

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=6b49bcfbf724472e9ee9b2fe377567c0-VS-V%C3%A9rifier-Session-<USER>

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

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

  scanning recent files

  scanning git: 🗑️ ["État Git courant"](#), 🗑️ ["Fichiers modifiés"](#), 🗑️ ["Résumé du diff"](#), 🗑️ ["Révision actuelle"](#)

  reading project: 📁 g ["Docs liveSession"](#), 📁 g ["Docs Prompt"](#), 📁 g ["Docs message bus"](#), 📁 g ["Docs EasyAgents"](#), 📄 [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: 🗑️ ["Inventaire du commit"](#), 🗑️ ["Statistiques du commit"](#)

  reading

project:

📄 [LiveSessionRun.type.ts](#),
📄 [LiveSessionPanel.type.ts](#),
📄 [LiveSessionOpenResult.type.ts](#),
📄 [LiveSessionReplay.type.ts](#),
📄 [SidebarStore.ts](#),
📄 [SidebarUiMessages.ts](#),
📄 [MessageRegistry.ts](#),
📄 [aiStopSuffix.util.ts](#),
📄 [parseSlimbkStreamBodyError.util.ts](#),
📄 [SlimbkResult.type.ts](#), 📄 [SlimbkEndpoint.ts](#), 📄 [Slimbk.ts](#), 📄 [slimbkBuildHeader.ts](#)

  scanning git: 🗑️ ["Qualité du diff"](#)

  reading

project:

📄 [LiveSessionRecord.type.ts](#),
📄 [LiveSessionRegistry.service.ts](#),
📄 [LiveSessionService.ts](#),
📄 [LiveSessionOpen.service.ts](#),
📄 [LiveSessionSourceLifecycle.service.ts](#),
📄 [NormalPromptLiveSessionSource.service.ts](#),
📄 [PromptLiveSessionPanelQuery.util.ts](#),
📄 [PromptLiveSessionPanelCoordinator.service.ts](#),
📄 [PromptLiveSessionDetachedStop.service.ts](#),
📄 [PromptThreadCleanup.manager.ts](#),
📄 [SidebarQueryRunningBridge.ts](#),
📄 [iaStreamHandlers.ts](#),
📄 [liveSessionStreamHandlers.ts](#),
📄 [stream.endpoint.ts](#),
📄 [abortSource.endpoint.ts](#),
📄 [abort.endpoint.ts](#),
📄 [openPanel.util.ts](#), 📄 [delete.endpoint.ts](#), 📄 [EasyAgentsBatchRunRetry.service.ts](#), 🕒 h ["Réouverture panneau live"](#), 🕒 e ["Tests reveal panneau"](#), 🕒 e ["Tests descripteur"](#), 🕒 h ["Watch live query"](#), 🕒 h ["Frontend Watch live"](#), 🕒 h ["Preuves reveal live"](#), 📄 [EasyAgentsBatch.liveWatch.test.tsx](#) [L217-L260], 🕒 h ["Start transactionnel"](#), 🕒 h ["Replay normal Prompt"](#), 🕒 h ["Barrière fallback"](#), 🕒 h ["Commandes start commit"](#), 🕒 h ["Queues identité exacte"](#), 🕒 h ["Hydratation preserveLocal"](#), 🕒 h ["Bootstrap query brute"](#), 🕒 h ["Activité Stop détaché"](#), 🕒 h ["Start et finalisation"](#), 🕒 h ["Replay et handoff"](#), 🕒 h ["Barrière compositeur"](#), 🕒 h ["Commandes transaction"](#),
📄 [PromptLiveSessionBootstrap.adapter.ts](#),

[NormalPromptLiveSession.adapter.ts](#),
[AiStreamStartArgs.type.ts](#),
[SidebarActivityQuery.tsx](#),
[applyHydratedPromptSessionState.util.ts](#), [NormalPromptLiveSession.adapter.test.ts](#), 🧑‍🔬 e "[Composer replay échoué](#)", 🧑‍🔬 h "[Interaction composer échec](#)", 🧑‍🔬 e "[Test composer Retry](#)", [PromptInput.batchLiveViewerMode.test.tsx](#) [L1-L120], 🧑‍🔬 e "[Restauration focus](#)", 🧑‍🔬 e "[Effet focus compositeur](#)", [Prompt.tsx](#) [L1-L330], [usePromptLiveSession.hook.ts](#), [PromptLiveSession.types.ts](#), [useAiStream.hook.ts](#) [L217-L541], [useAiStream.hook.ts](#) [L865-L1018], [useAiStreamQueue.hook.ts](#) [L1-L220], [useAiStreamQueue.hook.ts](#) [L325-L610], [usePromptStreamDispatchGateQueue.hook.ts](#) [L1-L260], [usePromptStreamDispatchGateQueue.hook.ts](#) [L865-L1198], [usePromptRunController.finalization.hook.ts](#), 🧑‍🔬 h "[Finalisation contrôleur](#)",
[preparePromptContinuationStreaming.util.ts](#),
[preparePromptStreamingTurn.util.ts](#), [usePromptThinkingLifecycle.hook.ts](#), 🧑‍🔬 e "[Plan continuation replay](#)", 🧑‍🔬 e "[Bootstrap observer](#)", 🧑‍🔬 e "[Réinitialisation séparateur](#)", 🧑‍🔬 e "[Utilisations séparateur](#)", [usePromptToolUse.hook.ts](#), 🧑‍🔬 e "[Transitions activité](#)", 🧑‍🔬 e "[Transition Stop activité](#)", 🧑‍🔬 e "[Transition source finie](#)", 🧑‍🔬 e "[Transition panneau fermé](#)", 🧑‍🔬 e "[Réhydratation activité](#)", 🧑‍🔬 e "[Réconciliation activité](#)", 🧑‍🔬 h "[Idle source ouverte](#)", 🧑‍🔬 e "[Transitions activité](#)", 🧑‍🔬 e "[Fermeture activité](#)", 🧑‍🔬 e "[Réhydratation activité](#)", [SidebarStore.handlers.manager.ts](#), [SidebarStore.queryWindow.listeners.manager.ts](#) [L300-L390], [PromptQueryDoneSound.manager.ts](#), [extTypedMessageBus.util.ts](#), [extMessageBus.util.ts](#) [L1-L260], [extMessageBus.streamOriginContext.test.ts](#), [extMessageBus.util.ts](#) [L260-L383],
[OrchMessagesMemoryAssistantPatch.ts](#),
[OrchMessagesMemoryAssistantPatch.errorSuffix.test.ts](#),
[parseSlimbkStreamBodyError.util.test.ts](#), [Slimbk.streamingErrorHandler.test.ts](#), 🧑‍🔬 h "[Test double ouverture](#)", 🧑‍🔬 h "[Test intégration reveal live](#)", [usePromptSessionHydration.hook.ts](#), [usePromptSessionLocator.hook.ts](#) [L1-L140], [usePromptSessionLocator.hook.ts](#) [L220-L303], [EasyAgentsBatchLiveSessionAdapter.service.ts](#), 🧑‍🔬 e "[RunRef EasyAgents](#)", 🧑‍🔬 e "[SourceId EasyAgents](#)", [Prompt.tsx](#) [L620-L780], [LiveSessionReplay.service.ts](#), 🧑‍🔬 e "[Release viewer EasyAgents](#)", 🧑‍🔬 e "[Release lease adaptateur](#)", 🧑‍🔬 e "[Chemin last viewer](#)", [SidebarStore.core.manager.ts](#) [L1-L160], [SidebarStore.core.manager.ts](#) [L150-L200]
🔧 ✅ scanning git: 🧑‍🔬 e "[Interdits du patch](#)", 🧑‍🔬 e "[TODO du patch](#)", ❌ 🧑‍🔬 e "[Imports dynamiques](#)", ❌ 🧑‍🔬 e "[Require du patch](#)", 🧑‍🔬 e "[Any explicites](#)", 🧑‍🔬 e "[Types any du patch](#)", 🧑‍🔬 e "[Fichiers générés](#)", 🧑‍🔬 e "[Imports dynamiques](#)", 🧑‍🔬 e "[Require du patch](#)", 🧑‍🔬 e "[Assertions any](#)", 🧑‍🔬 e "[Annotations any](#)", 🧑‍🔬 e "[Ignore ESLint](#)", 🧑‍🔬 e "[Non-null ajoutés](#)", 🧑‍🔬 e "[Casts unknown any](#)"
🔧 ✅ reading project: 🧑‍🔬 e "[cleanupOnPromptDispose](#)", [PromptInput.tsx](#) [L100-L390], 🧑‍🔬 e "[Usages openMode](#)", 🧑‍🔬 e "[Prop disableAfterClick](#)", 🧑‍🔬 h "[Test Stop runtime absent](#)", [abortSource.endpoint.test.ts](#) [L1-L140], 🧑‍🔬 h "[Test séparateur tentative](#)", 🧑‍🔬 e "[Tests thinking identité](#)", [usePromptThinkingLifecycle.continuation.test.tsx](#) [L1-L125]

Vérification effectuée 🧑‍🔬

Joel, cette tranche est presque un **petit système distribué déguisé en onglet Prompt** 😊 — et, franchement, le niveau d'architecture et de couverture est impressionnant.

J'ai vérifié le dépôt propre au HEAD **87769c86**, puis recoupé les contrats, les chemins d'exécution, les courses asynchrones et les tests avec les documents de maintenance suivants :

- [documentation générique liveSession](#)
- [documentation EasyAgents](#)
- [invariant d'isomorphisme streaming/hydratation](#)
- [invariants thinking/hourglass](#)
- [documentation Queue B](#)
- [documentation fallback persist](#)

Les contrôles de qualité du diff ne montrent **aucun ajout de TODO, eslint-disable, eslint-ignore, require() ou any explicite évident**. Les imports dynamiques ajoutés sont confinés aux tests ESM. Le contrôle de whitespace Git est propre.

1) La spécification entière est-elle implémentée ?

Réponse

❌ Non, pas entièrement.

Mon estimation est :

Mesure	Estimation	Verdict
Présence structurelle des modules et contrats demandés	98 %	✅
Conformité fonctionnelle générale	94 %	⚠️
Complétude globale de la spécification	96 %	⚠️
Validation possible sans correction	Non	❌
La quasi-totalité de la tranche existe effectivement :		
- identité exacte <code>locator + turnId + sourceId</code> ; - réservations synchrones, époques, retraites, tombstones et claims destructifs ; - origine webview exacte extension-only ; - source Prompt normale continuant après fermeture ; - replay backlog + tail ; - reconstruction <code>replaceTurnAssistants / continueAssistant</code> ; - démarrage frontend transactionnel ; - gardes <code>sourceId + streamInstanceId</code> dans Queue A et Queue B ; - protocole V1 premier-header ; - interception des enveloppes Slimbk post-header ; - persistance de <code>meta.error</code> avant append ; - Activité <code>open/closed</code>, Stop détaché, son exact ; - handoff direct et hydratation <code>preserveLocal</code> ; - fallback ack corrélé et démasquage sans effacement ; - adaptation EasyAgents et opérations destructives sous claim.		
Cependant, j'ai trouvé :		
- un bug matériel de conservation du lease/descripteur lorsque le panneau existe déjà et est seulement révélé ; - un bug matériel d'interactivité du compositeur après armement du replay puis échec/retry ; - un écart d'invariant mineur concernant le reset du séparateur thinking entre deux <code>streamInstanceId</code>.		
Conclusion Q1 : la tranche est très proche de la finalisation, mais elle ne peut pas être déclarée entièrement implémentée tant que les deux défauts matériels ne sont pas corrigés.		

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

Les 160 entrées de la spécification sont regroupées ci-dessous par **contrat technique cohérent**, plusieurs fichiers participant souvent au même invariant.

Domaine fonctionnel	État	Complétude estimée	Vérification
Identité exacte <code>turnId / sourceId</code>	✅	100 %	Le runRef, la run key et les schémas stricts sont correctement définis dans le contrat de run . La key reste indépendante du <code>sourceId</code> .
Descripteur de panneau strict et partagé	✅	100 %	L'union EasyAgents/normalPrompt, les politiques de fermeture, les paramètres privés et la contrainte <code>sourceId === turnId</code> EasyAgents sont correctement définis dans le contrat de panneau .
Contrat Open, enveloppe sélective et replay mode	✅	100 %	Les branches <code>live</code> , <code>terminal</code> , <code>not_ready</code> , <code>not_ready</code> et les champs interdits présents à <code>undefined</code> et <code>not_ready</code> l'enveloppe à session opaque sont implémentés dans le contrat Open et le contrat replay .
Parsing brut de query extension + frontend	✅	99 %	Les doublons, paramètres inconnus, descripteurs et locators ambigus sont rejetés dans le parseur extension et le dispatcher frontend .
Frontière React externe des queries	✅	100 %	le runtime Prompt sélectionne une frontière in

Domaine fonctionnel	État	Complétude estimée	Vérification
invalides			avant de monter contrôleur, hydratation ou au
Snapshot brut unique et figé	✓	100 %	Le snapshot est lu une seule fois via le hook d Prompt et transmis au dispatcher pur.
Origine webview exacte non sérialisée	✓	100 %	Le contexte <code>HtmlPage + Webview</code> est créé par le bus extension et le bus typé . Les deux h comparent bien la webview émettrice au pann courant.
Démarrage frontend transactionnel	✓	99 %	useAiStream revendique le single-flight synchroniquement, génère les identités après acceptation et ne modifie rien lors d'un refus.
Plans purs et commit après acceptation	✓	99 %	Send, SPECIF, Retry, Edit, Compact, Reply et suivent le plan demandé dans les commandes préparation replace et la préparation continuat
Queue A / Queue B et gardes de tentative	✓	98 %	Les deux files transportent <code>sourceId +</code> <code>streamInstanceId</code> , retiennent les erreurs e rejetent le travail stale et attendent le commit dans Queue A , Queue B et le propriétaire du s
Source Prompt normale et durabilité terminale	✓	99 %	Capture de baseline, promotion après priming persistance de chaque erreur avant append, ter synthétique conditionnel et absence de finalisa prématurée sont correctement placées dans la normalPrompt .
Lifecycle lazy Readable partagé	✓	99 %	Priming, premier <code>next()</code> , cleanup, settlement agrégation déterministe sont isolés dans le life partagé .
Protocole V1 et erreur Slimbk post-header	✓	98 %	L'encodage/décodage V1 et la détection des e body sont implémentés dans les fabriques de terminal et le parseur body Slimbk .
Registre : réservations, époques, retraites, tombstones	✓	98 %	La séparation des états, la priorité missing-bas cinq résultats de finalisation, les leases et les v immuables sont bien présents dans le registre liveSession .
Claims destructifs et génération ABA distincte	✓	99 %	Acquisition/libération bumpent une génération Open revalide les deux générations et l'évictio handle exact.
Open/replay/handoff backend	✓	98 %	le service Open recharge l'enveloppe récente, le replay mode, valide le terminal structurel et après I/O.
Replay backlog + tail complet	✓	100 %	le service Replay conserve le buffer complet, copie quadratique et attend handoff/éviction p le simple settlement.
FIFO extension, terminal unique et outcome transport	✓	98 %	le handler source normal et le handler de repla les enveloppes avant les chunks et agrègent le après drainage.
Activité : Idle/open et Running/source/open closed	✓	99 %	Les transitions et la reconstitution suivent l'un dans les transitions Activity , les mutations Act réhydratation Activity .
Stop détaché	✓	99 %	Le frontend ne le rend que pour <code>Running/source/closed</code> , avec <code>SquareStop</code> , <code>small</code> , <code>danger</code> , identité

Domaine fonctionnel	État	Complétude estimée	Vérification
Propagation synchrone de stopRequested	✓	99 %	par sourceId et désactivation immédiate dans l'Activité frontend . Le latch local est même robuste que la seule prop disableAfterCli refus RPC peut réactiver le bouton. Le runtime est muté avant l'abort dans le run Activity , l'endpoint Stop source et le service S détaché .
Son exact par run et tentative	✓	98 %	La déduplication terminale, les attempts mono cleanup exact sont implémentés dans le gestion son .
Suppression durable sous claim	✓	97 %	L'endpoint de suppression ferme le panneau, s puis attend les sources normales, supprime sur avant éviction et réconcilie Activity en cas d'é dans l'endpoint Delete .
Retry EasyAgents destructif sous claim	✓	97 %	Le claim est acquis juste avant la fermeture, c travers éviction/delete/restart et libéré en finally dans le service Retry .
Préservation du contrat EasyAgents	⚠	92 %	sourceId=turnId, replay replace, lifecycle observer ownership et last-viewer kill sont pré bug de lease lors d'un simple reveal peut néan empêcher le last-viewer kill dans un cas réel.
Coordination panneau fermeture/réouverture	⚠	86 %	La sérialisation par locator, les claims et clean corrects dans le coordinateur de panneau . La conservation du descripteur réel lors d'un reve incorrecte.
Compositeur normalPrompt après armement du replay	⚠	82 %	Il devient interactif pendant liveSessionSt mais est verrouillé à nouveau sur liveSessionReplayFailed et pendant même si la baseline et le replay avaient déjà é Il exige terminal,
Handoff direct normalPrompt	✓	98 %	outcome completed, streamInstanceId e et finalizedStreamInstanceId, sans sec ni nouvelle hydratation du compositeur.
Hydratation preserveLocal réellement opaque	✓	100 %	l'application d'hydratation ne lit ni le draft, ni attachments, ni nextTurnId dans cette bran La barrière { turnId, streamInstanceId } efface uniquement s exact et démasque sans clear sur finalisation e dans le composant Prompt .
Fallback ack et finalisation sans ack	✓	100 %	La structure de continuation et les cibles think correctes. Le flag de séparateur n'est toutefois explicitement remis à zéro à chaque changeme de streamInstanceId.
Continuation et thinking	⚠	95 %	Les documents demandés sont mis à jour et la couverture comporte les courses source/panne V1, Slimbk body, Open, replay, fallback, cont preserveLocal, Delete et Retry. Deux intégrati précises manquent cependant pour détecter les dessous.
Programming docs et tests spécialisés	✓	97 %	

Conclusion Q2 : le cœur backend, les identités exactes, la durabilité et les files frontend sont complets ; les deux zones non conformes sont la propriété du descripteur de panneau lors d'un reveal et la persistance de l'interactivité locale après armement du replay.

3) Y a-t-il des bugs ?

Réponse

✓ **Oui : deux bugs matériels confirmés et un écart d'invariant mineur.**

✗ **Bug 1 — Le lease réellement utilisé par un panneau peut être remplacé par le lease d'une simple demande de reveal**

Sévérité : élevée — confiance : 99 %.

Le coordinateur remplace son descripteur par celui de **chaque nouvelle demande d'ouverture** avant d'appeler le port :

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

Copy

```
if (descriptor) {
    this.descriptorByLocator.set(locatorKey, descriptor);
}
this.detachedLocatorKeys.delete(locatorKey);
try {
    args.port.open(query);
}
```

Mais le low-level opener n'applique pas la nouvelle query lorsqu'un panneau existe déjà : il se contente de le révéler.

[AICode/src/extension/src/extension/panels/core/openPanel.util.ts](#)

Copy

```
const existing = openedPanelsRegistry.get(key);
if (existing) {
    existing.reveal(VsCode.enums.ViewColumn.One);
    // ...
    return;
}
```

Par ailleurs, le frontend EasyAgents génère volontairement un lease frais à chaque clic, y compris au deuxième clic :

[AICode/src/frontend/src/app/easy-agents/easy-agents-batch/core/_test_/EasyAgentsBatch.liveWatch.test.tsx](#)

Copy

```
expect(firstLease).toEqual(expect.any(String));
expect(secondLease).toEqual(expect.any(String));
expect(secondLease).not.toBe(firstLease);
```

Scénario exact

1. Premier clic `Watch live` : le panneau est créé avec le lease **L1**.
2. Le frontend attache réellement **L1** au record.
3. Deuxième clic pendant que le panneau existe : une query contenant **L2** est produite.
4. Le coordinateur remplace son descripteur par **L2**.
5. Le low-level opener révèle simplement le panneau existant ; le frontend continue d'utiliser **L1**.
6. À la fermeture, le coordinateur tente de libérer **L2**.
7. La libération est un no-op, car **L2** n'a jamais été acquis.
8. **L1 reste attaché.**
9. Pour EasyAgents, la disparition du dernier viewer n'est pas détectée et le `kill(...)` de politique produit peut ne pas être déclenché.
Cela viole directement :
 - l'identité exacte du viewer ;

- la capture du descripteur initial ;
- la politique `last viewer → kill EasyAgents` ;
- la garantie qu'une fermeture réelle libère le viewer réel.

Le test d'ouverture existant vérifie seulement qu'un panneau déjà ouvert est révélé, et le test frontend vérifie seulement que les leases successifs sont différents : **aucun test ne joint actuellement ces deux faits.**



Bug 2 — Le compositeur `normalPrompt` est reverrouillé après un échec de replay pourtant déjà armé Sévérité : moyenne à élevée — confiance : 100 %.

L'adaptateur réel considère le compositeur interactif uniquement pendant `liveSessionStreaming` :

[AI/Code/src/frontend/src/app/prompt/liveSession/adapters/NormalPromptLiveSession.adapter.ts](#)

Copy

```
const replayBaselineInstalled = args.mode === 'liveSessionStreaming';
// ...
composerInteractive: replayBaselineInstalled,
```

Or, après une erreur transport ou une fin sans terminal :

- le hook passe à `liveSessionReplayFailed` ;
- la baseline et le compositeur local peuvent déjà avoir été correctement installés ;
- `preserveLocal` doit conserver ce compositeur pendant le Retry ;
- mais l'adaptateur retourne alors `composerInteractive: false`.

Le test bas niveau de [PromptInput.batchLiveViewerMode.test.tsx](#) exprime le comportement attendu en construisant manuellement un modèle `failed` à partir du modèle `streaming` :

[AI/Code/src/frontend/src/app/prompt/PromptInput/ __test__ /PromptInput.batchLiveViewerMode.test.tsx](#)

Copy

```
const NORMAL_PROMPT_REPLAY_FAILED: PromptLiveSessionViewModel = {
  ...NORMAL_PROMPT_STREAMING,
  transportStopAvailable: false,
  // ...
};
```

Ce test ne traverse cependant pas l'adaptateur réel. Le test de l'adaptateur `normalPrompt` ne vérifie pas `composerInteractive` pour le mode `failed`.

Le problème ne peut pas être corrigé en rendant simplement **tous** les `liveSessionReplayFailed` interactifs :

- un échec du premier Open avant hydratation doit rester verrouillé ;
- un échec après hydratation et acceptation de l'observer doit rester interactif ;
- un Retry après armement doit également conserver l'interactivité pendant son nouvel état `opening`.

Il manque donc un état explicite du type `replayComposerArmed`, indépendant du mode grossier.



Écart 3 — Le flag de séparateur `thinking` n'est pas réellement réinitialisé par `streamInstanceId` Sévérité : faible / latent — confiance : 95 %.

Dans [le lifecycle thinking](#), un changement de `streamInstanceId` remet à zéro :

- le texte `thinking` accumulé ;
- la longueur précédente.

Mais il ne remet pas explicitement `pendingHrRef.current` à `false`.

La recherche dans le sous-système montre que le reset explicite du flag se fait uniquement dans le reset complet de [usePromptRunController.hook.ts](#), pas à chaque tentative acceptée.

Le comportement visible actuel masque en grande partie le problème : lorsque le nouveau stream reçoit son premier paquet `thinking`, le flag `stale` est consommé sans insérer de séparateur puisque le texte local est encore vide. Néanmoins :

- l'invariant documenté est faux ;
- un état de tentative A survit réellement dans la tentative B ;
- une évolution future du traitement des premiers paquets pourrait rendre le défaut visible.

Conclusion Q3 : deux bugs doivent être corrigés avant validation, et le reset `thinking` mérite un petit durcissement pour rendre le code conforme à son propre invariant.

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

Avis sur le code source

Critère	Estimation	Avis
Architecture	93 %	✓ Très forte séparation registre/source/replay/coordinator/adaptateurs/runtime Activity.
Typage et contrats	96 %	✓ Identités strictes, unions fermées, handles opaques et branches Zod bien construites.
Gestion des courses	94 %	✓ Excellent travail sur les époques, générations destructives, IDs de tentative et revalidations après I/O.
Tests et commentaires	97 %	✓ Couverture remarquable et commentaires de maintenance très utiles.
Maintenabilité	87 %	⚠ Le nombre d'automates corrélés reste élevé et certains états historiques sont réduits à des modes trop grossiers.
Validation finale actuelle	94 %	⚠ Deux intégrations manquantes empêchent encore une validation complète.

Les meilleurs choix sont :

- la distinction `turnId`, `sourceId`, `streamInstanceId` et identité Open ;
- la réservation synchronisée avant tout `await` ;
- la séparation des époques de run et générations destructives ;
- la persistance de chaque erreur avant `append` ;
- la finalisation seulement après les files ;
- le claim destructif opaque ;
- la défense en profondeur extension/frontend des queries ;
- `preserveLocal` qui ne lit réellement pas les champs ignorés ;
- le terminal unique lorsque le transport échoue après les données terminales.

La principale faiblesse architecturale est la même dans les deux bugs :

un état historique important est représenté par un état courant trop grossier.

- Le coordinateur mémorise « le dernier descripteur demandé » au lieu du **descripteur réellement attaché à l'instance de panneau**.
- L'adaptateur mémorise seulement « le mode courant » au lieu de savoir si le **compositeur avait déjà franchi la barrière baseline + replay armé**.

Avis sur la spécification

La spécification est excellente sur les invariants et les courses : **environ 94 % de qualité architecturale**.

Elle a néanmoins une faiblesse documentaire : sa « liste exacte » de fichiers n'était pas complètement exacte. Pour implémenter le protocole de header non-200 gardant le body ouvert, le code a légitimement dû étendre :

- [le contrat de résultat Slimbk](#)
- [le contrat d'endpoint Slimbk](#)
- [le runtime Slimbk](#)
- [le générateur de client Slimbk](#)

Ces changements supplémentaires sont **justifiés et correctement testés**. Je ne les considère pas comme une régression ni comme du hors-périmètre nuisible. Les sorties `*.generated.*` correspondent à la régénération attendue, sans contournement de typage.

La consolidation des nettoyages dans `PromptThreadCleanup.manager.ts` est moins segmentée que le texte de la spécification, mais le coordinateur impose correctement l'autorité, le claim et la revalidation. Je considère cette simplification **KIS et acceptable**, pas comme une correction nécessaire.

Conclusion Q4 : le code est d'un très bon niveau production et la spécification est exceptionnellement rigoureuse, mais deux erreurs de modélisation d'état montrent que les tests doivent davantage croiser les frontières entre modules, pas seulement valider chaque module isolément.

5) Résumé du rapport

TL-DR

Question	Résultat	Estimation	Décision
Spécification entièrement implémentée ?	Non, deux comportements matériels restent incorrects.	96 %	⚠
Architecture	Réervations, registres, replay, finalisation et	98 %	✓

Question	Résultat	Estimation	Décision
backend <code>liveSession</code>	claims correctement réalisés.		
Transport et protocoles d'erreur	V1, erreurs body Slimbk, durabilité terminale et terminal unique sont cohérents.	98 %	✓
Démarrage et files frontend	Transactionnel, tentative exacte, Queue A/B et commit final correctement gardés.	98 %	✓
Activité, Stop détaché et son	Fonctionnement conforme hors bug de lease au reveal.	97 %	✓ / ⚠
Fermeture/réouverture du panneau	Le descripteur courant peut ne plus correspondre au panneau réel.	86 %	✗
EasyAgents last-viewer	Correct normalement, mais cassable par un second <code>watch live</code> sur le panneau existant.	92 %	✗
Compositeur normalPrompt	Correct pendant le stream, incorrect après échec/retry déjà armé.	82 %	✗
Fallback persist sans ack	Conforme : démasquage sans effacement et focus restaurable.	100 %	✓
Thinking/continuation	Visuellement cohérent, avec un reset de flag interne incomplet.	95 %	⚠
Qualité du code	Très élevée, strictement typée et fortement testée.	93 %	✓
Risque résiduel actuel	Principalement reveal/lease et état du compositeur.	≈ 30 %	⚠
Risque estimé après corrections	Reste lié au vaste couplage asynchrone de la tranche.	≈ 18 %	✓
Conclusion Q5 : la tranche est techniquement très avancée et presque terminée, mais les deux bugs d'intégration empêchent de considérer le comportement produit comme totalement conforme.			

6) Conclusion en une phrase

L'implémentation est globalement excellente et presque complète, mais elle ne doit pas être validée définitivement avant d'avoir conservé le lease réellement attaché lors d'un reveal et maintenu l'interactivité du compositeur après qu'un replay normalPrompt a déjà été armé.

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

Lot A — Corriger la propriété du descripteur et du lease lors d'un reveal 🔒

Priorité : P1 élevée.

- Faire retourner au port d'ouverture une disposition stricte `created | revealed`.
- Conserver le descripteur précédent lorsque le low-level ne fait que révéler un panneau.
- Ne jamais associer au panneau un lease qui n'a pas été transmis à son frontend.
- Ajouter un test d'intégration :
 - ouverture EasyAgents avec L1 ;
 - deuxième demande avec L2 ;
 - reveal du panneau existant ;
 - dispose ;
 - vérification que L1, et non L2, est libéré ;
 - vérification du chemin last-viewer `kill(...)`.
- Fichiers centraux :
 - [coordinateur de panneau](#)
 - [low-level opener](#)
 - [tests du coordinateur](#)
 - [tests d'ouverture de panneau](#)

Lot B — Conserver l'interactivité locale après armement du replay ⚡

Priorité : P1.

- Ajouter un état explicite `replayComposerArmed`, indépendant de `PromptRuntimeMode`.
- Le positionner uniquement après :

- hydratation live réussie ;
- acceptation du stream observer ;
- installation du scaffold.
- Le conserver pendant :
- `liveSessionReplayFailed` ;
- un `Retry preserveLocal` ;
- un `Open Retry` en cours.
- Le laisser faux lorsqu'un premier `Open` échoue avant installation de la baseline.
- Faire dépendre le modèle `normalPrompt` de cet état plutôt que du seul mode.
- Ajouter un test traversant réellement `hook` → adaptateur → `PromptInput`, sans modèle manuel fabriqué.
- Fichiers centraux :
- [types live frontend](#)
- [hook live frontend](#)
- [adaptateur normalPrompt](#)
- [wiring Prompt](#)
- [tests normalPrompt](#)

Lot C — Aligner strictement le séparateur thinking sur `streamInstanceId` 🧠

Priorité : P2 faible.

- Remettre `pendingHrRef.current = false` dans la branche qui détecte un nouveau `streamInstanceId`.
- Ajouter un test où la tentative A se termine avec un séparateur armé, puis la tentative B démarre avec une identité différente.
- Vérifier qu'aucun état interne de A ne survit dans B.
- Fichiers centraux :
- [lifecycle thinking](#)
- [tests thinking continuation](#)

Lot D — Toutes les corrections 🚀

Inclure les lots A, B et C avec leurs tests d'intégration, sans modifier les algorithmes `Queue A`, `Queue B`, `locks` ou `sanitizers`.

Conclusion Q7 : les lots A et B sont nécessaires pour valider la spécification ; le lot C est un durcissement léger mais recommandé afin que le code respecte exactement son invariant documenté.



Coder la correction des leases au reveal



Coder la correction du compositeur live



Coder le reset thinking par tentative



Coder toutes les corrections

</ASSISTANT>

<USER>

[Coder toutes les corrections]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>



reading project: [liveSession.programming-doc-readme.md](#), [thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md](#),



[PromptLiveSessionPanelCoordinator.service.ts](#),



[openPanel.util.ts](#),



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



[openPanel.util.test.ts](#),



[PromptLiveSession.types.ts](#),



[usePromptLiveSession.hook.ts](#),



[NormalPromptLiveSession.adapter.ts](#),



[EasyAgentsBatchLiveSession.adapter.ts](#), [Prompt.tsx](#), [PromptInput.tsx](#), [usePromptThinkingLifecycle.hook.ts](#),

[usePromptLiveSession.normalPrompt.test.tsx](#),
[NormalPromptLiveSession.adapter.test.ts](#),
[usePromptThinkingLifecycle.continuation.test.tsx](#), [r "buildViewModel"](#), [r "PromptLiveSessionApi"](#), [e "Appels buildViewModel"](#), [e "Ports panneau Prompt"](#), [e "Open port tests"](#), [e "Ports void coordinateur"](#), [e "Ports throw coordinateur"](#),
[EasyAgentsBatchLiveSession.adapter.test.ts](#), [PromptInput.batchLiveViewerMode.test.tsx](#), [e "Adaptateurs mockés"](#), [e "Mocks coordinateur panneau"](#), [e "Assertions API live exactes"](#), [e "Args modèle tests"](#), [e "Port open coordinateur"](#)

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

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

Copy

Review

Accept

PatchFull file

```
-  
- interface PromptPanelOpenPort {  
-   open(query: { name: string; value: string }[]): void;  
- }  
-  
+  
+ /** Result reported by the low-level Prompt opener for the exact physical panel  
+ instance. */  
+ export type PromptPanelOpenDisposition = 'created' | 'revealed';  
+  
+ /** Port used by the coordinator without importing the low-level panel implementation.  
+ */  
+ interface PromptPanelOpenPort {  
+   open(  
+     query: { name: string; value: string }[],  
+   ): PromptPanelOpenDisposition;  
+ }  
+  
-----  
-       const wasDetached = this.detachedLocatorKeys.has(locatorKey);  
-  
+       const wasDetached = this.detachedLocatorKeys.has(locatorKey);  
+  
+       // Install the requested descriptor provisionally before opening. A newly  
+ created panel may  
+       // dispose synchronously inside the low-level port and must already expose  
+ its exact identity.  
+  
-----  
-       this.descriptorByLocator.set(locatorKey, descriptor);  
-  
+       this.descriptorByLocator.set(locatorKey, descriptor);  
+     } else {  
+       this.descriptorByLocator.delete(locatorKey);  
+  
-----  
-       try {  
-         // Install the descriptor before opening so even a synchronously  
disposed low-level panel  
-         // observes the exact live identity. A failed open restores the  
previous coordinator state.  
-         args.port.open(query);  
-  
+       try {  
+         const disposition = args.port.open(query);  
+         if (disposition === 'revealed') {
```

```

+                                // Reveal does not reload the webview query. Restore the descriptor
+already bound to
+                                // the existing panel so a fresh, unused request lease can never
+replace its real lease.
+                                this.restoreDescriptorSnapshot({
+                                    locatorKey,
+                                    hadDescriptor: hadPreviousDescriptor,
+                                    descriptor: previousDescriptor,
+                                });
+                                }
+
-                                } catch (error) {
-                                    if (hadPreviousDescriptor && previousDescriptor) {
-                                        this.descriptorByLocator.set(
-                                            locatorKey,
-                                            previousDescriptor,
-                                        );
-                                    } else {
-                                        this.descriptorByLocator.delete(locatorKey);
-                                    }
+                                } catch (error) {
+                                    // A failed physical open did not bind the provisional descriptor to
+any panel.
+                                    this.restoreDescriptorSnapshot({
+                                        locatorKey,
+                                        hadDescriptor: hadPreviousDescriptor,
+                                        descriptor: previousDescriptor,
+                                    });
+
-                                }
-                            }
+                            }
+                        }
+
+                        /** Restore the exact descriptor that belonged to the physical panel before one
+open attempt. */
+                        private restoreDescriptorSnapshot(args: {
+                            locatorKey: string;
+                            hadDescriptor: boolean;
+                            descriptor: LiveSessionPanelDescriptor | undefined;
+                        }): void {
+                            // Map presence is part of the snapshot because an absent descriptor means a
+classic Prompt panel.
+                            if (args.hadDescriptor && args.descriptor) {
+                                this.descriptorByLocator.set(args.locatorKey, args.descriptor);
+                                return;
+                            }
+                            this.descriptorByLocator.delete(args.locatorKey);
+                        }
+

```

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

Copy

Review

Accept

  patching file:  [openPanel.util.ts](#)

[AI/Code/src/extension/src/extension/panels/core/openPanel.util.ts](#)

Error

([what is it?](#))

  patching file:  [PromptLiveSession.types.ts](#)

[AICode/src/frontend/src/app/prompt/liveSession/core/PromptLiveSession.types.ts](#)

Copy

Review

Accept

PatchFull file

```
-         mode: PromptRuntimeMode;
-
+         mode: PromptRuntimeMode;
+         /** True after baseline hydration and one exact observer stream were accepted.
+ */
+         replayComposerArmed: boolean;
+
```

[AICode/src/frontend/src/app/prompt/liveSession/core/PromptLiveSession.types.ts](#)

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

```
-         retryHandoff: () => void;
-
+         retryHandoff: () => void;
+         /** Whether baseline hydration and one exact replay observer have armed the local
+ composer. */
+         replayComposerArmed: boolean;
+
```

```
-         useState(hasInitialSourceStopAuthority);
-
+         useState(hasInitialSourceStopAuthority);
+         const [replayComposerArmed, setReplayComposerArmed] = useState(false);
+
```

```
-                                     const latest = paramsRef.current;
-
+                                     const latest = paramsRef.current;
+                                     // Terminal hydration completes the live bootstrap atomically
with composer ownership.
+                                     setReplayComposerArmed(false);
+
```

```
-                                     startResult.streamInstanceId;
-
+                                     startResult.streamInstanceId;
+                                     // Hydration and the accepted observer scaffold are now both
installed. This latch
+                                     // intentionally survives replay failure and preserve-local Retry
opening states.
+                                     setReplayComposerArmed(true);
+
```

```
-         handoffStartedRef.current = true;
-
+         handoffStartedRef.current = true;
```

```

+      // Publish normal-mode ownership and retire the replay latch in the same
handoff transition.
+      setReplayComposerArmed(false);
+
-      if (params.promptRuntimeMode !== 'normal') return;
-
+      if (params.promptRuntimeMode !== 'normal') return;
+      // Once handoff completed, normal Prompt owns the composer and no replay-
specific latch remains.
+      setReplayComposerArmed(false);
+
-      () => ({
-        retryHandoff,
-
+      () => ({
+        retryHandoff,
+        replayComposerArmed,
+
-      [
-        ownsSourceStopRoute,
-
+      [
+        ownsSourceStopRoute,
+        replayComposerArmed,
+

```

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

Copy

Review

Accept

  **patching file:**  [NormalPromptLiveSession.adapter.ts](#)

[AI Code/src/frontend/src/app/prompt/liveSession/adapters/NormalPromptLiveSession.adapter.ts](#)

Copy

Review

Accept

PatchFull file

```

-      buildViewModel: ({
-        mode,
-
+      buildViewModel: ({
+        mode,
+        replayComposerArmed,
+
-      mode,
-
+      mode,
+      replayComposerArmed,
+
-      mode: PromptRuntimeMode;
-
+      mode: PromptRuntimeMode;
+      replayComposerArmed: boolean;
+
-      args.mode === 'liveSessionReplayFailed';
-      const replayBaselineInstalled = args.mode === 'liveSessionStreaming';
-
+      args.mode === 'liveSessionReplayFailed';

```

```

+
-         showComposer: true,
-         // Local draft and attachment edits are allowed only after Open hydration and
observer arming completed.
-         composerInteractive: replayBaselineInstalled,
-
+         showComposer: true,
+         // Runtime mode may move through failure and Retry after arming. The dedicated
latch preserves
+         // local draft and attachment editing without unlocking Send before direct
handoff completes.
+         composerInteractive: args.replayComposerArmed,
+
AI Code/src/frontend/src/app/prompt/liveSession/adapters/NormalPromptLiveSession.adapter.ts

```

Copy

Review

Accept

  patching file:  [Prompt.tsx](#)

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

Copy

Review

Accept

PatchFull file

```

-
-     /** Optional handoff failure details shown while the Prompt stays read-only. */
-
+
+     /** Optional replay/handoff failure details shown under the adapter-owned
interaction policy. */
+
-
-         mode: promptRuntimeMode,
-
+         mode: promptRuntimeMode,
+         // This early projection reads only Send/history/cleanup policy, which
is independent
+         // from the later replay-composer latch owned by usePromptLiveSession.
+         replayComposerArmed: false,
+
-
-     const {
-         retryHandoff,
-
+     const {
+         retryHandoff,
+         replayComposerArmed,
+
-
-     return liveSessionReadyBootstrap.presentationAdapter.buildViewModel({
-         mode: promptRuntimeMode,
-
+     return liveSessionReadyBootstrap.presentationAdapter.buildViewModel({
+         mode: promptRuntimeMode,
+         replayComposerArmed,
+
-
-     liveSessionReadyBootstrap,
-     promptRuntimeMode,
-
+     liveSessionReadyBootstrap,
+     promptRuntimeMode,

```

```
+      replayComposerArmed,
+
-      useEffect(() => {
-        // LiveSession panels are read-only observers. Invalid live descriptors must
also perform zero
-        // backend work before their explanatory dialog closes the panel.
-
+      useEffect(() => {
+        // Live-session adapters explicitly decide when the local composer is writable.
Invalid live
+        // descriptors must still perform zero backend work before their explanatory
dialog closes the panel.
+
AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx
```

Copy
Review
Accept

  *patching file:*  [usePromptThinkingLifecycle.hook.ts](#)
[AICode/src/frontend/src/app/prompt/Prompt/usePromptThinkingLifecycle.hook.ts](#)

Copy
Review
Accept

PatchFull file

```
-      streamInstanceIdRef.current = streamInstanceId;
-      displayTextRef.current = '';
-      prevLenRef.current = 0;
-
+      streamInstanceIdRef.current = streamInstanceId;
+      displayTextRef.current = '';
+      prevLenRef.current = 0;
+      // A separator belongs only to the exact accepted stream that armed it.
Clearing it at the
+      // identity boundary prevents a TOOL_USE packet from attempt A leaking into
attempt B.
+      pendingHrRef.current = false;
+
AICode/src/frontend/src/app/prompt/Prompt/usePromptThinkingLifecycle.hook.ts
```

Copy
Review
Accept

  *reading project:*  [openPanel.util.ts](#) [L20-L55],  [openPanel.util.ts](#) [L345-L470],  [openPanel.util.ts](#) [L585-L680],  [openPanel.util.ts](#) [L315-L345]

  *patching file:*  [openPanel.util.ts](#)
[AICode/src/extension/src/extension/panels/core/openPanel.util.ts](#)

Copy
Review
Accept

PatchFull file

```
- import type { LiveSessionPostForwardingFinalizationResult } from
'../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
- import { PromptLiveSessionPanelCoordinatorService } from
'../../prompt/liveSession/PromptLiveSessionPanelCoordinator.service.js';
-
+ import type { LiveSessionPostForwardingFinalizationResult } from
'../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
+ import {
```

```

+     PromptLiveSessionPanelCoordinatorService,
+     type PromptPanelOpenDisposition,
+ } from '../prompt/liveSession/PromptLiveSessionPanelCoordinator.service.js';
+
-     queryString?: { name: string; value: string }[],
- ): void {
-
+     queryString?: { name: string; value: string }[],
+ ): PromptPanelOpenDisposition {
+
-     }
-     return;
-
+     }
+     return 'revealed';
+
-     void clearChatSelection();
-     }
- });
-
+     void clearChatSelection();
+     }
+ });
+
+ // The caller may commit the query descriptor because this exact panel loaded it.
+ return 'created';
+

```

[AI Code/src/extension/src/extension/panels/core/openPanel.util.ts](#)

Copy

Review

Accept

  scanning git:  *"Ports de test à typer"*,  *"Mocks ouverture panneau"*

  reading project:  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L1-L35],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L175-L445],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L510-L550],  [openPanel.util.test.ts](#) [L1-L55],  [openPanel.util.test.ts](#) [L115-L175],  [openPanel.util.test.ts](#) [L210-L255],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L120-L180]

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

[AI Code/src/extension/src/extension/panels/prompt/liveSession/ test /](#)

[PromptLiveSessionPanelCoordinator.service.test.ts](#)

Error

([what is it?](#))

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

[AI Code/src/extension/src/extension/panels/prompt/liveSession/ test /](#)

[PromptLiveSessionPanelCoordinator.service.test.ts](#)

Copy

Review

Accept

PatchFull file

```

- import type { LiveSessionRunRef } from
- '../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
-
+ import type { LiveSessionRunRef } from
+ '../../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';

```

```

+ import type { PromptPanelOpenDisposition } from
+ '../PromptLiveSessionPanelCoordinator.service.js';
+
-
- /** Let one detached coordinator queue and its rejection handler settle. */
-
+ /** Build a typed low-level panel port mock with one deterministic create/reveal
+ result. */
+ function createPanelOpenMock(
+   disposition: PromptPanelOpenDisposition = 'created',
+ ): jest.MockedFunction<
+   (
+     query: { name: string; value: string }[],
+   ) => PromptPanelOpenDisposition
+ > {
+   return jest.fn<
+     (
+       query: { name: string; value: string }[],
+     ) => PromptPanelOpenDisposition
+   >(() => disposition);
+ }
+
+ /** Let one detached coordinator queue and its rejection handler settle. */
+
-
- it('injects normalPrompt only for a classic open with an active source', async ()
=> {
-   activeSnapshots = [reservation()];
-   const open =
-     jest.fn<(query: { name: string; value: string }[]) => void>();
-
+   it('injects normalPrompt only for a classic open with an active source', async ()
=> {
+     activeSnapshots = [reservation()];
+     const open = createPanelOpenMock();
+
-
-   ];
-   const open =
-     jest.fn<(query: { name: string; value: string }[]) => void>();
-
+   ];
+   const open = createPanelOpenMock();
+
-
-   activeSnapshots = [reservation(RUN, 'observer')];
-   const open =
-     jest.fn<(query: { name: string; value: string }[]) => void>();
-
+   activeSnapshots = [reservation(RUN, 'observer')];
+   const open = createPanelOpenMock();
+
-
-   it('rejects an explicit invalid live query before calling the low-level port',
async () => {
-     const open =
-       jest.fn<(query: { name: string; value: string }[]) => void>();
-
+     it('rejects an explicit invalid live query before calling the low-level port',
async () => {
+       const open = createPanelOpenMock();
+
-
-       activeSnapshots = [reservation()];

```

```

-     const open =
-         jest.fn<(query: { name: string; value: string }[]) => void>();
-
+     activeSnapshots = [reservation()];
+     const open = createPanelOpenMock();
+


---


-
-     it('rolls back the descriptor when the low-level opener fails', async () => {
-
+
+     it('keeps the descriptor bound to the existing panel when a fresh EasyAgents lease
only reveals it', async () => {
+         activeSnapshots = [reservation(EASY_AGENTS_RUN, 'observer')];
+         const coordinator = createCoordinator();
+         const revealOnlyQuery = EASY_AGENTS_QUERY.map((entry) =>
+             entry.name === 'liveSessionViewerLeaseId'
+                 ? { ...entry, value: 'unused-reveal-lease' }
+                 : entry,
+         );
+
+         await coordinator.openPromptPanel({
+             locator: LOCATOR,
+             query: EASY_AGENTS_QUERY,
+             port: { open: createPanelOpenMock('created') },
+         });
+         await coordinator.openPromptPanel({
+             locator: LOCATOR,
+             query: revealOnlyQuery,
+             port: { open: createPanelOpenMock('revealed') },
+         });
+         await coordinator.onPanelDisposed(LOCATOR);
+
+         // The revealed webview never loaded the second query, so only its original
acquired lease is releasable.
+         expect(releaseEasyAgentsViewerMock).toHaveBeenCalledTimes(1);
+         expect(releaseEasyAgentsViewerMock).toHaveBeenCalledWith({
+             runRef: EASY_AGENTS_RUN,
+             leaseId: 'lease',
+         });
+     });
+
+     it('rolls back the descriptor when the low-level opener fails', async () => {
+


---


-         port: { open: () => {} },
-     });
-
-     await coordinator.onPanelDisposed(LOCATOR);
-
+         port: { open: () => 'created' },
+     });
+
+     await coordinator.onPanelDisposed(LOCATOR);
+


---


-         query: EASY_AGENTS_QUERY,
-         port: { open: () => {} },
-
+         query: EASY_AGENTS_QUERY,
+         port: { open: () => 'created' },
+


---


-         port: { open: () => {} },
-     });

```

```
-      await coordinator.onPanelDisposed(LOCATOR);
-
-      activeSnapshots = [];
-      cleanupOnPromptDisposeMock
-
+      port: { open: () => 'created' },
+    });
+    await coordinator.onPanelDisposed(LOCATOR);
+
+    activeSnapshots = [];
+    cleanupOnPromptDisposeMock
+
```

```
-      port: { open: () => {} },
-    });
-    await coordinator.onPanelDisposed(LOCATOR);
-
-    activeSnapshots = [];
-
+    port: { open: () => 'created' },
+  });
+  await coordinator.onPanelDisposed(LOCATOR);
+
+  activeSnapshots = [];
+
```

```
-      query: BASE_QUERY,
-      port: { open: () => {} },
-
+      query: BASE_QUERY,
+      port: { open: () => 'created' },
+
```

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

Copy
Review
Accept



patching file: [openPanel.util.test.ts](#)

[AICode/src/extension/src/extension/panels/core/_test_/openPanel.util.test.ts](#)

Copy
Review
Accept

PatchFull file

```
- } from '../../../../../sharedlib/types/easyAgentsTypes/EasyAgentItemTabId.type.js';
-
+ } from '../../../../../sharedlib/types/easyAgentsTypes/EasyAgentItemTabId.type.js';
+ import type { PromptPanelOpenDisposition } from
+ '../prompt/liveSession/PromptLiveSessionPanelCoordinator.service.js';
+
```

```
-      (args: { batchParentLocator: ChatSessionLocator }) => Promise<void>
-    >(async () => {}));
-
+      (args: { batchParentLocator: ChatSessionLocator }) => Promise<void>
+    >(async () => {}));
+ const promptOpenDispositions: PromptPanelOpenDisposition[] = [];
+
```

```
-      query: { name: string; value: string }[];
-      port: { open: (query: { name: string; value: string }[]) => void };
-
```

```

+         query: { name: string; value: string }[];
+         port: {
+             open: (
+                 query: { name: string; value: string }[],
+             ) => PromptPanelOpenDisposition;
+         };
+
- >(async (args) => {
-     args.port.open(args.query);
-
+ >(async (args) => {
+     promptOpenDispositions.push(args.port.open(args.query));
+
-
-     detachHistoryViewMock.mockClear();
-
+     detachHistoryViewMock.mockClear();
+     promptOpenDispositions.length = 0;
+
-
-     coordinatorOpenMock.mockImplementation(async (args) => {
-         args.port.open(args.query);
-
+     coordinatorOpenMock.mockImplementation(async (args) => {
+         promptOpenDispositions.push(args.port.open(args.query));
+
-
-     expect(coordinatorOpenMock).toHaveBeenCalledTimes(2);
-
+     expect(coordinatorOpenMock).toHaveBeenCalledTimes(2);
+     expect(promptOpenDispositions).toEqual(['created', 'revealed']);
+

```

[AI Code/src/extension/src/extension/panels/core/_test_/openPanel.util.test.ts](#)

Copy

Review

Accept

  **patching file:**  [NormalPromptLiveSession.adapter.test.ts](#)

[AI Code/src/frontend/src/app/prompt/liveSession/adapters/_test_/NormalPromptLiveSession.adapter.test.ts](#)

Copy

Review

Accept

PatchFull file

```

-         mode: 'liveSessionOpening',
-
+         mode: 'liveSessionOpening',
+         replayComposerArmed: false,
+
-
-         mode: 'liveSessionStreaming',
-
+         mode: 'liveSessionStreaming',
+         replayComposerArmed: true,
+
-
-     });
-     const failedWhileStopping = ready.presentationAdapter.buildViewModel({
-
+     });
+     const retryOpening = ready.presentationAdapter.buildViewModel({
+         mode: 'liveSessionOpening',
+         replayComposerArmed: true,
+

```

```

+         transportStopAvailable: true,
+         stopEnabled: true,
+     });
+     const failedWhileStopping = ready.presentationAdapter.buildViewModel({
+
-         mode: 'liveSessionReplayFailed',
-
+         mode: 'liveSessionReplayFailed',
+         replayComposerArmed: true,
+
-     });
-     expect(failedWhileStopping.banner).toEqual(
-
+     );
+     expect(retryOpening).toEqual(
+         expect.objectContaining({ composerInteractive: true }),
+     );
+     expect(failedWhileStopping).toEqual(
+         expect.objectContaining({ composerInteractive: true }),
+     );
+     expect(failedWhileStopping.banner).toEqual(
+

```

[AI Code/src/frontend/src/app/prompt/liveSession/adapters/ test /NormalPromptLiveSession.adapter.test.ts](#)

Copy

Review

Accept

  patching file:  [EasyAgentsBatchLiveSession.adapter.test.ts](#)

[AI Code/src/frontend/src/app/prompt/liveSession/adapters/ test /EasyAgentsBatchLiveSession.adapter.test.ts](#)

Copy

Review

Accept

PatchFull file

```

-         mode: 'liveSessionStreaming',
-
+         mode: 'liveSessionStreaming',
+         replayComposerArmed: true,
+

```

[AI Code/src/frontend/src/app/prompt/liveSession/adapters/ test /EasyAgentsBatchLiveSession.adapter.test.ts](#)

Copy

Review

Accept

  patching file:  [usePromptThinkingLifecycle.continuation.test.tsx](#)

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

Copy

Review

Accept

PatchFull file

```

-         ).toBe('one');
-
+         ).toBe('one');
+
+         // A TOOL_USE separator belongs to stream A and must be cleared before stream B
+         receives data.
+         result.current.pendingHrRef.current = true;
+         act(() =>
+             rerender({

```

```
+         id: 'b',
+         thinkingTokens: [],
+         tokens: [],
+     })),
+ );
+ expect(result.current.pendingHrRef.current).toBe(false);
+
```

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

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

The typed extension bus attaches an extension-only origin context containing the emitting `HtmlPage` and exact `vscode.Webview`. This identity is never serialized. Source handlers must compare that webview with the currently registered panel before reservation so a queued request from a disposed panel cannot start on a reopened panel sharing the same locator.

The typed extension bus attaches an extension-only origin context containing the emitting `HtmlPage` and exact `vscode.Webview`. This identity is never serialized. Source handlers must compare that webview with the currently registered panel before reservation so a queued request from a disposed panel cannot start on a reopened panel sharing the same locator.

The low-level Prompt open port reports whether it created a physical webview or merely revealed the existing locator panel. The coordinator installs a requested descriptor provisionally so synchronous disposal stays safe, but commits it only when the webview was created with that query. A reveal restores the descriptor already bound to the existing panel; therefore a fresh viewer lease present only in the reveal request is never mistaken for an acquired lease and cannot replace the lease released on disposal.

Frontend replay composer arming

Normal Prompt tracks `replayComposerArmed` independently from the coarse runtime mode. The latch becomes true only after live snapshot hydration and synchronous acceptance of the exact observer stream, when the replay scaffold is already installed. It remains true through replay failure and `preserveLocal` Retry opening so the user can continue editing the next draft and attachments while Send stays locked. Initial Open failures remain unarmed, and direct or snapshot handoff to normal Prompt clears the replay-specific latch.

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

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

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

```

-
+
+ import type {
+   AiStreamOutcome,
+   AiStreamStartResult,
+ } from '../../../../../../messageBus/AiStreamStartArgs.type.js';
+
-
- const refuseLiveObserver = (): AiStreamStartResult => ({ accepted: false });
-
+ const refuseLiveObserver = (): AiStreamStartResult => ({ accepted: false });
+
+ type NormalPromptStreamState = {
+   readonly isStreaming?: boolean;
+   readonly streamInstanceId?: string;
+   readonly finalizedStreamInstanceId?: string;
+   readonly streamOutcome?: AiStreamOutcome;
+   readonly terminalDataSeen?: boolean;
+ };
+
-
-   startLiveObserver: () => AiStreamStartResult = refuseLiveObserver,
-
+   startLiveObserver: () => AiStreamStartResult = refuseLiveObserver,
+   streamState?: NormalPromptStreamState,
+
-
-   startLiveObserver,
-   isStreaming: false,
-   streamInstanceId: undefined,
-   finalizedStreamInstanceId: undefined,
-   streamOutcome: undefined,
-   terminalDataSeen: false,
-
+   startLiveObserver,
+   isStreaming: streamState?.isStreaming ?? false,
+   streamInstanceId: streamState?.streamInstanceId,
+   finalizedStreamInstanceId: streamState?.finalizedStreamInstanceId,
+   streamOutcome: streamState?.streamOutcome,
+   terminalDataSeen: streamState?.terminalDataSeen ?? false,
+
-
-   });
-   return useMemo(() => ({ api, failure, mode }), [api, failure, mode]);
-
+   });
+   const viewModel = useMemo(
+     () =>
+       bootstrap.presentationAdapter.buildViewModel({
+         mode,
+         replayComposerArmed: api.replayComposerArmed,
+         failureDetails: failure,
+         transportStopAvailable: api.sourceStopAvailable,
+         stopEnabled: !api.sourceStopPending,
+       }),
+     [
+       api.replayComposerArmed,
+       api.sourceStopAvailable,
+       api.sourceStopPending,
+       failure,
+       mode,
+     ],
+   );
+   return useMemo(
+     () => ({ api, failure, mode, viewModel }),

```

```

+      [api, failure, mode, viewModel],
+    );
+
-
-    expect(result.current.api.ownsSourceStopRoute).toBe(true);
-    expect(result.current.api.sourceStopAvailable).toBe(false);
-
+    expect(result.current.api.ownsSourceStopRoute).toBe(true);
+    expect(result.current.api.sourceStopAvailable).toBe(false);
+    expect(result.current.api.replayComposerArmed).toBe(false);
+    expect(result.current.viewModel.composerInteractive).toBe(false);
+
-
-    expect(result.current.mode).toBe('liveSessionStreaming');
-
+    expect(result.current.mode).toBe('liveSessionStreaming');
+    expect(result.current.api.replayComposerArmed).toBe(true);
+    expect(result.current.viewModel.composerInteractive).toBe(true);
+
-
-    it('allows manual Retry to consume a direct terminal snapshot without creating a
stream identity', async () => {
-
+
+    it('keeps the composer armed through replay failure and preserve-local Retry
opening', async () => {
+      let resolveRetryOpen: ((value: unknown) => void) | undefined;
+      const startLiveObserver = jest.fn<() => AiStreamStartResult>(() => ({
+        accepted: true,
+        runRef: bootstrap.runRef,
+        streamInstanceId: 'stream-live',
+      }));
+      callLocalApiMock
+        .mockResolvedValueOnce({
+          kind: 'live',
+          session: {
+            meta: {
+              formatVersionNumber: 1,
+              sessionId: locator.sessionId,
+              title: 'Chat',
+              createdAt: 1,
+              updatedAt: 2,
+              firstSendState: 'postSend',
+            },
+            messages: [],
+          },
+          threadId: locator.sessionId,
+          turnId: bootstrap.runRef.turnId,
+          sourceId: bootstrap.runRef.sourceId,
+          replayMode: 'continueAssistant',
+        })
+        .mockImplementationOnce(
+          () =>
+            new Promise((resolve) => {
+              resolveRetryOpen = resolve;
+            }),
+        );
+      const { result, rerender } = renderHook(
+        ({ streamState }: { streamState: NormalPromptStreamState }) =>
+          useNormalPromptLiveSessionHarness(
+            startLiveObserver,
+            streamState,
+          ),
+        { initialProps: { streamState: {} } },

```

```

+     );
+
+     await act(async () => {
+       await Promise.resolve();
+       await Promise.resolve();
+     });
+     expect(result.current.mode).toBe('liveSessionStreaming');
+     expect(result.current.api.replayComposerArmed).toBe(true);
+     expect(result.current.viewModel.composerInteractive).toBe(true);
+
+     await act(async () => {
+       rerender({
+         streamState: {
+           streamInstanceId: 'stream-live',
+           streamOutcome: 'transportError',
+         },
+       });
+       await Promise.resolve();
+     });
+     expect(result.current.mode).toBe('liveSessionReplayFailed');
+     expect(result.current.api.replayComposerArmed).toBe(true);
+     expect(result.current.viewModel.composerInteractive).toBe(true);
+
+     act(() => result.current.api.retryHandoff());
+     expect(result.current.mode).toBe('liveSessionOpening');
+     expect(result.current.api.replayComposerArmed).toBe(true);
+     expect(result.current.viewModel.composerInteractive).toBe(true);
+     expect(callLocalApiMock).toHaveBeenCalledTimes(2);
+
+     await act(async () => {
+       resolveRetryOpen?.({
+         kind: 'terminal',
+         session: {
+           meta: {
+             formatVersionNumber: 1,
+             sessionId: locator.sessionId,
+             title: 'Chat',
+             createdAt: 1,
+             updatedAt: 2,
+             firstSendState: 'postSend',
+           },
+           messages: [],
+         },
+         threadId: locator.sessionId,
+         turnId: bootstrap.runRef.turnId,
+         sourceId: bootstrap.runRef.sourceId,
+       });
+       await Promise.resolve();
+       await Promise.resolve();
+     });
+     expect(result.current.mode).toBe('normal');
+     expect(result.current.api.replayComposerArmed).toBe(false);
+     expect(result.current.viewModel.composerInteractive).toBe(true);
+   });
+
+   it('allows manual Retry to consume a direct terminal snapshot without creating a
stream identity', async () => {

```

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

Copy
Review
Accept

  reading project:  e "[Appels modèle incomplets](#)",  [PromptLiveSessionPanelCoordinator.service.ts](#) [L1-L150],  [PromptLiveSessionPanelCoordinator.service.ts](#) [L330-L410],  [openPanel.util.ts](#) [L25-L45],  [openPanel.util.ts](#) [L320-L455],  [openPanel.util.ts](#) [L610-L685],  [PromptLiveSession.types.ts](#),  [usePromptLiveSession.hook.ts](#) [L45-L110],  [usePromptLiveSession.hook.ts](#) [L220-L315],  [usePromptLiveSession.hook.ts](#) [L485-L575],  [NormalPromptLiveSession.adapter.ts](#),  [Prompt.tsx](#) [L410-L455],  [Prompt.tsx](#) [L655-L765],  [usePromptThinkingLifecycle.hook.ts](#) [L50-L95],  [Prompt.tsx](#) [L465-L510],  [PromptLiveSessionPanelCoordinator.service.ts](#) [L410-L465],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L1-L30],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L135-L385],  [openPanel.util.test.ts](#) [L1-L55],  [openPanel.util.test.ts](#) [L140-L255],  [NormalPromptLiveSession.adapter.test.ts](#),  [EasyAgentsBatchLiveSession.adapter.test.ts](#),  [usePromptLiveSession.normalPrompt.test.tsx](#) [L1-L115],  [usePromptLiveSession.normalPrompt.test.tsx](#) [L290-L500],  [usePromptThinkingLifecycle.continuation.test.tsx](#) [L65-L125],  [liveSession.programming-doc-readme.md](#) [L80-L145],  [usePromptLiveSession.normalPrompt.test.tsx](#) [L500-L560]

  scanning git:  "[Résumé des corrections](#)",  "[Contrôle whitespace](#)"

  scanning recent files

  reading project:  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L385-L570],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L570-L626]

  patching file:  [usePromptLiveSession.normalPrompt.test.tsx](#)
[AICode/src/frontend/src/app/prompt/liveSession/core/ test /usePromptLiveSession.normalPrompt.test.tsx](#)

Grouped
([what is it?](#))

  reading project:  [Prompt.tsx](#) [L285-L315],  [Prompt.tsx](#) [L1035-L1080],  e "[Commentaire read only](#)",  [Prompt.tsx](#) [L190-L210]

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

Grouped
([what is it?](#))

  reading project:  e "[Adaptateurs présentation](#)"

  patching file:  [usePromptLiveSession.normalPrompt.test.tsx](#)
[AICode/src/frontend/src/app/prompt/liveSession/core/ test /usePromptLiveSession.normalPrompt.test.tsx](#)

Grouped
([what is it?](#))

  reading project:  [usePromptLiveSession.hook.ts](#) [L315-L365]

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

Grouped
([what is it?](#))

  reading project:  e "[MockedFunction exemples](#)"

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

Grouped

([what is it?](#))

  reading project:  e "[Latch compositeur replay](#)",  e "[Disposition panneau](#)",  e "[Ancien port void](#)"

Corrections appliquées

Joel, les trois lots ont été implémentés proprement — le panneau ne joue plus au bonneteau avec ses leases, le compositeur garde enfin la mémoire de son armement, et le thinking laisse ses séparateurs dans la bonne tentative. 

Correction	Implémentation	Couverture ajoutée ou renforcée	État
Lease exact lors d'un reveal	Le low-level opener retourne désormais une disposition stricte <code>created</code> <code>revealed</code> . Le coordinateur ne conserve le nouveau descripteur que si la webview a réellement été créée avec cette query. Lors d'un simple reveal, il restaure le descripteur déjà attaché au panneau physique.	Ouverture avec lease L1, nouvelle demande avec L2, simple reveal, puis fermeture : seule L1 est libérée. Le test d'intégration vérifie aussi les dispositions successives <code>created</code> , puis <code>revealed</code> .	 Complet
Compositeur conservé après armement du replay	Ajout d'un latch frontend strict <code>replayComposerArmed</code> . Il devient vrai uniquement après hydratation et acceptation synchrone du stream observer. Il survit aux états d'échec et au Retry <code>preserveLocal</code> , puis est retiré atomiquement lors du handoff vers le mode normal. Send reste verrouillé.	Les tests couvrent l'échec avant armement, l'armement après acceptation, l'erreur transport, le Retry en cours avec compositeur interactif, puis le terminal direct qui rend la propriété au Prompt normal.	 Complet
Isolation du séparateur thinking	Le séparateur pending est maintenant remis à zéro dès que le <code>streamInstanceId</code> change, avant même l'arrivée du premier thinking token de la nouvelle tentative.	Le test arme le séparateur dans la tentative A, démarre la tentative B sans token, vérifie le reset immédiat, puis vérifie la nouvelle accumulation. Le document décrit	 Complet
Documentation de maintenance	Les invariants <code>created/revealed</code> , lease réellement attachée et <code>replayComposerArmed</code> ont été documentés.	explicitement la durée de vie du latch et le comportement des reveals sans rechargement de query.	 Complet
Contraintes de conservation	Aucun changement dans les algorithmes Queue A, Queue B, les locks, les sanitizers ou les fichiers générés. Aucun fallback de compatibilité ajouté.	Les corrections restent localisées autour des frontières identifiées pendant la vérification.	 Respectées

Coordination extension et panneau

- [coordinateur liveSession des panneaux Prompt](#)
- Ajout de la disposition stricte `created` | `revealed`.
- Installation provisoire du descripteur pour préserver la sécurité d'un dispose synchrone.
- Restauration du descripteur réellement attaché lorsque le panneau existant est seulement révélé.
- Suppression correcte d'un ancien descripteur lorsqu'un nouveau panneau classique est réellement créé.
- [façade d'ouverture des panneaux](#)
- Le low-level opener distingue désormais explicitement une création d'un reveal.
- La disposition est retournée au coordinateur sans introduire de dépendance inverse.
- [tests du coordinateur Prompt](#)

- Vérification qu'un second lease présent dans une query de reveal ne remplace jamais le lease réellement chargé par la webview.
- Vérification que la fermeture libère uniquement le premier lease acquis.
- [tests de la façade d'ouverture](#)
- Vérification que deux ouvertures du même locator produisent successivement `created`, puis `revealed`.
- Conservation de l'absence de seconde création physique.

Compositeur normalPrompt

- [contrats frontend liveSession](#)
- Le contrat de présentation reçoit maintenant obligatoirement l'état `replayComposerArmed`.
- [machine d'état Open/replay](#)
- Le latch est armé après hydratation et acceptation de l'observer exact.
- Il reste actif pendant `liveSessionReplayFailed` et le Retry `preserveLocal`.
- Il est désarmé dans la même transition qu'un terminal direct ou qu'un handoff direct.
- Le passage en mode normal reste une barrière défensive supplémentaire.
- [adaptateur normalPrompt](#)
- L'interactivité ne dépend plus du mode grossier `liveSessionStreaming`.
- Le brouillon et les attachments restent modifiables après armement, même pendant l'échec ou la réouverture Retry.
- Send, l'historique interactif et le cleanup restent verrouillés jusqu'au handoff.
- [intégration Prompt](#)
- Propagation du latch vers le modèle final.
- Conservation des politiques statiques de Send, historique et cleanup.
- Correction des commentaires qui décrivaient encore toutes les sessions live comme strictement read-only.
- [tests de l'adaptateur normalPrompt](#)
- Distinction entre première ouverture non armée et Retry déjà armé.
- Vérification de l'interactivité pendant l'échec.
- [tests de la machine normalPrompt](#)
- Vérification complète de la durée de vie du latch.
- Vérification du retour terminal au mode normal sans résidu d'état live.
- [tests de l'adaptateur EasyAgents](#)
- Adaptation au contrat strict sans changement de son UX read-only.

Thinking par tentative

- [lifecycle thinking Prompt](#)
- Reset explicite du séparateur pending à la frontière exacte du `streamInstanceId`.
- [tests de continuation thinking](#)
- Vérification que l'état `TOOL_USE` d'une ancienne tentative ne survit pas dans la suivante.

Documentation

- [documentation générique liveSession](#)
- Documentation de la propriété du descripteur physique.
- Documentation de la distinction création/reveal.
- Documentation du latch d'armement du compositeur et de sa conservation à travers les retries.

Résultat final

Les **deux bugs matériels** et le **défaut latent d'isolation thinking** identifiés pendant la vérification sont maintenant corrigés, avec des contrats stricts et des tests ciblant directement les courses concernées. ✅

Nom du commit

text

Copy

fix(live-session): preserve panel leases and replay composer state

🎵 Les leases restent fidèles 🛠️ Le brouillon survit aux retries, et le thinking repart propre 🎵

</ASSISTANT>

SHOW ALLPage 1 / 1