,
+           content: [
+             {
+               toolUse: {
+                 kind: 'static',
+                 name: AiToolName.file_write_full,
+                 input: { path, content },
+                 toolUseId,
+               },
+               fileWriteUnicityClass: 'primary',
+               fileWritePrimaryToolUseId: toolUseId,
+             },
+           ],
+         },
+       ]),
+       {
+         role: 'user',
+         turnId: TURN_ID,
+         content: [
+           {
+             toolResult: {
+               toolUseId,
+               status: 'success',
+               content: [{ text: 'success' }],
+             },
+           },
+         ],
+       },
+       {
+         role: 'assistant',
+         turnId: TURN_ID,
+         content: [{ text: 'Done.' }],
+         stopReason: AiStopReason.END_TURN,
+       },
+     ],
+     'replaceTurnAssistants',

```

```

+         );
+
+         expect(toComparableMessages(hydratedMessages)).toEqual(
+             toComparableMessages(streamedMessages),
+         );
+         expect(selectFileWriteGrouping(toolUseId)).toEqual(
+             streamedGroupingSnapshot,
+         );
+     });
+
+
-     );
- });
-
+     );
+
+     it('keeps one durable terminal projection when the transport later reports
failure', () => {
+         const suffix = buildStopSuffix(AiStopErrorReason.SYSTEM_ERROR);
+         const terminalMeta: AiStreamedChunk = {
+             turnId: TURN_ID,
+             type: 'meta',
+             value: {
+                 metaType: 'error',
+                 value: {
+                     reason: AiStopErrorReason.SYSTEM_ERROR,
+                     message: 'source error',
+                     details: 'source details',
+                 },
+             },
+         };
+
+         const terminalText = `Partial answer${suffix}`;
+         const streamedMessages: PromptMessage[] = [
+             {
+                 turnId: TURN_ID,
+                 uid: 'user-terminal',
+                 role: 'user',
+                 text: 'Ask',
+             },
+             {
+                 turnId: TURN_ID,
+                 uid: 'assistant-terminal',
+                 role: 'assistant',
+                 text: terminalText,
+                 assistantBubbleKind: 'msg',
+                 lastMetaChunk: terminalMeta,
+                 fileEditsUiFinalized: true,
+             },
+         ];
+
+         const hydratedMessages = hydrateLiveSession(
+             buildSession([
+                 {
+                     role: 'user',
+                     turnId: TURN_ID,
+                     content: [{ text: 'Ask' }],
+                 },
+                 {
+                     role: 'assistant',
+                     turnId: TURN_ID,
+                     content: [{ text: terminalText }],
+                     errorReason: AiStopErrorReason.SYSTEM_ERROR,
+                     errorMessage: 'source error',
+                     errorDetails: 'source details',
+                 },
+             ]),

```



```

+         ]),
+         'replaceTurnAssistants',
+     );
+
+     // A later transportError changes only the attempt outcome; the durable
terminal projection stays unique.
+     expect(toComparableMessages(hydratedMessages)).toEqual(
+         toComparableMessages(streamedMessages),
+     );
+     expect(
+         hydratedMessages.filter(
+             (message) =>
+                 message.role === 'assistant' &&
+                 message.lastMetaChunk?.type === 'meta' &&
+                 message.lastMetaChunk.value.metaType === 'error',
+         ),
+     ).toHaveLength(1);
+     expect(
+         hydratedMessages.find(
+             (message) => message.role === 'assistant',
+         )?.text.split(suffix),
+     ).toHaveLength(2);
+   });
+ });
+

```

[AI Code/src/frontend/src/app/prompt/liveSession/core/_test_/usePromptLiveSession.normalPromptIsomorphism.test.tsx](#)

Copy

Review

Accept

  reading project:  [usePromptRunController.finalization.hook.ts](#),  [usePromptRunCommands.hook.ts](#) [L1-L620],  [usePromptRunController.hook.ts](#) [L210-L700],  [usePromptRunController.liveSessionObserver.test.tsx](#) [L1-L380],

 [extMessageBus.streamOriginContext.test.ts](#),  [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) [L1-L620]

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

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

Error

([what is it?](#))

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

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

Grouped

([what is it?](#))

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

[AI Code/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts](#)

Grouped

([what is it?](#))

  patching file:  [usePromptRunController.finalization.hook.ts](#)

[AI Code/src/frontend/src/app/prompt/Prompt/usePromptRunController.finalization.hook.ts](#)

Grouped

([what is it?](#))

  patching file:  [usePromptRunController.liveSessionObserver.test.tsx](#)

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

Grouped

([what is it?](#))

-   reading project:  e ["Appels commandes Prompt"](#),  e ["Param identité cible"](#)
-   scanning git:  ["Diff ciblé courant"](#),  ["Contrôle espaces diff"](#)
-   reading

project:

-  [usePromptRunCommands.continueResponse.test.tsx](#),
-  [usePromptRunCommands.startAcceptance.test.tsx](#),
-  [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) [L620-

L703],  [usePromptRunController.finalization.hook.ts](#)

-   patching file:  [usePromptRunController.liveSessionObserver.test.tsx](#)

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

Grouped

([what is it?](#))

-   reading project:  [usePromptRunCommands.hook.ts](#) [L270-

L620],  [usePromptRunController.fileEditsUiFinalization.hook.ts](#),  e ["Seed grouping state"](#),  [specifUiCodes.util.ts](#) [L1-L140]

-   patching file:  [usePromptRunController.liveSessionObserver.test.tsx](#)

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

Grouped

([what is it?](#))

-   patching file:  [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#)

[AICode/src/frontend/src/app/prompt/hydration/isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#)

Copy

Review

Accept

PatchFull file

- Every mounted listener plus every Queue A/Queue B entry carries the exact `sourceId + streamInstanceId`; stale callbacks and post-`await` continuations cannot mutate a successor retry.
- Every mounted listener plus every Queue A/Queue B entry carries the exact `sourceId + streamInstanceId`; stale callbacks and post-`await` continuations cannot mutate a successor retry.
- Every accepted Prompt scaffold binds its mutable assistant/thinking target refs to the exact `streamInstanceId`. Passive finalization may still publish the predecessor's completion barrier, but it cannot attach metadata, finalize notices, clear buffers, or retire refs after a successor owns those targets.

[AICode/src/frontend/src/app/prompt/hydration/isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#)

Copy

Review

Accept

-   patching file:  [extMessageBus.streamOriginContext.test.ts](#)

[AICode/src/extension/src/messageBus/core/ test /extMessageBus.streamOriginContext.test.ts](#)

Grouped

([what is it?](#))

🔧  patching file:  [extMessageBus.streamOriginContext.test.ts](#)
[AICode/src/extension/src/messageBus/core/_test_/extMessageBus.streamOriginContext.test.ts](#)
Grouped
([what is it?](#))

🔧  reading project:  [.prettierrc](#),  e ["Jest each readonly"](#)
🔧  scanning recent files
🔧  reading project:  e ["Schéma grouping persisté"](#),  h ["Schéma session outil grouping"](#),  ["Schéma session grouping"](#)

 **Tous les correctifs ont été appliqués**

Joel, la finalisation Prompt possède maintenant son propre contrôle d'identité — les anciens streams ne peuvent plus venir repeindre le successeur pendant que React a le dos tourné 😊.

Changements réalisés

Domaine

Implémentation

Propriété exacte des cibles Prompt Chaque démarrage accepté associe désormais les références mutables du scaffold au `stream`. Cela couvre Send, SPECIF, Continue, Retry, Edit, Compact context, Reply et les observers liés dans [usePromptRunCommands.hook.ts](#) et [usePromptRunController.hook.ts](#).

Finalisation

protégée contre les prédécesseurs stale [usePromptRunController.finalization.hook.ts](#) fige maintenant l'identité, le `turn` ne peut plus être mis à jour et le `turn` ne peut plus être réinitialisé.

Suppression du

fallback assistant ambigu La finalisation meta-only cible exclusivement l'UID installé transactionnellement par la tentative de `startAcceptance`.

Course

prédécesseur → successeur Le test [usePromptRunController.liveSessionObserver.test.tsx](#) simule une finalisation de `liveSessionObserver` par `B`.

Contrat

transactionnel des commandes Les tests [usePromptRunCommands.startAcceptance.test.tsx](#) et [usePromptRunCommands.startAcceptance.test.tsx](#) désormais l'identité propriétaire.

Convention Jest frontend

L'import direct de `expect` depuis `@jest/globals` a été retiré de [usePromptRunCommands.hook.ts](#).

Matrice

d'isomorphisme étendue [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) couvre maintenant les cas de `prompt`, `retryQuery`, `editLastPrompt`, `compactContextNow`, `specifAction` et `compactContextNow`.

Outils, CodeFrames et regroupement d'édérations

La matrice reconstruit un `file_write_full`, sa notice, son frame `PROPOSE_FILE` et son `editLastPrompt`.

Domaine
EOF, throw et
terminal suivi
d'une erreur
transport

Implémentation

Les terminaux SYSTEM_ERROR/INTERRUPTED ainsi que le cas terminal durable puis tra

Origine webview
end-to-end

[extMessageBus.streamOriginContext.test.ts](#) passe désormais par le véritable wrap

Documentation
de maintenance

L'invariant de propriété des cibles par `streamInstanceId` est ajouté à [documentation d'iso](#)

Garanties conservées

- **Aucun fichier `*.generated.*` modifié.**
- **Aucun changement des algorithmes Queue A, Queue B, des locks ou des sanitizers.**
- **Aucun `any`, `eslint-disable`, TODO ou fallback de compatibilité ajouté.**
- **Aucun import dynamique de production ajouté.**
- **Tous les nouveaux commentaires et identifiants sont en anglais.**
- **Le terminal d'une tentative stale peut encore publier sa barrière propre sans altérer les cibles de la tentative courante.**

Nom de commit

text

Copy

fix(live-session): guard prompt finalization and complete replay coverage

🎵 Chaque stream garde maintenant son identité 🛠️ Les vieux callbacks ne voleront plus le scaffold d'à côté 🎵.

</ASSISTANT>

<USER>

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧  reading project:  [usePromptRunController.finalization.hook.ts](#) [L205-L245],  [usePromptRunController.finalization.hook.ts](#) [L245-L249]

🔧  patching file:  [usePromptRunController.finalization.hook.ts](#)
[AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.finalization.hook.ts](#)

Copy

Review

Accept

PatchFull file

```
-      isStreaming,  
-  
+      isStreaming,  
+      streamInstanceId,  
+
```

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

Copy

Review

Accept

  reading project:  [usePromptRunController.finalization.hook.ts](#) [L225-L249]

 Avertissement React corrigé

Bien vu, Joel — le linter avait raison : l'effet lit directement `streamInstanceId` pour figer l'identité de la tentative finalisée, mais cette valeur manquait dans son tableau de dépendances.

Fichier	Correction	Effet
usePromptRunController.finalization.hook.ts	Ajout de <code>streamInstanceId</code> aux dépendances du <code>useEffect</code>	L'effet est réévalué lors l'identité de stream change sans closure obsolète avertissement react hooks/exhaustive

La correction ne modifie **aucun comportement métier** : elle aligne simplement les dépendances React avec les valeurs réellement lues par l'effet.

Nom de commit

text

Copy

`fix(live-session): track stream identity in finalization effect`

 La dépendance est bien déclarée  Et la closure ne peut plus nous piéger .

</ASSISTANT>