

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=98d908292747410db7db7deb5f0fe035-VS-V%C3%A9rifier-finalise

<USER>

[Verify]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_204731

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧 ☒ scanning git: 📁 "État de l'implémentation", 📁 "Inventaire des écarts", 📁 "Volume des changements",
📁 "Fichiers touchés"

🔧 ☒ reading project: 📁 g "Docs live et Prompt"

🔧 ☒ scanning git: 📁 "Historique récent", 📁 "Contenu du HEAD"

🔧 ☒ 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: 📁 "Amplitude du commit", 📁 "Validité du diff", 📁 "Ajouts any potentiels", ❌ 📁 "Contournements potentiels", 📁 "Ancien canAbortSource", 📁 "Ancien streamKey", 📁 "Ancien store windows", 📁 "Anciennes fenêtres acti"

🔧 ☒ reading

project:

📄 [LiveSessionRun.type.ts](#),
📄 [LiveSessionPanel.type.ts](#),
📄 [LiveSessionOpenResult.type.ts](#),
📄 [LiveSessionReplay.type.ts](#),
📄 [SidebarStore.ts](#),
📄 [SidebarUiMessages.ts](#), 📄 [MessageRegistry.ts](#), 📄 [parseSlimbkStreamBodyError.util.ts](#), 📄 [aiStopSuffix.util.ts](#) [L1-L320], 📁 h "Registre live exact", 📁 h "Source Prompt durable", 📁 h "Coordination activité", 📁 h "Identités frontend", 📁 "Mécanique du registre", 📁 "Source et coordination", 📁 "Flux frontend exact", 📁 e "Refus Stop frontend", 📁 h "Tests refus Stop source", 📁 h "Stop replay normal", 📁 e "Règle refus documentée", 📁 "Test Stop transport", 📁 "Stop pré-replay normal", 📄 [LiveSessionRegistry.service.ts](#) [L1-L220], 📄 [LiveSessionRegistry.service.ts](#) [L300-L430], 📄 [LiveSessionRegistry.service.ts](#) [L430-L760], 📄 [LiveSessionRecord.type.ts](#) [L100-L280],

📄 [LiveSessionOpen.service.ts](#),
📄 [LiveSessionSourceLifecycle.service.ts](#),
📄 [stream.endpoint.ts](#),
📄 [iaStreamHandlers.ts](#),
📄 [liveSessionStreamHandlers.ts](#),
📄 [extTypedMessageBus.util.ts](#),
📄 [extMessageBus.util.ts](#),
📄 [Slimbk.ts](#),
📄 [SlimbkResult.type.ts](#),
📄 [usePromptRunCommands.hook.ts](#),
📄 [preparePromptStreamingTurn.util.ts](#),
📄 [preparePromptContinuationStreaming.util.ts](#),
📄 [usePromptThinkingLifecycle.hook.ts](#),
📄 [usePromptRunController.finalization.hook.ts](#), 📄 [AiStreamStartArgs.type.ts](#), 📁 e "Armer barrière fallback", 📁 h "Handlers acceptation Prompt", 📁 h "Bootstrap frontière Prompt", 📁 "Barrière transactionnelle",

📄 [applyHydratedPromptSessionState.util.ts](#),
📄 [NormalPromptLiveSession.adapter.ts](#), 📄 [PromptLiveSession.types.ts](#), 📄 [Prompt.tsx](#) [L1-L220], 📄 [PromptLiveSessionBootstrap.adapter.ts](#) [L100-L180],

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

[PromptLiveSessionDetachedStop.service.ts](#),
[SidebarStore.handlers.manager.ts](#), [NormalPromptLiveSessionSource.service.ts](#) [L1-L120],
[SidebarQueryRunningBridge.ts](#),
[SidebarStore.queryWindow.listeners.manager.ts](#),
[SidebarStore.queryWindow.mutations.manager.ts](#),
[SidebarActivityQuery.tsx](#),
[delete.endpoint.ts](#), [EasyAgentsBatchRunRetry.service.ts](#), [PromptThreadCleanup.manager.ts](#)
  scanning git:  ["Écart Queue A"](#),  ["Écart Queue B"](#),  ["Écart verrou Prompt"](#),  ["Écart sanitizers"](#)
  reading project: [useAiStream.hook.ts](#) [L1-L260], [useAiStream.hook.ts](#) [L260-L620], [useAiStream.hook.ts](#) [L620-L840], [useAiStream.hook.ts](#) [L1080-L1185], [usePromptCancelController.hook.ts](#), [Prompt.tsx](#) [L620-L745],
[usePromptRunController.hook.ts](#), [slimbkBuildHeader.ts](#), [callLocalApi.ai.orch.query.ask.generated.ts](#) [L1-L220], [callLocalApi.generated.ts](#) [L1-L180], [callLocalApi.generated.ts](#) [L180-L420], [PromptLiveSessionPanelCoordinator.service.ts](#) [L330-L520], [PromptQueryDoneSound.manager.ts](#),  e ["Usages openMode"](#),  e ["Usages param openMode"](#),  e ["Usages easyAgents line"](#), [usePromptSessionLocator.hook.ts](#)
  scanning git:  ["Ajouts eslint-disable"](#),  ["Ajouts eslint-ignore"](#),  ["Ajouts TODO"](#),  ["Imports dynamiques code"](#),  ["Require dans le code"](#)
  reading project: [OrchMessagesMemoryAssistantPatch.ts](#), [SidebarStore.rehydrate.manager.ts](#) [L1-L217], [SidebarStore.rehydrate.manager.ts](#) [L336-L700],  r ["openCentralPanel"](#)
  scanning git:  ["Historique scroll command"](#)
  reading project: [usePromptLiveSession.hook.ts](#),  e ["Usage unavailable live"](#)
  scanning git:  ["Ancienne gestion erreurs"](#)
  reading project: [EasyAgentsBatchLiveSession.adapter.ts](#),  e ["Titre watcher indispo"](#),  e ["Titre bootstrap échoué"](#),  h ["Tests dialogue EasyAgents"](#),  e ["Test popup live rejet"](#),  e ["Test fermeture live"](#),  e ["Mock alerte live"](#), [VscButton.tsx](#),  h ["Test Stop claim destructif"](#),  ["Couverture Stop détaché"](#),  e ["Destruction dans test Stop"](#), [PromptInput.tsx](#) [L1-L280],  h ["Composer et bandeau"](#),  ["Rendu Prompt live"](#),  h ["Verrou attachments live"](#), ["Interactions attachments"](#), e ["Enregistrement thread Prompt"](#), [EasyAgentsBatchLiveSessionAdapter.service.ts](#), e ["Promise then live backend"](#), e ["Promise then live front"](#), e ["Promise then shared"](#), [usePromptLiveSession.normalPrompt.test.tsx](#) [L1-L35], [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) [L1-L35], [useAiStream.hook.promptSourceIdentity.test.ts](#) [L1-L35], [usePromptRunCommands.startAcceptance.test.tsx](#) [L1-L35], [SidebarActivityQuery.test.tsx](#) [L1-L25], [frontSessionHydration.type.ts](#),  e ["Refus Stop test source"](#),  e ["Rejet popup EasyAgents"](#),  e ["CloseSelf testé live"](#),  ["Résultat Stop refusé"](#), [useStreamListener.hook.ts](#),  h ["Focus après fallback"](#),  ["Effet de focus"](#), [usePromptIndexPending.hook.tsx](#),  e ["Usage onBeforeSubmitUi"](#), [usePromptLiveSession.hydration.test.tsx](#) [L1-L320], [usePromptLiveSession.handoff.test.tsx](#) [L1-L320],  e ["Hydratation classique gating"](#),  e ["Garde hydratation live"](#)

Verdict de vérification

Très beau chantier, Joel : **23 409 insertions sur 189 fichiers** — `liveSession` a maintenant davantage de garde-fous qu'un coffre-fort bancaire 😊.

J'ai vérifié l'état courant au commit **4198db6a**, dont le nom correspond exactement à celui demandé :

`feat(live-session): keep prompt runs alive across panel close and reopen`

Le dépôt est propre. J'ai également lu les documents de maintenance pertinents : le [document d'architecture liveSession](#), le [document EasyAgents](#), le [contrat d'isomorphisme](#), les [règles du thinking](#), le [contrat Queue B](#) et le [contrat fallback persist](#).

Contrôle structurel

Chemins numérotés par la spécification présents dans le commit

Anciennes

Contrôle structurel

APIs `streamKey`, `canAbortSource`, `queryState.windows` et `SidebarActivityQueryWindowItem` en
présentes

Ajout de `any`, `TODO`, `eslint-disable` ou `eslint-ignore` dans le commit

Erreur de structure ou d'espaces détectée dans le diff

Correspondance du nom de commit

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

Réponse

✗ Pas entièrement au sens strict, mais la tranche principale est presque complète.

- **Couverture structurelle : 100 %** — les 160 chemins demandés ont bien été créés ou modifiés.
- **Couverture fonctionnelle principale estimée : 98 %.**
- **Conformité stricte globale estimée : 97 %.**
- **Écarts restants :**
 - un défaut fonctionnel confirmé dans la gestion d'échec initiale du viewer EasyAgents ;
 - une incohérence entre le contrat documenté et le comportement testé du Stop source ;
 - quelques violations explicites de la convention `async/await` ;
 - des scénarios de régression demandés mais pas couverts exactement.

Conclusion de la question 1

La fonctionnalité essentielle est bien livrée et l'architecture prévue existe, mais je ne classerais pas encore la spécification comme "100 % vérifiée et conforme".

2) L'implémentation actuelle fait-elle ce qu'elle doit faire selon la spécification ? Pour chaque fonctionnalité technique demandée, est-elle complète ?

Matrice de conformité technique

Domaine demandé	État	Conformité estimée	
Identité exacte { <code>locator</code> , <code>turnId</code> , <code>sourceId</code> }	✓ Complet	100 %	Le run key inclut bien <code>sourceId</code>
Descripteurs stricts EasyAgents/normalPrompt	✓ Complet	100 %	Union stricte, règles <code>sourceId=</code> dans LiveSessionPanel.type
Contrat Open live/terminal et enveloppe opaque	✓ Complet	99 %	Les branches, <code>sourceId</code> , repla dans LiveSessionOpenResult
Réservations synchrones, époques et refus des doublons	✓ Complet	99 %	Réservation avant <code>await</code> , pointer dans LiveSessionRegistry.s
Retraites, tombstones et cinq résultats de finalisation	✓ Complet	99 %	<code>reservationCancelled</code> , han avec priorité à la baseline manqua
Claims destructifs et génération indépendante	✓ Complet	99 %	Acquisition/libération, invalidatio présents.
Buffer raw-live ordonné, complet et sans limite ajoutée	✓ Complet	100 %	Le registre conserve le tableau co
Lifecycle partagé du <code>Readable</code> lazy	✓ Complet	99 %	Priming, première lecture, cleanu dans LiveSessionSourceLife
Source Prompt et persistance des <code>meta.error</code> avant <code>append</code>	✓ Complet	99 %	La séquence persist → <code>append</code> re dans NormalPromptLiveSessi
Synthèse SYSTEM_ERROR/INTERRUPTED sans doublon terminal	✓ Complet	99 %	Le terminal Orch observé reste au
Protocole V1 du premier <code>meta.error</code>	✓ Complet	99 %	Encodage/décodage strict, statut,
Erreur Slimbk post-header dans le body	✓ Complet	99 %	Détection avant traitement métier handlers extension.

Domaine demandé	État	Conformité estimée	
Origine exacte de la webview	✓ Complet	100 %	Le contexte extension-only contient dans extTypedMessageBus.ts
Ancien panneau remplacé interdit de démarrer une source	✓ Complet	100 %	Comparaison de la webview d'origine dans iaStreamHandlers.ts
FIFO extension, finalisation, son et Activity	✓ Complet	98 %	Chunks, erreurs auxiliaires, finalisation
Pas de second timer/son pendant le replay source	✓ Complet	100 %	liveSessionStreamHandler
Son exact par run et attempt	✓ Complet	99 %	Run key complet, <code>soundAttempt</code> dans PromptQueryDoneSound
Coordinateur panneau par locator	✓ Complet	98 %	Files sérialisées, priorité explicite dans PromptLiveSessionPanel
Fermeture sans arrêt de la source Prompt	✓ Complet	99 %	Le dispose conserve la source, dé
Réouverture par l'Activité sur la source exacte	✓ Complet	99 %	Le RPC Activité ouvre la locator
Activité Idle/open, observer/open, source/open closed	✓ Complet	100 %	Union stricte dans SidebarStore
Idle si panneau ouvert, suppression si panneau fermé	✓ Complet	99 %	Transitions exactes dans Sidebar
Stop détaché uniquement pour source/closed	✓ Fonctionnellement complet	98 %	Filtrage strict, état local immédiat
Stop normal unique dans Prompt	✓ Complet	100 %	L'état local est plus précis que di
Delete sous claim et attente de settlement	✓ Complet	99 %	La bannière normalPrompt n'affi
Retry EasyAgents atomique sous claim	✓ Complet	99 %	Abort exact, attente de la latch av
Dispatcher brut classic/valid/invalid	✓ Complet	100 %	dans delete.endpoint.ts
Frontière React externe pour query invalide	✓ Complet	100 %	Préchecks, claim adjacent, close, en <code>finally</code> dans EasyAgentsB
Hydratation classique et auto-run neutralisés	✓ Complet	100 %	<code>getAll</code> , doublons, inconnus, loc
Start synchronique et mutation nulle sur refus	✓ Complet	99 %	Aucun contrôleur Prompt n'est m
Plans purs puis commit pour les sept commandes	✓ Complet	99 %	Toute query live explicite désacti
Queue A/Queue B gardées par tentative exacte	✓ Complet	98 %	Single-flight ref, génération des i
Finalisation après Queue A, Queue B et commit	✓ Complet	99 %	Send, SPECIF, Retry, Edit, Comp
Terminal puis transportError sans second terminal	✓ Complet	100 %	dans usePromptRunCommands
Replay <code>replaceTurnAssistants/continueAssistant</code>	✓ Complet	99 %	Le coalescing et le gating restent
Handoff normalPrompt direct sans second Open	✓ Complet	100 %	des erreurs.
Retry <code>preserveLocal</code> réellement opaque	✓ Complet	100 %	L'outcome et la libération du sing
Composer armé après baseline + observer accepté	✓ Complet	99 %	Le terminal visible reste unique ;

Domaine demandé	État	Conformité estimée	
Fallback ack exact et démasquage sans clear	✓ Complet	100 %	Barrière { turnId, streamIn
Continuation et thinking par <code>streamInstanceId</code>	✓ Complet	99 %	Plan pur, nouvel UID thinking et dans preparePromptContinuation
Préservation EasyAgents	△ Presque complète	96 %	<code>sourceId=turnId</code> , readiness h
Documentation demandée	✓ Complète	100 %	d'échec initial du viewer présente
Couverture de tests demandée	△ Très large, pas totalement exacte	96 %	Les six documents de maintenanc
Conventions qualité strictes	△ Presque complètes	95 %	Les grandes courses sont couvertes documentaire.
			Typage strict, commentaires, absence de <code>.then(...)</code> ne respectent ce

Conclusion de la question 2

Le chemin nominal et presque toutes les courses critiques sont implémentés conformément à la spécification ; les écarts sont concentrés dans des branches d'échec, un contrat Stop contradictoire et quelques règles de qualité/test.

3) Existe-t-il des bugs ?

Bug confirmé 1 — la fermeture EasyAgents n'est pas garantie si l'affichage du dialogue échoue

Dans [usePromptLiveSession.hook.ts](#), le chemin d'échec des adaptateurs `terminalSnapshot` exécute le dialogue puis la fermeture séquentiellement :

[AICode/src/frontend/src/app/prompt/liveSession/core/usePromptLiveSession.hook.ts](#)

Copy

```
await showVscAlertDialog({
  icon: 'error',
  title: unavailable
    ? current.bootstrap.presentationAdapter.unavailableDialogTitle
    : current.bootstrap.presentationAdapter.failureDialogTitle,
  labelMessage: message,
});
await closeSelf();
```

Si `showVscAlertDialog(...)` rejette :

- `closeSelf()` n'est jamais appelé ;
- le panneau EasyAgents peut rester ouvert dans un état de bootstrap échoué ;
- la Promise détachée peut rejeter sans récupération ;
- le comportement historique, qui tentait indépendamment le dialogue puis la fermeture, n'est plus préservé.

En outre, le paramètre `unavailable` n'est jamais passé à `true`. Les réponses `not_ready` et `not_found` EasyAgents utilisent donc le titre d'échec générique au lieu du titre d'indisponibilité prévu par l'adaptateur.

Sévérité : moyenne △ — branche d'erreur, mais comportement utilisateur concret et violation de la robustesse historique.

Bug contractuel 2 — le Stop source après refus backend contredit le document d'architecture

Le [document d'architecture liveSession](#) stipule qu'un Stop source normalPrompt doit être définitivement retiré après :

- un claim accepté ;
 - **ou un refus backend structuré ;**
et que seule une erreur de transport doit permettre une nouvelle tentative.
- Or :
- [useAiStream.hook.ts](#) transforme `accepted:false` en exception puis restaure `transportStopAvailable` ;
 - [useAiStream.hook.promptSourceIdentity.test.ts](#) exige explicitement que le Stop soit restauré après **refus ou erreur RPC**.

Le code et son test sont donc cohérents entre eux, mais **contradictoires avec le contrat de maintenance livré dans le même commit**.

Conséquence possible : après que le backend a affirmé que la source n'est plus abortable, le bouton Stop peut réapparaître et produire plusieurs appels refusés jusqu'à la prochaine transition terminale.

Sévérité : faible à moyenne , mais correction nécessaire pour rendre le contrat non ambigu.

Écart qualité confirmé — nouveaux usages de `.then(...)`

La règle projet impose `async/await` dès qu'il est possible de l'utiliser. De nouveaux usages de `.then(...)` existent notamment dans :

- [iaStreamHandlers.ts](#) ;
- [liveSessionStreamHandlers.ts](#) ;
- [PromptLiveSessionPanelCoordinator.service.ts](#).

Ils ne sont pas nécessaires : des fonctions `async` immédiatement invoquées ou un petit helper de normalisation peuvent conserver exactement le même ordre.

Sévérité runtime : aucune identifiée. Conformité qualité : .

Lacunes de tests précises

1. [PromptLiveSessionDetachedStop.service.test.ts](#) couvre le claim d'abort indisponible, mais **pas explicitement le scénario de claim destructif** demandé par la spécification.
2. Les tests de `usePromptLiveSession` ne couvrent pas :
 - le rejet du dialogue EasyAgents ;
 - la vérification que la fermeture est tout de même tentée ;
 - le choix du titre `unavailableDialogTitle` pour `not_ready/not_found`.
3. Le test du Stop source valide actuellement le comportement opposé au document d'architecture.

Conclusion de la question 3

Oui : un bug fonctionnel robuste est confirmé, une incohérence de contrat Stop est confirmée, et quelques lacunes de qualité/test empêchent une validation à 100 %. Je n'ai pas identifié de défaut structurel dans les réservations, époques, claims, queues, replay ou finalisation durable examinés.

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

Sur la spécification

Excellente — 9,5/10 

Ses grandes qualités sont :

- des identités séparées et explicites ;
- une description précise des courses asynchrones ;
- une séparation nette entre époque de run et génération destructive ;
- des règles durables pour le terminal, le replay et les erreurs ;
- des critères de tests exceptionnellement détaillés ;
- une estimation de risque de **92 %** tout à fait justifiée.

Son principal défaut est sa taille : elle nécessite un suivi presque mécanique pour éviter qu'un petit contrat — comme le comportement d'un Stop refusé — diverge entre le code, les tests et la documentation.

Sur le code

Très bon — 9/10 

Points particulièrement réussis :

- `sourceId`, `streamInstanceId` et `replayOpenAttemptId` ont des responsabilités distinctes ;
- les réservations et claims sont synchrones aux frontières critiques ;
- le registre ne ressuscite pas les runs ;
- le terminal durable précède le replay ;
- la pipeline source sépare correctement persistance, append, forwarding et finalisation ;
- les panneaux, l'Activité, le son et le nettoyage sont isolés dans des services spécialisés ;
- `preserveLocal` évite réellement les lectures interdites ;
- le démarrage transactionnel couvre les différentes commandes Prompt ;
- les tests ciblent de nombreuses courses difficiles.

Points à surveiller :

- le volume de coordination reste très élevé ;
- `usePromptLiveSession` concentre plusieurs petits automates — Open, Retry, Stop, replay et handoff ;
- les deux politiques Stop sont proches mais déjà légèrement divergentes ;
- les erreurs UI EasyAgents ont perdu la robustesse indépendante de l'ancien helper ;
- les trois `.then(...)` sont de petits écarts, mais dans un projet aussi strict ils ne doivent pas s'accumuler.

Conclusion de la question 4

L'architecture est professionnelle, défensive et très bien documentée ; il faut corriger les écarts ciblés plutôt que lancer un nouveau refactoring transversal risqué.

5) Résumé du rapport

Sujet	Verdict	Estimation	Action
Présence des 160 chemins spécifiés	✓ Complet	100 %	Aucune
Architecture liveSession principale	✓ Complet	99 %	Aucune refonte
Fermeture et réouverture Prompt normal	✓ Complet	99 %	Aucune
Registre, réservations, époques et claims	✓ Complet	99 %	Aucune
Buffer/replay/handoff	✓ Complet	99 %	Aucune
Persistance des erreurs et protocoles Slimbk	✓ Complet	99 %	Aucune
Activité, Stop détaché et son Démarrage transactionnel frontend	✓ Presque complet	98 %	Compléter un scénario de test
	✓ Complet	99 %	Aucune
Composer, fallback ack et thinking	✓ Complet	99 %	Aucune
Préservation EasyAgents	⚠ Défaut d'erreur initiale	96 %	Corriger la fermeture et les titres
Contrat Stop après refus backend	⚠ Contradictoire	95 %	Aligner code, test et documentation
Respect strict <code>async/await</code>	✗ Incomplet	94 %	Remplacer trois <code>.then(...)</code>
Couverture de tests	⚠ Très forte mais incomplète	96 %	Ajouter les régressions manquantes
Conformité globale	⚠ Presque complète	97 %	Correctifs ciblés avant validation finale

6) Conclusion en une phrase

La tranche est presque entièrement livrée et architecturalement solide, mais elle ne doit pas encore être considérée comme totalement conforme tant que la fermeture EasyAgents, le contrat du Stop source et les écarts qualité/test identifiés ne sont pas corrigés.

TL-DR

- ✓ 160/160 chemins demandés sont implémentés.
- ✓ Le cœur fermeture → Activité → réouverture → replay → handoff fonctionne conformément à l'architecture.
- ⚠ La conformité globale est estimée à 97 %.
- ✗ Une branche d'échec EasyAgents peut ne pas fermer le panneau.
- ⚠ Le comportement du Stop source après `accepted:false` contredit la documentation.
- ⚠ Trois usages de `.then(...)` et quelques scénarios de test précis restent à corriger.

7) Si des corrections sont nécessaires, quels ensembles de corrections proposer ?

Ensemble A — Correctifs fonctionnels

- Restaurer une gestion EasyAgents où le dialogue et `closeSelf()` sont tentés indépendamment.
- Utiliser le titre d'indisponibilité pour `not_ready` et `not_found`.
- Aligner le Stop source actif sur le contrat one-shot documenté :
- refus structuré → disponibilité définitivement retirée ;
- erreur de transport → disponibilité restaurée.
- Adapter les tests correspondants.

Ensemble B — Conformité qualité et couverture

- Remplacer les nouveaux `.then(...)` par des séquences `async/await`.
- Ajouter les tests :
- rejet du dialogue EasyAgents avec fermeture toujours tentée ;
- titres indisponibilité/échec ;
- claim destructif pendant un Stop détaché ;
- distinction refus backend / erreur de transport du Stop source.

Ensemble C — Toutes les corrections

- Appliquer intégralement les ensembles A et B.
- Ne modifier ni l'architecture du registre, ni les algorithmes Queue A/Queue B, ni les sanitizers.
- Conserver les identités, ordres de finalisation et contrats EasyAgents existants.



Coder les correctifs fonctionnels



Coder la conformité qualité et tests



Coder toutes les corrections

</ASSISTANT>

<USER>

[Coder toutes les corrections]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

reading project: e "[Échec bootstrap live](#)", e "[Stop source refusé](#)", e "[Promesses handler normal](#)", e "[Promesses handler live](#)", e "[Promesses coordinateur](#)", [liveSession.programming-doc-readme.md](#), [prompt-stream-dispatch-lock.programming-doc-readme.md](#), [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#), [usePromptLiveSession.hook.ts](#) [L150-L360], [usePromptLiveSession.hydratation.test.tsx](#), [usePromptLiveSession.normalPrompt.test.tsx](#) [L1-L180], [useAiStream.hook.ts](#) [L960-L1035], [useAiStream.hook.promptSourceIdentity.test.ts](#) [L1-L120], [useAiStream.hook.promptSourceIdentity.test.ts](#) [L360-L436], [iaStreamHandlers.ts](#) [L100-L175], [liveSessionStreamHandlers.ts](#) [L50-L115], [PromptLiveSessionPanelCoordinator.service.ts](#) [L390-L445], [PromptLiveSessionDetachedStop.service.ts](#), [PromptLiveSessionDetachedStop.service.test.ts](#), [useAiStream.hook.promptSourceIdentity.test.ts](#) [L120-L230], e "[Stop avant replay](#)", h "[Stop source préreplay](#)", [usePromptLiveSession.hook.ts](#) [L90-L150], [usePromptLiveSession.hook.ts](#) [L460-L550], [LiveSessionService.ts](#) [L1-L130], e "[Claim destructif registre](#)", h "[API claim destructif](#)", [LiveSessionRegistry.service.ts](#) [L850-L985], [LiveSessionRegistry.service.ts](#) [L40-L100], [LiveSessionRegistry.service.ts](#) [L100-L180], e "[Méthode claim source](#)", [LiveSessionRegistry.service.ts](#) [L750-L815], r "[PromptLiveSessionDetachedStopService](#)", e "[Binder EasyAgents](#)", [EasyAgentsBatchLiveSession.adapter.ts](#) [L55-L120], [usePromptLiveSession.hook.ts](#) [L1-L195], [LiveSessionPanel.type.ts](#) [L1-L180], h "[Stop viewer test existant](#)", [useAiStream.hook.liveSessionObserver.test.ts](#) [L180-L275], [useAiStream.hook.liveSessionObserver.test.ts](#) [L130-L330]

patching file: [usePromptLiveSession.hook.ts](#)
[AICode/src/frontend/src/app/prompt/liveSession/core/usePromptLiveSession.hook.ts](#)

Copy

Review

Accept

PatchFull file

```
-         }
-         await showVscAlertDialog({
-             icon: 'error',
-             title: unavailable
-             ? current.bootstrap.presentationAdapter
```

```

-         .unavailableDialogTitle
-         : current.bootstrap.presentationAdapter.failureDialogTitle,
-         labelMessage: message,
-     });
-     await closeSelf();
-
+     }
+     // The historical viewer shutdown has two independent obligations: report
the failure and
+     // close the read-only panel. A popup failure must never strand the failed
viewer onscreen.
+     try {
+         await showVscAlertDialog({
+             icon: 'error',
+             title: unavailable
+                 ? current.bootstrap.presentationAdapter
+                   .unavailableDialogTitle
+                 : current.bootstrap.presentationAdapter
+                   .failureDialogTitle,
+             labelMessage: message,
+         });
+     } catch (dialogError) {
+         console.error(
+             '[usePromptLiveSession] failed to show live-session failure
dialog:',
+             dialogError,
+         );
+     }
+
+     // Closing is attempted even when the dialog failed, and close failures are
contained so
+     // this detached async bootstrap task cannot produce an unhandled
rejection.
+     try {
+         await closeSelf();
+     } catch (closeError) {
+         console.error(
+             '[usePromptLiveSession] failed to close failed live-session
panel:',
+             closeError,
+         );
+     }
+
-     void (async () => {
-
+     void (async () => {
+         // Availability branches are policy failures, not malformed snapshots
or transport failures.
+         // Preserve that classification across the shared catch path so product
adapters choose
+         // their dedicated unavailable title for both not-ready and not-found
responses.
+         let unavailableFailure = false;
+
-         if (!bootstrap.presentationAdapter.retryNotReady) {
-
+         if (!bootstrap.presentationAdapter.retryNotReady) {
+             unavailableFailure = true;
+
-         if (envelope.kind === 'not_found') {
-
+         if (envelope.kind === 'not_found') {

```

```

+                                     unavailableFailure = true;
+
-                                     } catch (error) {
-                                         if (ownsOpenAttempt(attemptId)) await fail(error);
-
+                                     } catch (error) {
+                                         if (ownsOpenAttempt(attemptId)) {
+                                             await fail(error, unavailableFailure);
+                                         }
+
+

```

[AI Code/src/frontend/src/app/prompt/liveSession/core/usePromptLiveSession.hook.ts](#)

Copy

Review

Accept



✓ patching file: [useAiStream.hook.ts](#)

[AI Code/src/frontend/src/messageBus/useAiStream.hook.ts](#)

Copy

Review

Accept

PatchFull file

```

-
-         // Capture the exact accepted attempt before consuming the capability. A
refusal may restore
-         // Stop only while this same attempt still owns the hook; a terminal or
successor attempt wins.
-
+
+         // Capture the exact accepted attempt before consuming the capability. Only a
retryable viewer
+         // refusal or transport exception may restore Stop while this same attempt
still owns the hook.
+
-
-         const runRef = identityAtRequest.runRef;
-
+         const runRef = identityAtRequest.runRef;
+         const usesViewerLeaseAbort =
+             currentStream.kind === 'liveSessionObserver' &&
+             currentStream.stopPolicy === 'viewerLease';
+
-
-         const result =
-             currentStream.kind === 'liveSessionObserver' &&
-             currentStream.stopPolicy === 'viewerLease'
-
+         const result =
+             usesViewerLeaseAbort
+
-
-         if (!result.accepted) {
-             throw new Error(
-                 'Exact stream Stop was refused because the run is no longer
abortable.',
-
+         if (!result.accepted) {
+             // A structured source refusal proves that the one-shot source claim
was already consumed
+             // or retired, so source Stop stays unavailable. Viewer leases retain
their historical
+             // same-attempt retry policy because their authorization is lease-
specific and renewable.

```

```

+         if (
+             usesViewerLeaseAbort &&
+             isSameStreamAttempt(
+                 activeIdentityRef.current,
+                 identityAtRequest.attemptIdentity,
+             )
+         ) {
+             updateTransportStopAvailable(true);
+         }
+         console.error(
+             '[useAiStream] exact stream Stop was refused because the run is no
longer abortable.',
+
+
+         '[useAiStream] exact stream Stop was refused because the run is no
longer abortable.',
+     );
+
+     '[useAiStream] exact stream Stop was refused because the run is no
longer abortable.',
+ );
+     return { accepted: false };
+
+
+     } catch (e) {
+         // Refusal and transport failure are recoverable while the same attempt is
still active.
+         // Restoring the capability lets the operator retry instead of leaving the
composer in a
+         // permanent abort handshake after a transient RPC failure.
+
+     } catch (e) {
+         // Only a transport exception is retryable for source-owned Stop. Restore
the capability while
+         // the exact attempt still owns the hook; stale failures must not re-enable
a successor route.
+

```

[AI Code/src/frontend/src/messageBus/useAiStream.hook.ts](#)

Copy

Review

Accept



patching file: [iaStreamHandlers.ts](#)

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

Copy

Review

Accept

PatchFull file

```

-     // rejected broadcast is reported once instead of being duplicated by each FIFO
item and cleanup.
-     const activityPublicationReady = activity.publication.then(
-         () => true,
-         (error: unknown) => {
+
+     // rejected broadcast is reported once instead of being duplicated by each FIFO
item and cleanup.
+     const activityPublicationReady = (async (): Promise<boolean> => {
+         try {
+             await activity.publication;
+             return true;
+         } catch (error) {

```

```
-         return false;
-     },
- );
-
+         return false;
+     }
+ })();
+
```

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

Copy

Review

Accept



✅ patching file: [liveSessionStreamHandlers.ts](#)

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

Copy

Review

Accept

PatchFull file

```
-     const activityPublicationReady: Promise<boolean> = allocation
-     ? allocation.publication.then(
-         () => true,
-         (error: unknown) => {
-             errors.push(normalizeError(error));
-             return false;
-         }
-     )
-
+     const activityPublicationReady: Promise<boolean> = allocation
+     ? (async (): Promise<boolean> => {
+         try {
+             await allocation.publication;
+             return true;
+         } catch (error) {
+             errors.push(normalizeError(error));
+             return false;
+         }
+     })()
+
```

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

Copy

Review

Accept



✅ patching file: [PromptLiveSessionPanelCoordinator.service.ts](#)

[AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelCoordinator.service.ts](#)

Copy

Review

Accept

PatchFull file

```
-     const previous = this.queues.get(key) ?? Promise.resolve();
-     const next = previous.then(operation, operation);
-
+     const previous = this.queues.get(key) ?? Promise.resolve();
+     const next = (async (): Promise<void> => {
+         try {
+             await previous;
+         } catch {

```

```

+          // Each enqueueing caller observes its own failure. A failed predecessor
must still release
+          // the locator lane so the next reconciliation operation can repair the
authoritative state.
+      }
+      await operation();
+  })();
+

```

[AI Code/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelCoordinator.service.ts](#)

Copy

Review

Accept

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

[AI Code/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionDetachedStop.service.ts](#)

Copy

Review

Accept

PatchFull file

```

-      store: SidebarStore;
-      liveSession: Pick<LiveSessionService, 'claimSourceAbort'>;
-
+      store: SidebarStore;
+      liveSession: Pick<
+        LiveSessionService,
+        'claimSourceAbort' | 'isLocatorDestructionClaimed'
+      >;
+
+
+
+
+      return { accepted: false };
+    }
+
+      return { accepted: false };
+    }
+    // Delete, Retry, and cleanup hold a locator-wide destructive claim across
their critical section.
+    // Detached UI Stop must not consume the exact source abort claim while one of
those owners runs.
+    if (
+      this.deps.liveSession.isLocatorDestructionClaimed(runRef.locator)
+    ) {
+      return { accepted: false };
+    }
+

```

[AI Code/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionDetachedStop.service.ts](#)

Copy

Review

Accept

  patching file:  [liveSession.programming-doc-readme.md](#)

[AI Code/src/extension/src/aiTools/orch/liveSession/liveSession.programming-doc-readme.md](#)

Copy

Review

Accept

PatchFull file

Abort is claimed atomically for either an active observer lease or an exact source-owned reservation/record. Claiming synchronously latches `abortRequested`; the returned callback must be invoked without an intervening `await`. The normal-Prompt direct source

route permanently retires Stop after either an accepted claim or a structured backend refusal; only a transport exception restores it for retry. Generic active-transport routes retain their existing same-attempt retry policy. The detached Activity Stop keeps local pending state and restores it only when its exact RPC is refused or fails; an accepted claim updates the shared Activity latch before abort. When rendering already became idle but the transport capability remains stoppable, the handshake stays armed during the RPC and closes on its accepted response because no later `isStreaming` transition exists. A normal-Prompt source Stop latch is also released when replay converges to a stable failure, because that surface must keep Retry usable after a pre-baseline source retirement. Reservation snapshots expose their real source/observer ownership so panel coordination and Activity rehydration never reinterpret an EasyAgents reservation as a detached normal Prompt source. The registry emits deeply immutable advisory wake-ups only after complete observer-inactive and destruction-claim-release mutations, and isolates listener failures. When the last observer lease disappeared before EasyAgents completed its business handoff, the handoff emits the observer-inactive wake-up again so deferred panel cleanup is retried after the final authority transition. Exact sound state is retired for every source-handler finalization, including `stale`, because the full runRef prevents cleanup from touching a successor source.

Abort is claimed atomically for either an active observer lease or an exact source-owned reservation/record. Claiming synchronously latches `abortRequested`; the returned callback must be invoked without an intervening `await`. Every normal-Prompt source-owned route, whether direct before observer acceptance or carried by an active transport, permanently retires Stop after either an accepted claim or a structured backend refusal; only a transport exception restores it for retry. Viewer-lease active transports retain their existing same-attempt retry policy. The detached Activity Stop keeps local pending state and restores it only when its exact RPC is refused or fails; an accepted claim updates the shared Activity latch before abort. The detached route refuses synchronously while a locator-wide destruction claim is held, so Delete, Retry, or cleanup retains exclusive ownership of its critical section. When rendering already became idle but the transport capability remains stoppable, the handshake stays armed during the RPC and closes on its accepted response because no later `isStreaming` transition exists. A normal-Prompt source Stop latch is also released when replay converges to a stable failure, because that surface must keep Retry usable after a pre-baseline source retirement. Reservation snapshots expose their real source/observer ownership so panel coordination and Activity rehydration never reinterpret an EasyAgents reservation as a detached normal Prompt source. The registry emits deeply immutable advisory wake-ups only after complete observer-inactive and destruction-claim-release mutations, and isolates listener failures. When the last observer lease disappeared before EasyAgents completed its business handoff, the handoff emits the observer-inactive wake-up again so deferred panel cleanup is retried after the final authority transition. Exact sound state is retired for every source-handler finalization, including `stale`, because the full runRef prevents cleanup from touching a successor source.

[AICode/src/extension/src/aiTools/orch/liveSession/liveSession.programming-doc-readme.md](#)

Copy

Review

Accept



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

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

Copy

Review

Accept

PatchFull file

```
-  
- import { beforeEach, describe, it, jest } from '@jest/globals';  
-  
+  
+ import { afterEach, beforeEach, describe, it, jest } from '@jest/globals';  
+
```

```

- import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
- import type { PromptRuntimeMode } from '../../PromptLiveSession.types.js';
-
+ import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
+ import type {
+   PromptLiveSessionReadyBootstrap,
+   PromptRuntimeMode,
+ } from '../../PromptLiveSession.types.js';
+
-
- const setNextTurnIdMock = jest.fn<(value: string | undefined) => void>();
-
+ const setNextTurnIdMock = jest.fn<(value: string | undefined) => void>();
+ const closeSelfMock = jest.fn<() => Promise<void>>();
+ const showVscAlertDialogMock =
+   jest.fn<(args: unknown) => Promise<void>>();
+
-
- jest.unstable_mockModule('../../../../messageBus/closeSelf.util.js', () => ({
-   closeSelf: async () => {},
-
+ jest.unstable_mockModule('../../../../messageBus/closeSelf.util.js', () => ({
+   closeSelf: closeSelfMock,
+
-
-   '../../../../vsc-design-system/popups/VscAlertDialog.js',
-   () => ({ showVscAlertDialog: async () => {} })),
-
+   '../../../../vsc-design-system/popups/VscAlertDialog.js',
+   () => ({ showVscAlertDialog: showVscAlertDialogMock })),
+
-
-   await import('../../adapters/NormalPromptLiveSession.adapter.js');
-
+   await import('../../adapters/NormalPromptLiveSession.adapter.js');
+ const { bindEasyAgentsBatchLiveSession } =
+   await import('../../adapters/EasyAgentsBatchLiveSession.adapter.js');
+
-
-   locator,
-   );
-
+   locator,
+   );
+ const easyAgentsBootstrap = bindEasyAgentsBatchLiveSession(
+   {
+     openMode: 'liveSession',
+     liveSessionTurnId: 'easy-agents-turn',
+     liveSessionViewerLeaseId: 'easy-agents-lease',
+     liveSessionClosePolicy: 'stopOnLastViewerClose',
+     liveSessionSourceId: 'easy-agents-turn',
+     liveSessionAdapter: 'easyAgentsBatch',
+     easyAgentsBatchLineNumber: '7',
+   },
+   locator,
+   );
+
-
-   startLiveObserver: () => AiStreamStartResult = refuseLiveObserver,
-
+   startLiveObserver: () => AiStreamStartResult = refuseLiveObserver,
+   activeBootstrap: PromptLiveSessionReadyBootstrap = bootstrap,
+
-
-   enabled: true,
-   bootstrap,

```

```

-
+     enabled: true,
+     bootstrap: activeBootstrap,
+
-
+     jest.clearAllMocks();
-
+     jest.clearAllMocks();
+     callLocalApiMock.mockReset();
+     closeSelfMock.mockReset();
+     closeSelfMock.mockResolvedValue(undefined);
+     showVscAlertDialogMock.mockReset();
+     showVscAlertDialogMock.mockResolvedValue(undefined);
+
-
+     showVscAlertDialogMock.mockResolvedValue(undefined);
-   });
-
+     showVscAlertDialogMock.mockResolvedValue(undefined);
+   });
+
+   afterEach(() => {
+     jest.restoreAllMocks();
+   });
+
-   });
-
-   it('drops a stale Open continuation after unmount without reading or mutating
hydration state', async () => {
-
+   });
+
+   it.each(['not_ready', 'not_found'] as const)(
+     'uses the EasyAgents unavailable title for %s and closes the viewer',
+     async (kind) => {
+       callLocalApiMock.mockResolvedValueOnce({ kind });
+
+       renderHook(() =>
+         useHydrationHarness(
+           refuseLiveObserver,
+           easyAgentsBootstrap,
+         ),
+       );
+
+       await waitFor(() =>
+         expect(showVscAlertDialogMock).toHaveBeenCalledWith(
+           expect.objectContaining({
+             title: 'Live watcher unavailable',
+           }),
+         ),
+       );
+       expect(closeSelfMock).toHaveBeenCalledTimes(1);
+     },
+   );
+
+   it('uses the EasyAgents bootstrap-failure title for non-availability errors', async
() => {
+     callLocalApiMock.mockRejectedValueOnce(
+       new Error('EasyAgents Open transport failed.'),
+     );
+
+     renderHook(() =>
+       useHydrationHarness(refuseLiveObserver, easyAgentsBootstrap),
+     );

```

```

+
+     await waitFor(() =>
+       expect(showVscAlertDialogMock).toHaveBeenCalledWith(
+         expect.objectContaining({
+           title: 'Live watcher bootstrap failed',
+         }),
+       ),
+     );
+     expect(closeSelfMock).toHaveBeenCalledTimes(1);
+   });
+
+   it('still closes the EasyAgents viewer when the failure dialog rejects', async ()
=> {
+     jest.spyOn(console, 'error').mockImplementation(() => {});
+     callLocalApiMock.mockResolvedValueOnce({ kind: 'not_found' });
+     showVscAlertDialogMock.mockRejectedValueOnce(
+       new Error('Dialog renderer unavailable.'),
+     );
+
+     renderHook(() =>
+       useHydrationHarness(refuseLiveObserver, easyAgentsBootstrap),
+     );
+
+     await waitFor(() => expect(closeSelfMock).toHaveBeenCalledTimes(1));
+     expect(showVscAlertDialogMock).toHaveBeenCalledWith(
+       expect.objectContaining({ title: 'Live watcher unavailable' }),
+     );
+   });
+
+   it('drops a stale Open continuation after unmount without reading or mutating
hydration state', async () => {

```

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

Copy

Review

Accept



patching file: [useAiStream.hook.promptSourceIdentity.test.ts](#)

[AI Code/src/frontend/src/messageBus/ test /useAiStream.hook.promptSourceIdentity.test.ts](#)

Copy

Review

Accept

PatchFull file

```

-     container.remove();
-
+     container.remove();
+     jest.restoreAllMocks();
+
+
+
+   it('restores the exact Stop capability after refusal or RPC failure while the
attempt remains active', async () => {
+
+
+   it('permanently retires source Stop after a structured backend refusal', async ()
=> {
+     jest.spyOn(console, 'error').mockImplementation(() => {});
+
+
+
+     kind: 'promptQuery',
+     turnId: 'turn-retry-stop',
+

```

```

+         kind: 'promptQuery',
+         turnId: 'turn-refused-stop',
+
-     expect(harnessApiRef.current?.transportStopAvailable).toBe(true);
-
-     callLocalApiMock.mockRejectedValueOnce(
-
+     expect(harnessApiRef.current?.transportStopAvailable).toBe(false);
+
+     // A structured refusal is authoritative for source ownership. Repeated UI
actions must not
+     // issue another abort RPC for the already-retired one-shot source claim.
+     await act(async () => {
+         await expect(api.stop()).resolves.toEqual({ accepted: false });
+     });
+     expect(callLocalApiMock).toHaveBeenCalledTimes(1);
+ });
+
+ it('restores source Stop after a transport exception while the exact attempt
remains active', async () => {
+     jest.spyOn(console, 'error').mockImplementation(() => {});
+     await act(async () => {
+         root.render(React.createElement(Harness));
+     });
+     const api = harnessApiRef.current;
+     if (!api) throw new Error('HarnessApi not initialized');
+
+     let startResult: AiStreamStartResult | undefined;
+     act(() => {
+         startResult = api.start({
+             kind: 'promptQuery',
+             turnId: 'turn-transport-stop',
+             storageScopeRelPath: '',
+             aiQuery: { type: 'prompt', promptValue: 'Keep running' },
+             aiQueryOptions: {},
+         });
+     });
+     if (!startResult?.accepted) {
+         throw new Error('Expected one accepted prompt source.');
```

[AICode/src/frontend/src/messageBus/ test /useAiStream.hook.promptSourceIdentity.test.ts](#)

Copy

Review

Accept



patching file: [useAiStream.hook.liveSessionObserver.test.ts](#)

[AICode/src/frontend/src/messageBus/ test /useAiStream.hook.liveSessionObserver.test.ts](#)

Copy

Review

Accept

PatchFull file

```

-
-   it('keeps one source terminal when a later transport error finalizes the attempt',
async () => {
-
+
+   it('keeps the EasyAgents viewer-lease Stop retryable after a structured refusal',
async () => {
+     jest.spyOn(console, 'error').mockImplementation(() => {});
+     await act(async () => {
+       root.render(React.createElement(Harness));
+     });
+     const api = harnessApiRef.current;
+     if (!api) throw new Error('Hook harness was not initialized.');
```

+

```

+     let startResult: AiStreamStartResult | undefined;
+     act(() => {
+       startResult = api.start({
+         kind: 'liveSessionObserver',
+         runRef: RUN_REF,
+         viewerLeaseId: 'retryable-viewer-lease',
+         replayMode: 'replaceTurnAssistants',
+         stopPolicy: 'viewerLease',
+       });
+     });
+     if (!startResult?.accepted) {
+       throw new Error('Expected the observer start to be accepted.');
```

+

```

+     }
+
+     callLocalApiMock.mockResolvedValueOnce({ accepted: false });
+     await act(async () => {
+       await expect(api.stop()).resolves.toEqual({ accepted: false });
+     });
+     expect(harnessApiRef.current?.transportStopAvailable).toBe(true);
+
+     // Viewer authorization keeps the historical retry contract, unlike one-shot
source ownership.
+     await act(async () => {
+       await expect(api.stop()).resolves.toEqual({ accepted: true });
+     });
+     expect(callLocalApiMock).toHaveBeenCalledTimes(2);
+     expect(harnessApiRef.current?.transportStopAvailable).toBe(false);
+   });
+
+   it('keeps one source terminal when a later transport error finalizes the attempt',
async () => {
+

```

[AICode/src/frontend/src/messageBus/ test /useAiStream.hook.liveSessionObserver.test.ts](#)

Copy

Review

Accept

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

[AICode/src/extension/src/extension/panels/prompt/liveSession/ test /PromptLiveSessionDetachedStop.service.test.ts](#)

Error

([what is it?](#))

  reading project:  [PromptLiveSessionDetachedStop.service.test.ts](#)

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

[AICode/src/extension/src/extension/panels/prompt/liveSession/ test /PromptLiveSessionDetachedStop.service.test.ts](#)

Copy

Review

Accept

PatchFull file

```
-
- describe('PromptLiveSessionDetachedStopService', () => {
-
+
+ type DetachedStopLiveSession = Pick<
+   LiveSessionService,
+   'claimSourceAbort' | 'isLocatorDestructionClaimed'
+ >;
+
+ /** Build strict live-session dependencies with destructive coordination enabled by
+  * default. */
+ function createLiveSessionDeps(
+   claimSourceAbort: LiveSessionService['claimSourceAbort'],
+   isLocatorDestructionClaimed: LiveSessionService['isLocatorDestructionClaimed'] =
+   () => false,
+ ): DetachedStopLiveSession {
+   return { claimSourceAbort, isLocatorDestructionClaimed };
+ }
+
+ describe('PromptLiveSessionDetachedStopService', () => {
+


---


-   store,
-   liveSession: {
-     claimSourceAbort: () => ({
-       runRef: RUN,
-       abort: () => order.push('abort'),
-     }),
-   },
-
+   store,
+   liveSession: createLiveSessionDeps(() => ({
+     runRef: RUN,
+     abort: () => order.push('abort'),
+   })),
+


---


-   store,
-   liveSession: { claimSourceAbort: claim },
-
+   store,
+   liveSession: createLiveSessionDeps(claim),
+


---


-   const claim = jest.fn<LiveSessionService['claimSourceAbort']>();
-   const service = new PromptLiveSessionDetachedStopService({
-     store: createSidebarStore([item]),
-     liveSession: { claimSourceAbort: claim },
-
+   const claim = jest.fn<LiveSessionService['claimSourceAbort']>();
+   const service = new PromptLiveSessionDetachedStopService({
+     store: createSidebarStore([item]),
+     liveSession: createLiveSessionDeps(claim),
+


---


-     store: createSidebarStore([item]),
-     liveSession: { claimSourceAbort: claim },
-
+     store: createSidebarStore([item]),
+     liveSession: createLiveSessionDeps(claim),
+


---


```

```

-         store: createSidebarStore([createClosedSourceItem()]),
-         liveSession: { claimSourceAbort: () => undefined },
-
+         store: createSidebarStore([createClosedSourceItem()]),
+         liveSession: createLiveSessionDeps(() => undefined),
+         activity: { onSourceStopRequested: activity },
+     });
+
+     expect(service.stop(RUN)).toEqual({ accepted: false });
+     expect(activity).not.toHaveBeenCalled();
+ });
+
+ it('refuses detached Stop while a locator destruction claim owns the critical
section', () => {
+     const claim = jest.fn<LiveSessionService['claimSourceAbort']>();
+     const isLocatorDestructionClaimed =
+         jest.fn<LiveSessionService['isLocatorDestructionClaimed']>()
+         () => true,
+     );
+     const activity = jest.fn<() => boolean>();
+     const service = new PromptLiveSessionDetachedStopService({
+         store: createSidebarStore([createClosedSourceItem()]),
+         liveSession: createLiveSessionDeps(
+             claim,
+             isLocatorDestructionClaimed,
+         ),
+     },
+     activity: { onSourceStopRequested: activity },
+ });
+
+     expect(service.stop(RUN)).toEqual({ accepted: false });
+
+     ),
+     activity: { onSourceStopRequested: activity },
+ });
+
+     expect(service.stop(RUN)).toEqual({ accepted: false });
+     expect(isLocatorDestructionClaimed).toHaveBeenCalledWith(LOCATOR);
+     expect(claim).not.toHaveBeenCalled();
+
+
+     store: createSidebarStore([createClosedSourceItem()]),
+     liveSession: {
+         claimSourceAbort: () => ({
+             runRef: RUN,
+             abort: () => order.push('abort'),
+         }),
+     },
+
+     store: createSidebarStore([createClosedSourceItem()]),
+     liveSession: createLiveSessionDeps(() => ({
+         runRef: RUN,
+         abort: () => order.push('abort'),
+     })),
+

```

[AI Code/src/extension/src/extension/panels/prompt/liveSession/_test_/PromptLiveSessionDetachedStop.service.test.ts](#)

Copy

Review

Accept

  reading project:  [usePromptLiveSession.hook.ts](#) [L180-L335],  [usePromptLiveSession.hydraton.test.tsx](#),  [useAiStream.hook.ts](#) [L960-L1045],  [useAiStream.hook.promptSourceIdentity.test.ts](#) [L380-L455],  [useAiStream.hook.liveSessionObserver.test.ts](#) [L225-L325],  [iaStreamHandlers.ts](#) [L125-L160],  [liveSessionStreamHandlers.ts](#) [L70-L105],  [PromptLiveSessionPanelCoordinator.service.ts](#) [L410-L440],
 [PromptLiveSessionDetachedStop.service.ts](#),
 [PromptLiveSessionDetachedStop.service.test.ts](#),  [useAiStream.hook.promptSourceIdentity.test.ts](#) [L430-L475]
  patching file:  [useAiStream.hook.liveSessionObserver.test.ts](#)
[AICode/src/frontend/src/messageBus/_test_/useAiStream.hook.liveSessionObserver.test.ts](#)
Grouped
([what is it?](#))

  patching file:  [PromptLiveSessionDetachedStop.service.test.ts](#)
[AICode/src/extension/src/extension/panels/prompt/liveSession/_test_/PromptLiveSessionDetachedStop.service.test.ts](#)
Grouped
([what is it?](#))

  scanning git:  ["Intégrité des correctifs"](#),  ["Résumé des correctifs"](#)
  reading project:  ["Promesses handler normal"](#),  ["Promesses handler live"](#),  ["Promesses coordinateur"](#),  ["Instanciations Stop détaché"](#),  ["Façade claim destructif"](#),  ["Résultat Stop contrôleur"](#),  ["Handshake résultat Stop"](#),  [usePromptCancelController.hook.ts](#) [L90-L185],  [usePromptCancelController.hook.ts](#) [L185-L255],  [useAiStream.hook.ts](#) [L925-L970], ["Stop disponibilité rendu"](#), [Prompt.tsx](#) [L660-L730], ["État running input"](#), [Prompt.tsx](#) [L1225-L1270], [liveSession.programming-doc-readme.md](#) [L65-L85]
  patching file:  [useAiStream.hook.ts](#)
[AICode/src/frontend/src/messageBus/useAiStream.hook.ts](#)
Grouped
([what is it?](#))

  patching file:  [usePromptCancelController.hook.ts](#)
[AICode/src/frontend/src/app/prompt/Prompt/usePromptCancelController.hook.ts](#)
Copy
Review
Accept
PatchFull file

```
-  
-           // Ask the backend to stop streaming without tearing down locally; an accepted  
request keeps  
-           // the handshake armed until terminal metadata arrives. A refusal restores Stop  
immediately.  
+  
+           // Ask the backend to stop streaming without tearing down locally. An accepted  
request keeps  
+           // the handshake armed until terminal metadata arrives. A refusal clears this  
UI handshake;  
+           // the transport owner independently decides whether its exact Stop authority  
is retryable.
```

[AICode/src/frontend/src/app/prompt/Prompt/usePromptCancelController.hook.ts](#)
Copy
Review
Accept

  *patching file:*  [liveSession.programming-doc-readme.md](#)
[AICode/src/extension/src/aiTools/orch/liveSession/liveSession.programming-doc-readme.md](#)
Grouped
([what is it?](#))

  *reading project:*  [useAiStream.hook.liveSessionObserver.test.ts](#) [L265-L320],  [PromptLiveSessionDetachedStop.service.test.ts](#) [L75-L235],  [usePromptCancelController.hook.ts](#) [L180-L215],  [liveSession.programming-doc-readme.md](#) [L73-L80],  [usePromptLiveSession.hook.ts](#) [L330-L365],  [useAiStream.hook.ts](#) [L945-L1035]
  *scanning recent files*

Correctifs appliqués

Joel, **tous les correctifs issus de la vérification ont été codés directement**. Les zones les plus sensibles — arrêt exact, fermeture EasyAgents, claims destructifs et séquençement asynchrone — sont maintenant alignées. Le bouton Stop a désormais son propre service de sécurité rapprochée 😊.

J'ai également relu et synchronisé le [document d'architecture liveSession](#) avec le comportement final.

Résumé des changements

Zone	Correctif appliqué	Résultat	Complétude
Échec initial EasyAgents	Le dialogue d'erreur et la fermeture du panneau sont maintenant deux obligations indépendantes	Un échec du dialogue ne peut plus laisser un viewer défaillant ouvert	
Classification Open	<code>not_ready</code> et <code>not_found</code> conservent explicitement leur classification d'indisponibilité	Le titre Live watcher unavailable est utilisé au lieu du titre générique	
Stop source normal Prompt	Un refus backend structuré retire définitivement l'autorité Stop de cette source	Aucun nouvel appel d'abort n'est envoyé pour un claim source déjà consommé	
Erreur de transport du Stop	Une vraie exception RPC restaure l'autorité tant que la tentative exacte reste active	L'utilisateur peut réessayer après une panne de transport transitoire	
Stop viewer EasyAgents	Le comportement historique retryable après refus structuré est conservé	Aucun changement de contrat EasyAgents Delete, Retry et cleanup gardent l'exclusivité de leur section critique	
Stop détaché	Refus immédiat lorsqu'un claim destructif couvre la locator	Même ordre, même isolation des erreurs auxiliaires	
Promesses d'Activité	Remplacement des nouvelles chaînes <code>.then(...)</code> par <code>async/await</code>	Une opération échouée ne bloque toujours pas la réconciliation suivante	
File du coordinateur	Remplacement de <code>previous.then(operation, operation)</code> par une séquence <code>await</code> explicite	Les nouveaux contrats sont matérialisés dans les tests	
Tests de régression	Ajout des scénarios manquants et séparation claire des politiques source/viewer		

Zone	Correctif appliqué	Résultat	Complétude
Documentation	Contrat source, viewer et claim destructif clarifié	Code, tests et documentation racontent désormais la même histoire	✓

Robustesse de l'échec EasyAgents

Dans [usePromptLiveSession.hook.ts](#) :

- les réponses `not_ready` et `not_found` transportent maintenant leur classification jusqu'au gestionnaire d'échec ;
- un viewer EasyAgents utilise donc correctement son titre d'indisponibilité ;
- le dialogue d'erreur est tenté en premier ;
- la fermeture du panneau est ensuite tentée **même si le dialogue rejette** ;
- une erreur de dialogue ou de fermeture est contenue et ne produit plus de rejet asynchrone non géré ;
- le comportement normalPrompt `directAfterReplay` reste inchangé : il conserve son écran de Retry au lieu de fermer le panneau.

Les régressions correspondantes sont couvertes dans [usePromptLiveSession.hydratation.test.tsx](#) :

- `not_ready` → titre d'indisponibilité et fermeture ;
- `not_found` → titre d'indisponibilité et fermeture ;
- erreur de transport Open → titre d'échec du bootstrap ;
- rejet du dialogue → fermeture toujours exécutée.

Contrat exact du Stop

Dans [useAiStream.hook.ts](#), les deux formes d'autorité sont maintenant explicitement séparées.

Source normal Prompt

- `accepted: true` consomme le Stop ;
- `accepted: false` est une réponse backend autoritaire : le Stop source reste retiré ;
- une seconde action ne génère pas un second appel RPC ;
- seule une **exception de transport** restaure le Stop pour la même tentative exacte ;
- une erreur tardive appartenant à une ancienne tentative ne peut toujours pas réactiver le Stop d'une tentative successeur.

Cette politique est couverte dans [useAiStream.hook.promptSourceIdentity.test.ts](#) avec deux scénarios désormais indépendants :

1. refus structuré définitif ;
2. exception de transport retryable.

Viewer EasyAgents

Le Stop lease-specific conserve son comportement historique :

- un refus structuré restaure la capacité pour la même tentative ;
 - le second appel peut réussir ;
 - la route utilise toujours l'endpoint viewer avec sa lease exacte.
- Ce non-régressif est couvert dans [useAiStream.hook.liveSessionObserver.test.ts](#).

Le commentaire de maintenance du contrôleur dans [usePromptCancelController.hook.ts](#) précise également que :

- le contrôleur retire toujours son handshake UI après un refus ;
- le propriétaire du transport décide séparément si l'autorité Stop reste retryable.

Protection contre les claims destructifs

Dans [PromptLiveSessionDetachedStop.service.ts](#) :

- la projection Activité exacte est toujours validée en premier ;
- le service vérifie ensuite synchroniquement si la locator est couverte par un claim destructif ;
- dans ce cas, il refuse avant de réclamer l'abort source ;
- aucune mutation `stopRequested` et aucun abort ne sont déclenchés ;
- la vérification et la prise éventuelle du claim d'abort restent sans `await`, donc sans fenêtre d'interleaving.

Le scénario est matérialisé dans [PromptLiveSessionDetachedStop.service.test.ts](#) :

- claim destructif actif ;
- refus exact du Stop détaché ;
- absence d'appel au claim source ;
- absence de mutation Activité.

Conformité `async/await`

Les trois usages identifiés ont été remplacés sans modifier les algorithmes environnants :

Emplacement	Conservation comportementale	État
iaStreamHandlers.ts	La publication Activité démarre immédiatement, son rejet est mémorisé une seule fois et les chunks continuent	✓
liveSessionStreamHandlers.ts	Le backlog reste indépendant d'un éventuel échec de publication Activité	✓
PromptLiveSessionPanelCoordinator.service.ts	La file par locator attend son prédécesseur, ignore uniquement son échec déjà attribué, puis exécute l'opération suivante	✓

Garanties préservées

- ✓ Aucun fichier `*.generated.*` modifié.
- ✓ Aucun changement d'algorithme dans Queue A ou Queue B.
- ✓ Aucun changement de `PromptStreamLockStore`.
- ✓ Aucun changement des sanitizers.
- ✓ Aucun `any`, `cast`, `TODO` ou contournement ESLint ajouté.
- ✓ Aucun import dynamique de production ajouté.
- ✓ Les callbacks React ajoutés ou touchés restent dans les structures existantes.
- ✓ Le comportement one-shot des sources et le comportement retryable des viewers sont maintenant explicitement distincts.
- ✓ Le Stop détaché ne peut plus concurrencer une opération destructive sur la même locator.

Fichiers modifiés

Groupe

	Fichiers
Runtime	
frontend	usePromptLiveSession.hook.ts , useAiStream.hook.ts , usePromptCancelControl.hook.ts
liveSession	
Tests frontend	usePromptLiveSession.hydratation.test.tsx , useAiStream.hook.promptSourceId.test.tsx
Relais et coordination	iaStreamHandlers.ts , liveSessionStreamHandlers.ts , PromptLiveSessionPanelCoordinator.service.ts
extension	
Stop détaché	PromptLiveSessionDetachedStop.service.ts , PromptLiveSessionDetachedStop.hook.ts
Documentation	liveSession.programming-doc-readme.md
Nom du commit	

text

Copy

fix(live-session): harden failure shutdown and stop ownership

🎵 Les viewers ferment sans se perdre 🦋 Les Stops gardent désormais la bonne frontière 🎵

</ASSISTANT>

SHOW ALLPage 1 / 1