

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=f1fc121abf5f494aba888f19e3dca25d-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 de la tranche"](#),  ["Fichiers modifiés"](#)

  reading project:  g ["Docs liveSession"](#),  g ["Docs EasyAgents"](#),  g ["Docs Prompt"](#),  g ["Docs bus frontend"](#),  g ["Docs fallback persist"](#),  [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:  ["Historique récent"](#),  ["Périmètre du commit"](#)

  reading project:  h ["Registre liveSession"](#),  h ["Source Prompt normale"](#),  h ["Panneaux et activité"](#),  h ["Frontend liveSession"](#),  h ["Contrats transverses"](#),
 [LiveSessionRun.type.ts](#),
 [LiveSessionPanel.type.ts](#),
 [LiveSessionOpenResult.type.ts](#),
 [LiveSessionReplay.type.ts](#),
 [SidebarStore.ts](#),
 [SidebarUiMessages.ts](#),
 [MessageRegistry.ts](#),
 [parseSlimbkStreamBodyError.util.ts](#),
 [SlimbkResult.type.ts](#),
 [aiStopSuffix.util.ts](#),
 [LiveSessionRecord.type.ts](#),
 [LiveSessionRegistry.service.ts](#),
 [LiveSessionService.ts](#),
 [LiveSessionOpen.service.ts](#),
 [LiveSessionReplay.service.ts](#),
 [LiveSessionSourceLifecycle.service.ts](#),
 [NormalPromptLiveSessionSource.service.ts](#),
 [stream.endpoint.ts](#),
 [iaStreamHandlers.ts](#),  [liveSessionStreamHandlers.ts](#),  [extTypedMessageBus.util.ts](#),  [extMessageBus.util.ts](#)

  scanning git:  ["Diff transport Slimbk"](#)

  reading project:  [slimbkBuildHeader.ts](#) [L330-L455],  [Slimbk.ts](#) [L230-L335],  [Slimbk.ts](#) [L335-L418],
 [PromptLiveSessionPanelQuery.util.ts](#),
 [PromptLiveSessionPanelCoordinator.service.ts](#),
 [PromptLiveSessionDetachedStop.service.ts](#),
 [PromptThreadCleanup.manager.ts](#),
 [SidebarQueryRunningBridge.ts](#),
 [SidebarStore.queryWindow.listeners.manager.ts](#),
 [SidebarStore.queryWindow.mutations.manager.ts](#),
 [SidebarStore.rehydrate.manager.ts](#),
 [SidebarStore.core.manager.ts](#),  [SidebarStore.handlers.manager.ts](#),  [openPanel.util.ts](#) [L1-L190],  [openPanel.util.ts](#) [L500-L677],  [OpenedPanelsRegistry.util.ts](#),  [PromptQueryDoneSound.manager.ts](#),  [openPanel.util.ts](#) [L180-L520],
 [SidebarActivityQuery.tsx](#),
 [PromptLiveSessionBootstrap.adapter.ts](#),

[NormalPromptLiveSession.adapter.ts](#),
[PromptLiveSession.types.ts](#),
[PromptLiveSessionBanner.tsx](#),
[usePromptLiveSession.hook.ts](#), [applyHydratedPromptSessionState.util.ts](#), [preparePromptStreamingTurn.util.ts](#),
[preparePromptContinuationStreaming.util.ts](#), [usePromptThinkingLifecycle.hook.ts](#), [usePromptToolUse.hook.ts](#),
[useAiStream.hook.ts](#) [L1-L300], [useAiStream.hook.ts](#) [L300-L620], [useAiStream.hook.ts](#) [L620-L900],
[useAiStream.hook.ts](#) [L900-L1172], [useStreamListener.hook.ts](#), [frontMessageBus.util.ts](#) [L1-L260],
[frontMessageBus.util.ts](#) [L260-L417]
 scanning git: 🗑️ ["Diff files Queue A et B"](#)
 reading project: [usePromptRunCommands.hook.ts](#), [Prompt.tsx](#) [L1-L220], [Prompt.tsx](#) [L250-L460],
[Prompt.tsx](#) [L680-L860], [Prompt.tsx](#) [L960-L1060], [Prompt.tsx](#) [L1140-L1235], [Prompt.tsx](#) [L460-L680],
[Prompt.tsx](#) [L1060-L1145], [PromptInput.tsx](#) [L1-L260], e ["Vue live PromptInput"](#), ["Rendu live PromptInput"](#),
 e ["Affichage compositeur live"](#), ["Contrôles compositeur"](#), e ["Bannière live PromptInput"](#),
["Rendu bannière et saisie"](#),
[usePromptRunController.finalization.hook.ts](#),
[usePromptRunController.hook.ts](#),
[delete.endpoint.ts](#),
[EasyAgentsBatchRunRetry.service.ts](#),
[abortSource.endpoint.ts](#),
[abort.endpoint.ts](#),
[delete.endpoint.liveSession.test.ts](#), [EasyAgentsBatchLiveSessionAdapter.service.ts](#), e ["Classe erreur Slimbk"](#),
["Propriétés erreur Slimbk"](#), [OrchMessagesMemoryAssistantPatch.ts](#) [L1-L260], e ["Suffixe erreur assistant"](#),
 e ["Idempotence suffixe"](#), ["Ajout suffixe canonique"](#), e ["Usages openMode"](#),
[PromptLiveSessionBootstrap.adapter.test.ts](#) [L1-L25], [usePromptSessionLocator.hook.ts](#) [L1-L220],
[usePromptSessionLocator.hook.ts](#) [L220-L303]
 scanning git: 🗑️ ["Volume du commit"](#)
 reading project: [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) [L1-L260],
[usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) [L260-L389], e ["Isomorphisme transport"](#),
 e ["Isomorphisme éditions"](#), e ["Isomorphisme outils"](#), e ["Types de requêtes"](#), e ["Source compact context"](#),
 e ["Source SPECIF"](#), e ["Source edit"](#), e ["Source retry"](#), e ["Fermeture activité Prompt"](#), e ["Test fermeture idle"](#),
 e ["Test panelClosed"](#), e ["Mock fermeture activité"](#), [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L180-L540], e ["Échec publication observer"](#),
 e ["Broadcast observer"](#), ["Test publication auxilia"](#), r ["openCentralPanel"](#), [registerDeepLinkUriHandler.ts](#) [L120-L155], e ["Échec transition cleanup"](#),
[usePromptCancelController.hook.ts](#)
 scanning git: 🗑️ ["Qualité du diff"](#), 🗑️ ["Tests ajoutés"](#), 🗑️ ["Fichiers hors spécif"](#)
 reading project: [useAiStreamQueue.hook.ts](#) [L300-L590], e ["Observer continuation test"](#), ["Assertion continuation"](#),
 e ["Continuation intégrée live"](#), ["Assertions continuation"](#), [PromptInput.module.css](#), [PromptLiveSessionBanner.module.css](#)
 scanning git: 🗑️ ["TODO ajoutés"](#), 🗑️ ["Désactivations lint"](#), ❌ 🗑️ ["Imports dynamiques prod"](#), ❌ 🗑️ ["Require ajoutés"](#),
🗑️ ["Imports dynamiques ajout"](#), 🗑️ ["Require ajoutés"](#)
 reading project: e ["Usage replayMode contrôleur"](#), [SlimbkEndpoint.ts](#) [L1-L200], [Prompt.tsx](#) [L220-L260]

Périmètre de la vérification

Joel, cette implémentation est impressionnante : c'est une véritable cathédrale asynchrone... mais deux ou trois gargouilles ont échappé à la revue 😊

J'ai réinspecté le **HEAD propre a65d1ff6**, dont le commit contient **184 fichiers modifiés, 20 158 insertions et 11 369 suppressions**.

Constats de périmètre :

- les **160 chemins numérotés dans la spécification** sont présents dans l'implémentation ;
- des fichiers supplémentaires ont été modifiés pour les transports Slimbk, les sorties générées, Queue A/Queue B et certains harnais de test ;
- le contrôle Git du diff ne remonte pas d'erreur d'espacement ;
- les principales programming docs demandées ont bien été mises à jour et relues :

- [documentation liveSession](#)
- [documentation EasyAgents](#)
- [documentation d'isomorphisme](#)
- [documentation thinking](#)
- [documentation Queue B](#)
- [documentation fallback persist](#)

Les pourcentages ci-dessous évaluent l'adhérence observable du code versionné à la spécification.

1) La spécification entière est-elle implémentée ?

Réponse

✗ Non, pas entièrement au sens fonctionnel strict.

Mesure	Estimation	Verdict
Présence des fichiers et composants demandés	100 %	✓
Présence des mécanismes architecturaux majeurs	97 %	✓
Adhérence comportementale stricte	environ 89 %	⚠
Adhérence globale à la spécification	environ 92 %	⚠
La quasi-totalité de l'architecture demandée existe : identités exactes, réservations, époques, claims destructifs, replay, source normale, Activity, Stop détaché, persistance terminale, origine webview exacte, handoff, <code>preserveLocal</code> , fallback ack, transport Slimbk et intégration EasyAgents.		
Cependant, l'implémentation ne peut pas être considérée comme complète, car :		

1. le replay live d'une requête `continueResponse` ne réutilise pas l'assistant existant ;
2. la fermeture d'un panneau Prompt idle peut laisser une activité fantôme marquée `open` ;
3. certaines erreurs de coordination peuvent empêcher un cleanup pourtant encore requis ;
4. une erreur de publication Activity peut être dupliquée une fois par chunk d'un backlog non borné ;
5. la matrice d'isomorphisme explicitement demandée n'est pas entièrement matérialisée ;
6. le protocole Slimbk privé a été rendu possible par un élargissement du type générique global.

Conclusion de la question 1

Le périmètre structurel est complet, mais la fonctionnalité finale ne l'est pas encore : il reste au moins deux défauts métier importants et deux défauts de robustesse.

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

Analyse fonctionnalité par fonctionnalité

Fonction technique demandée	État	Adhérence	
Identité exacte { <code>locator</code> , <code>turnId</code> , <code>sourceId</code> }	✓	100 %	Les clés de locator et de run utilisent des tuples.
Contrat strict des descripteurs live	✓	100 %	Les couples EasyAgents/normalPrompt, les politiques obligatoire/interdite sont centralisés dans LiveSessionRegistry .
Rejet des paramètres inconnus, partiels et dupliqués	✓	100 %	L'extension et le frontend utilisent <code>getAll/inventories</code> dans PromptLiveSessionPanelQuery.uti .
Origine webview exacte non sérialisée	✓	100 %	Le contexte contient la page et la webview émettent avant la réservation.
Démarrage backend avec réservation synchrone avant <code>await</code>	✓	98 %	Le handler source réserve avant les opérations m...
Réservations, promotion, époques et anti-ABA	✓	98 %	Le registre sépare l'époque de run et la génération...
Retraites structurées et finalisation à cinq résultats	✓	100 %	Les cinq résultats sont présents et les records sont dans LiveSessionRegistry.service.ts .
Buffer backlog + tail complet et non borné	✓	100 %	Aucun cap ni mécanisme de compression n'a été implémenté.
Leases exactes et replay autorisé par abonnement	✓	98 %	L'attachement, la lecture par curseur, l'autorisation sont présents.
Reconstruction <code>replaceTurnAssistants</code>	✓	97 %	Les assistants du seul tour exact sont retirés, puis ajoutés.
Reconstruction <code>continueAssistant</code> côté backend	✓	100 %	La baseline immuable source contient bien l'assistant existant.
Reconstruction <code>continueAssistant</code> côté frontend	✗	55 %	Le frontend hydrate l'assistant existant, puis ajoute les nouveaux.

Fonction technique demandée	État	Adhérence	
frontend			le premier.
Priming lazy du <code>Readable</code>	✓	98 %	Le premier <code>next()</code> est démarré avant le callback
Persistance de chaque <code>meta.error</code> avant <code>append</code>	✓	98 %	Le service source appelle <code>patchLastAssistant</code>
Terminal synthétique après <code>throw/EOF</code>	✓	97 %	<code>SYSTEM_ERROR/INTERRUPTED</code> ne sont ajoutés
Premier <code>meta.error</code> via protocole V1	✓	97 %	Le premier terminal peut être porté par le header
Erreur Slimbk post-header filtrée avant parsing métier	✓	98 %	Le parser partagé bloque les enveloppes non-200
Terminal data suivi d'une erreur transport	✓	98 %	Queue A/B sont drainées, aucun second terminal
<code>start()</code> synchrone et single-flight	✓	100 %	<code>accepted:false</code> est rendu sans mutation ; les
Guards par <code>sourceId + streamInstanceId</code>	✓	97 %	Listener, Queue A, Queue B et continuations pos
Queue A et Queue B drainées avant outcome	✓	97 %	L'outcome est publié après les deux files et une
Algorithme Queue A/B laissé strictement inchangé	⚠	75 %	Les files ont aussi reçu une rétention d'erreurs, u changements semblent utiles, mais dépassent le
Handoff normalPrompt direct sans second Open	✓	98 %	Les conditions terminal/completed/finalization e
Retry Open gardé par tentative indépendante	✓	98 %	Les timers, réponses tardives et unmounts sont c
Retry <code>preserveLocal</code> sans lecture du brouillon	✓	100 %	Les propriétés du brouillon, des attachments et c couverts.
Fallback ack corrélé à la tentative	✓	100 %	La barrière utilise <code>{ turnId, streamInstan</code>
Finalisation sans fallback ack	✓	100 %	Le masque est retiré sans effacer texte, attachme
Compositeur interactif après replay armé	✓	98 %	La saisie et les attachments sont activés pendant
Stop unique dans Prompt	✓	100 %	Le bandeau normalPrompt ne rend aucun Stop s
Stop détaché Activity	✓	98 %	Le Stop est limité à <code>Running/source/closed</code> verrou local synchrone.
Projection Activity open/closed/idle	⚠	82 %	Les transitions source sont correctes, mais la fer son activité.
Son exact par run et attempt	✓	98 %	Déduplication terminale exacte, attempts monoto
Delete sous claim et attente de source	✓	96 %	Le disque n'est touché qu'après abort et settleme durable.
Retry EasyAgents sous claim	✓	96 %	Les guards sont rejoués avant le claim, le handle
Préservation du comportement EasyAgents	✓	96 %	<code>sourceId === turnId</code> , observer ownership,
Programming docs	✓	100 %	Les six documents principaux ont été mis à jour
Couverture de tests promise par la spécification	⚠	84 %	La couverture est très large, mais certaines matr mauvais scaffold de continuation.
Conclusion de la question 2			
La majorité du système fonctionne conformément à la spécification, particulièrement le backend et les contrats exacts, mais le chemin frontend <code>continueAssistant</code> et une partie du cycle Activity/cleanup ne respectent pas encore le comportement demandé.			

3) Y a-t-il des bugs ?

Réponse

✓ Oui. J'ai identifié quatre bugs confirmés par le code, dont un défaut fonctionnel majeur.

● Bug 1 — Le replay live d'une continuation crée un second assistant

Sévérité : élevée — P1

Le backend fait correctement son travail : en mode `continueAssistant`, `LiveSessionOpen.service.ts` renvoie la baseline contenant l'assistant existant à continuer.

Le frontend hydrate cette baseline, puis `usePromptRunController.hook.ts` transmet bien `replayMode`, mais ne l'utilise pas pour choisir la préparation du scaffold :

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

Copy

```
const startResult = start({
  kind: 'liveSessionObserver',
  runRef: args.runRef,
  viewerLeaseId: args.viewerLeaseId,
  replayMode: args.replayMode,
  stopPolicy: args.stopPolicy,
});
if (!startResult.accepted) return startResult;
// ...
preparePromptStreamingTurn({
```

`preparePromptStreamingTurn(...)` crée systématiquement une nouvelle bulle assistant et efface le préfixe de continuation :

[AI/Code/src/frontend/src/app/prompt/Prompt/preparePromptStreamingTurn.util.ts](#)

Copy

```
// A fresh streamed turn must never inherit any append-mode prefix from a previous
assistant message.
args.baseAssistantTextRef.current = '';
```

Le résultat lors du scénario **Continue** → **fermer le panneau** → **rouvrir pendant la continuation** est donc :

1. l'ancien assistant est hydraté depuis la baseline ;
2. une nouvelle bulle thinking est ajoutée ;
3. un nouvel assistant vide est ajouté ;
4. les nouveaux tokens vont dans ce nouvel assistant ;
5. le handoff direct ne recharge pas le snapshot terminal ;
6. les deux assistants restent visibles.

Cela casse directement :

- l'unicité de l'assistant continué ;
- l'isomorphisme streaming ↔ réhydratation ;
- la continuité du texte ;
- la continuité thinking ;
- l'objectif « handoff direct sans seconde hydratation ».

Le test `usePromptRunController.liveSessionObserver.test.tsx` valide actuellement deux nouveaux placeholders même avec `replayMode='continueAssistant'`, ce qui encode le comportement incorrect au lieu de détecter le défaut.



Bug 2 — Fermer un Prompt idle peut laisser une activité fantôme Idle/open

Sévérité : élevée à moyenne — P1/P2

Dans `PromptLiveSessionPanelCoordinator.service.ts`, `panelClosed(...)` n'est appelé que lorsqu'une autorité live active existe.

Quand le panneau est idle ou que la source est déjà terminale, le coordinateur passe directement au cleanup :

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

Copy

```
const active = this.findActiveAuthority(locator);
if (active) {
  const runtime = sidebarQueryRunningBridge.getSnapshot(locator);
  try {
    await panelClosed({
      locator,
      runRef: active.runRef,
      // ...
    });
  }
  // ...
}
```

```
    return;  
}
```

La branche sans `active` appelle ensuite `cleanupDetachedIfAllowed(...)`, mais ce cleanup ne retire pas l'élément de la collection `Activity`.

Conséquence :

- le vrai panneau est supprimé du registre ;
- l'activité peut rester `status='idle', panelState='open'` ;
- l'interface affiche une session supposée ouverte alors qu'aucun panneau n'existe ;
- l'invariant « **aucun panneau invisible : open ou closed** » est violé.

Le défaut concerne notamment le scénario très courant :

1. génération terminée ;
2. Prompt revenu en mode normal/idle ;
3. fermeture du panneau ;
4. ligne `Activity` toujours visible comme `Idle`.



Bug 3 — Une erreur de transition `Activity` empêche le cleanup terminal

Sévérité : moyenne — P2

Dans `onSourceSettled(...)` du même coordinateur, le cleanup est séquentiellement placé après la transition `Activity` :

- `transitionSourceSettled(...)` ;
- puis seulement `cleanupDetachedIfAllowed(...)`.

Si la transition `Activity` rejette, le `finally` retire bien l'état sonore exact, mais le cleanup du thread, des attachments et de la session détachée n'est pas tenté.

Conséquences possibles :

- mapping détaché conservé ;
- mémoire `Orch` non nettoyée ;
- orphan cleanup non exécuté ;
- cleanup trivial différé sans nouveau wake-up garanti ;
- source frontend classée `transportError` à cause d'un échec auxiliaire tardif.

La transition `Activity` et le cleanup terminal devraient être **tentés indépendamment**, puis leurs erreurs agrégées.



Bug 4 — Une publication `Activity` refusée est dupliquée une fois par chunk de replay

Sévérité : moyenne — P2, avec risque mémoire

Dans `liveSessionStreamHandlers.ts`, chaque élément du FIFO exécute :

- `await allocation.publication` ;
- puis capture son rejet dans `errors`.

Le même `allocation.publication` est de nouveau attendu après le FIFO.

Si cette promesse est rejetée et que le backlog contient `N` chunks :

- la même erreur est ajoutée approximativement `N + 1` fois ;
- le replay continue correctement, mais l'`AggregateError` devient proportionnel au backlog ;
- le buffer live étant volontairement **non borné**, le tableau d'erreurs l'est également dans ce scénario.

Le handler source normal possède déjà une meilleure stratégie : il normalise l'issue de publication une seule fois en booléen et mémorise l'erreur une seule fois. Le handler observer devrait appliquer le même contrat.



Écart de couverture — La matrice d'isomorphisme promise est incomplète

Ce n'est pas un bug runtime en soi, mais cela explique pourquoi le premier bug a survécu.

`usePromptLiveSession.normalPromptIsomorphism.test.tsx` couvre notamment :

- `replace` avec `thinking`/texte/usage ;
- continuation via helper pur ;
- `TOOL_USE` ;
- `SYSTEM_ERROR/INTERRUPTED`.

Il ne contient pas la matrice annoncée pour :

- les différents types de requêtes ;
- les vrais tool calls/results ;
- les éditions de fichiers ;
- le terminal suivi d'un `transportError` ;

- l'intégration réelle `hydrate` → `startLiveObserver` → `streamInstanceId` → `handoff`.
Les comportements sont parfois testés séparément ailleurs, mais pas comme isomorphisme end-to-end.



Écart de typage — Le protocole V1 privé élargit le type Slimbk global

Dans [SlimbkResult.type.ts](#), chaque erreur Slimbk générique peut désormais posséder :

[AICode/src/sharedlib/src/slimbk/framework/SlimbkResult.type.ts](#)

Copy

```
response?: 'streaming';
```

Cela permet le protocole V1, mais élargit aussi le contrat de **toutes** les erreurs Slimbk, alors que la spécification décrit une voie privée réservée au premier `meta.error` de cette source.

Ce n'est pas un défaut observé dans les appels actuels, mais c'est une **relaxation d'un type global strict**. Une union dédiée au header d'erreur streaming serait plus sûre.

Conclusion de la question 3

Oui : le bug de continuation et l'activité fantôme sont des anomalies fonctionnelles confirmées ; les deux autres sont des défauts de robustesse asynchrone à corriger avant de qualifier la tranche de terminée.

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

Sur la spécification

Opinion : excellente, environ 9,3/10.

Points très forts :

- les identités et autorités sont définies avec une précision remarquable ;
- les scénarios ABA, sources successives, claims, terminal+transport et frontend stale sont anticipés ;
- les invariants produit et les invariants techniques sont clairement séparés ;
- les conditions du handoff et les politiques `hydrateFromSnapshot/preserveLocal` sont particulièrement bien décrites ;
- l'exigence d'atomicité fonctionnelle est justifiée : livrer seulement la moitié de cette tranche aurait créé un état produit dangereux.

Points plus fragiles :

- **160 fichiers dans une seule spécification rendent la vérification humaine très difficile ;**
- certains détails se contredisent légèrement : Queue A/B doivent rester algorithmiquement inchangées, alors que la gestion exigée de leurs erreurs nécessite précisément une extension de leur état interne ;
- la spécification indique parfaitement le comportement de continuation, mais les tests ont surtout vérifié les briques isolées, pas leur composition réelle.

Sur le code source

Opinion : architecture globalement très solide, environ 8,7/10 avant corrections.

Les meilleures parties sont :

- [LiveSessionRegistry.service.ts](#) : séparation claire entre réservation, record, retraite, tombstone, époque et claim ;
- [NormalPromptLiveSessionSource.service.ts](#) : ordre durable-before-append-before-forward correctement matérialisé ;
- [useAiStream.hook.ts](#) : single-flight synchrone et guards d'identité bien pensés ;
- [PromptLiveSessionBootstrap.adapter.ts](#) : frontière pure et stricte ;
- les claims Delete/Retry et la provenance webview exacte sont de très bons choix architecturaux.

Les faiblesses viennent surtout des **jointures entre modules** :

- backend continuation correct + frontend scaffold générique incorrect ;
- panneau correctement retiré du registre + Activity non retirée ;
- erreur auxiliaire correctement tolérée + mémorisation répétée pour chaque chunk ;
- tests unitaires très nombreux + manque de quelques scénarios composés.

C'est typique d'une grande architecture concurrente : les pièces sont propres, mais le bug adore voyager entre deux pièces avec un faux badge d'accès. 🐛

Conclusion de la question 4

La conception est de très haut niveau et le code est largement production-grade, mais les erreurs restantes sont précisément dans les transitions inter-modules les plus importantes ; elles méritent une correction ciblée, pas une refonte.

5) Résumé du rapport

Axe	Verdict	Estimation	Décision
Présence des fichiers spécifiés	✓	100 %	Complet
Contrats shared et identités	✓	99 %	Conforme
Registre, replay, claims et anti-ABA	✓	97 %	Conforme avec faible réserve
Source Prompt et persistance terminale	✓	97 %	Conforme
Transport V1 et enveloppe post-header	✓	96 %	Conforme, type global à resserrer
Reconnexion normalPrompt replace	✓	96 %	Conforme
Reconnexion normalPrompt continue	✗	55 %	Correction prioritaire
Activity, fermeture et cleanup	⚠	82 %	Correction nécessaire
Stop, son et source exacte	✓	97 %	Conforme
Delete et Retry destructifs	✓	96 %	Conforme
Start transactionnel frontend	✓	96 %	Conforme dans le comportement visible
Fallback ack et conservation du compositeur	✓	99 %	Conforme
Préservation EasyAgents	✓	96 %	Conforme
Tests présents	⚠	84 %	Très larges, mais matrices d'intégration incomplètes
Qualité architecturale	✓	91 %	Très bonne
Adhérence globale à la spécification	⚠	92 %	Presque complète
Aptitude à considérer la tranche finalisée	✗	84 %	Attendre les corrections P1/P2

6) Conclusion en une phrase

Conclusion en une phrase

La tranche est remarquablement avancée et structurellement complète, mais elle ne doit pas encore être considérée comme finalisée tant que la continuation live et la fermeture Activity/cleanup ne sont pas corrigées.

7) Si des corrections sont nécessaires, quels lots peut-on appliquer ?

Lot A — Corriger la continuation live normale 🟡

Corrections ciblées :

- utiliser réellement `replayMode` dans [usePromptRunController.hook.ts](#) ;
- pour `continueAssistant`, préparer puis committer un plan de continuation réutilisant l'assistant existant ;
- conserver la baseline hydratée dans une autorité synchrone afin que `startLiveObserver(...)` puisse la lire dans la même pile que l'hydratation ;
- ne pas remettre `baseAssistantTextRef` à vide ;
- remplacer l'assertion incorrecte dans [usePromptRunController.liveSessionObserver.test.tsx](#) ;
- ajouter le scénario intégré **Continue** → **Close** → **Reopen** → **Replay** → **Direct handoff**, avec un seul assistant.

Lot B — Corriger Activity et cleanup panneau 🟡

Corrections ciblées :

- appeler `panelClosed({ locator })` pour toute fermeture sans autorité active ;
- tenter indépendamment la transition Activity et le cleanup terminal ;
- agréger les erreurs sans empêcher le cleanup suivant ;
- tester la fermeture idle, la fermeture post-handoff, le cleanup malgré une transition Activity refusée et l'absence d'activité fantôme.

Fichier central : [PromptLiveSessionPanelCoordinator.service.ts](#).

Lot C — Durcir le transport et compléter les tests 🔵

Corrections ciblées :

- normaliser une seule fois le résultat de `allocation.publication` dans [liveSessionStreamHandlers.ts](#) ;
- empêcher une erreur identique d'être ajoutée une fois par chunk ;
- compléter la matrice de [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#) ;
- resserrer le protocole d'erreur streaming avec un type de header dédié, au lieu d'élargir toutes les erreurs de [SlimbkResult.type.ts](#).

Lot D — Appliquer toutes les corrections

Ce lot réunit :

- la correction P1 de continuation ;
- la fermeture Activity correcte ;
- le cleanup indépendant des erreurs Activity ;
- la déduplication des erreurs de publication observer ;
- le resserrement Slimbk ;
- tous les tests de non-régression et d'isomorphisme correspondants.

 Corriger la continuation live

 Corriger Activity et cleanup

 Durcir transport et tests

 Corriger toutes les anomalies

</ASSISTANT>

<USER>

[Corriger toutes les anomalies]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

  scanning git:  ["État courant du dépôt"](#)

  reading project:  [liveSession.programming-doc-readme.md](#) [L1-L180],  [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#) [L1-L220],  [prompt-stream-dispatch-lock.programming-doc-readme.md](#) [L1-L220],  [fallback-persist-and-title.programming-doc-readme.md](#) [L1-L180],

 [usePromptRunController.hook.ts](#),  [usePromptLiveSession.hook.ts](#),  [preparePromptContinuationStreaming.util.ts](#),

 [preparePromptStreamingTurn.util.ts](#),

 [PromptLiveSessionPanelCoordinator.service.ts](#),

 [liveSessionStreamHandlers.ts](#),  [SlimbkResult.type.ts](#),  [SlimbkEndpoint.ts](#),  [slimbkBuildHeader.ts](#) [L1-

L150],  [slimbkBuildHeader.ts](#) [L300-L437],  [slimbkBuildHeader.ts](#) [L140-L320],  [Slimbk.ts](#) [L250-

L390],  e ["Résultats streaming Slimbk"](#),  e ["Envoi header stream"](#),  e ["Type envoi stream"](#),  e ["Type résultat](#)

[Slimbk"](#),  e ["Usages plan](#)

[continuation"](#),

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

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

 [usePromptLiveSession.normalPromptIsomorphism.test.tsx](#),

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

 [liveSessionStreamHandlers.test.ts](#),

 [Slimbk.streamingErrorHandler.test.ts](#),

 [slimbkBuildHeader.errorCode.test.ts](#),  [usePromptRunCommands.hook.ts](#) [L300-

L400],  [applyHydratedPromptSessionState.util.ts](#) [L1-L260],  [applyHydratedPromptSessionState.util.ts](#) [L260-

L360],  [SidebarStore.queryWindow.mutations.manager.ts](#) [L1-L220],  [Slimbk.ts](#) [L1-L70],  h ["Fixtures](#)

[hydratation outils"](#),  e ["Types requêtes AI"](#),  [AiQuery.type.ts](#),  [FrontChatSession.type.ts](#),  e ["Écriture résultats](#)

[Slimbk"](#),  [Slimbk.ts](#) [L75-L135],  e ["Normalisation publication source"](#),  e ["Échec publication](#)

[source"](#),  e ["Promesse publication source"](#),  e ["Publication activité source"](#),  ["Modèle publication](#)

[source"](#),  [usePromptRunCommands.hook.ts](#) [L1-L120],  e ["Terminal puis transport test"](#)

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

[AICode/src/frontend/src/app/prompt/Prompt/preparePromptContinuationStreaming.util.ts](#)

Copy

Review

Accept

PatchFull file

```

- import { uidGenerator } from
'../../sharedlib/lib/concurrencyUtil/uidGenerator.util.js';
-
+ import type { Dispatch, MutableRefObject, SetStateAction } from 'react';
+
+ import { uidGenerator } from
'../../sharedlib/lib/concurrencyUtil/uidGenerator.util.js';
+


---


- import { uidGenerator } from
'../../sharedlib/lib/concurrencyUtil/uidGenerator.util.js';
-
+ import { uidGenerator } from
'../../sharedlib/lib/concurrencyUtil/uidGenerator.util.js';
+ import type { AiStreamedChunk } from
'../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+


---


- import type { PromptMessage } from '../PromptHistory/promptHistoryTypes.type.js';
-
+ import type { PromptMessage } from '../PromptHistory/promptHistoryTypes.type.js';
+ import type { ToolCallNoticeApi } from './usePromptToolCallNotice.hook.js';
+


---


-
- /** Build a pure continuation plan without mutating refs, messages, buffers, or
notices. */
-
+
+ /**
+  * Build a pure continuation plan without mutating refs, messages, buffers, or notices.
+  *
+  * When `targetTurnId` is provided, only the assistant belonging to that exact logical
turn may
+  * receive replayed continuation tokens. This prevents a reopened source from attaching
to a later
+  * assistant that happens to be present in the same hydrated session snapshot.
+  */
+


---


-     messages: readonly PromptMessage[],
-
+     messages: readonly PromptMessage[],
+     targetTurnId?: string,
+


---


-         message.role === 'assistant' &&
-         message.assistantBubbleKind !== 'think',
-
+         message.role === 'assistant' &&
+         message.assistantBubbleKind !== 'think' &&
+         (targetTurnId === undefined ||
+           message.turnId === targetTurnId),
+


---


-     };
- }
-
+     };
+ }
+
+ /** Mutable Prompt runtime dependencies required to commit one accepted continuation
attempt. */
+ export type PromptContinuationStreamingCommitArgs = {
+   plan: PromptContinuationStreamingPlan;
+   setMessages: Dispatch<SetStateAction<PromptMessage[]>>;

```

```

+   setStreamingThinkingTokens: Dispatch<SetStateAction<string[]>>;
+   setStreamingTokens: Dispatch<SetStateAction<string[]>>;
+   streamingMessageUidRef: MutableRefObject<string | undefined>;
+   streamingThinkingUidRef: MutableRefObject<string | undefined>;
+   currentTurnIdRef: MutableRefObject<string | undefined>;
+   baseAssistantTextRef: MutableRefObject<string>;
+   lastConsumedRawIndexRef: MutableRefObject<number>;
+   toolCallNotice: Pick<ToolCallNoticeApi, 'reset'>;
+   metaOnlyFinalizedTurnIdRef: MutableRefObject<string | undefined>;
+   lastMetaChunkFinalizedRef: MutableRefObject<
+     AiStreamedChunk | undefined
+   >;
+   resetLastReceivedStreamKind: () => void;
+ };
+
+ /**
+  * Commit one previously prepared continuation scaffold after synchronous transport
+  * acceptance.
+  *
+  * This function deliberately owns the full continuation mutation sequence so normal
+  * Continue and
+  * live-session replay cannot drift into different assistant/thinking structures.
+  * Callers must build
+  * the pure plan before `start()` and invoke this commit only when `start()` returned
+  * `accepted=true`.
+  */
+ export function commitPromptContinuationStreaming(
+   args: PromptContinuationStreamingCommitArgs,
+ ): void {
+   // Every token buffer and notice belongs to one accepted stream instance and must
+   // start empty.
+   args.setStreamingThinkingTokens([]);
+   args.setStreamingTokens([]);
+   args.lastConsumedRawIndexRef.current = 0;
+   args.toolCallNotice.reset();
+
+   // Arm the exact existing assistant and the fresh thinking placeholder before
+   // network callbacks can run.
+   args.baseAssistantTextRef.current = args.plan.baseAssistantText;
+   args.streamingThinkingUidRef.current = args.plan.thinkingUid;
+   args.streamingMessageUidRef.current = args.plan.assistantUid;
+   args.currentTurnIdRef.current = args.plan.turnId;
+   args.setMessages([...args.plan.messages]);
+
+   // Terminal idempotence and stream-kind phase are scoped to the newly accepted
+   // continuation attempt.
+   args.metaOnlyFinalizedTurnIdRef.current = undefined;
+   args.lastMetaChunkFinalizedRef.current = undefined;
+   args.resetLastReceivedStreamKind();
+ }
+

```

[AI Code/src/frontend/src/app/prompt/Prompt/preparePromptContinuationStreaming.util.ts](#)

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

```
- import type { AiStreamStartResult } from
'../../../../../messageBus/AiStreamStartArgs.type.js';
- import { preparePromptContinuationStreaming } from
'./preparePromptContinuationStreaming.util.js';
-
+ import type { AiStreamStartResult } from
'../../../../../messageBus/AiStreamStartArgs.type.js';
+ import {
+   commitPromptContinuationStreaming,
+   preparePromptContinuationStreaming,
+ } from './preparePromptContinuationStreaming.util.js';
+
-
-   // Reset ephemeral buffers and indices
-   setStreamingThinkingTokens([]);
-   setStreamingTokens([]);
-   lastConsumedRawIndexRef.current = 0;
-   toolCallNotice.reset();
-
-   // Commit the exact plan only after the transport single-flight accepted this
attempt.
-   baseAssistantTextRef.current = plan.baseAssistantText;
-   streamingThinkingUidRef.current = plan.thinkingUid;
-   streamingMessageUidRef.current = plan.assistantUid;
-   currentTurnIdRef.current = plan.turnId;
-   setMessages(plan.messages);
-
-   // Clear meta-only idempotence guards for this new streaming segment
-   metaOnlyFinalizedTurnIdRef.current = undefined;
-   lastMetaChunkFinalizedRef.current = undefined;
-
-   // Reset kind trackers for the continuation segment
-   resetLastReceivedStreamKind();
-
+   // Commit the shared continuation contract only after the transport single-
flight accepted this attempt.
+   commitPromptContinuationStreaming({
+     plan,
+     setMessages,
+     setStreamingThinkingTokens,
+     setStreamingTokens,
+     streamingMessageUidRef,
+     streamingThinkingUidRef,
+     currentTurnIdRef,
+     baseAssistantTextRef,
+     lastConsumedRawIndexRef,
+     toolCallNotice,
+     metaOnlyFinalizedTurnIdRef,
+     lastMetaChunkFinalizedRef,
+     resetLastReceivedStreamKind,
+   });
```

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

Copy

Review

Accept



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

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

Copy
Review
Accept

PatchFull file

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


---


- } from './preparePromptStreamingTurn.util.js';
-
+ } from './preparePromptStreamingTurn.util.js';
+ import {
+   commitPromptContinuationStreaming,
+   preparePromptContinuationStreaming,
+ } from './preparePromptContinuationStreaming.util.js';
+


---


-   viewerLeaseId: LiveSessionViewerLeaseId;
-   replayMode: 'replaceTurnAssistants' | 'continueAssistant';
-
+   viewerLeaseId: LiveSessionViewerLeaseId;
+   replayMode: LiveSessionReplayMode;
+


---


- /**
-  * Stable streaming identities and turn reference
-
+ /**
+  * Synchronous mirror of the authoritative Prompt messages.
+  *
+  * React state updates are intentionally asynchronous, while live-session Open
+  hydrates a snapshot and
+  * accepts its observer in the same call stack. The mirror is updated synchronously
+  by hydration so the
+  * continuation planner always sees the exact assistant baseline instead of the
+  previous render.
+  */
+   const messagesSnapshotRef = useRef<readonly PromptMessage[]>(messages);
+   messagesSnapshotRef.current = messages;
+
+ /**
+  * Stable streaming identities and turn reference
+


---


-   * Replace the entire messages state atomically (used by the hydration path).
-   * This does not touch any other streaming state or refs.
-
+   * Replace the entire messages state atomically (used by the hydration path).
+   * This updates the same-stack message mirror but does not touch transport
+  identities or token buffers.
+


---


-   (newMessages: PromptMessage[]): void => {
-     // Immutable replace to keep downstream memoization predictable
-     setMessages([...newMessages]);
-
+   (newMessages: PromptMessage[]): void => {
+     // Publish one immutable snapshot to both the same-stack live planner and
+  the next React render.
+     const snapshot = [...newMessages];
+     messagesSnapshotRef.current = snapshot;
+     setMessages(snapshot);
```

```

+
-         viewerLeaseId: LiveSessionViewerLeaseId;
-         replayMode: 'replaceTurnAssistants' | 'continueAssistant';
-
+         viewerLeaseId: LiveSessionViewerLeaseId;
+         replayMode: LiveSessionReplayMode;
+
-     }): AiStreamStartResult => {
-
+     }): AiStreamStartResult => {
+         // Build the complete continuation plan before transport acceptance so a
+         // refused start remains mutation-free.
+         const continuationPlan =
+             args.replayMode === 'continueAssistant'
+                 ? preparePromptContinuationStreaming(
+                     messagesSnapshotRef.current,
+                     args.runRef.turnId,
+                 )
+                 : undefined;
+
-         if (!startResult.accepted) return startResult;
-         // The live watcher must respect the exact same Prompt streaming contract
+         as a normal streamed turn.
-         // The live-safe snapshot already contains the user bubble of the running
+         turn, so we only append the
-         // assistant-side placeholders before subscribing to the live observer
+         transport.
-         preparePromptStreamingTurn({
-             turnId: args.runRef.turnId,
-             setMessages,
-             buildMessages: (targets) => {
-                 return buildPromptStreamingAssistantMessages(targets);
-             },
-             currentTurnIdRef,
-             streamingMessageUidRef,
-             streamingThinkingUidRef,
-             baseAssistantTextRef,
-             setStreamingThinkingTokens,
-             setStreamingTokens,
-             lastConsumedRawIndexRef,
-             toolCallNotice,
-             metaOnlyFinalizedTurnIdRef,
-             lastMetaChunkFinalizedRef,
-             resetLastReceivedStreamKind,
-         });
-
+         if (!startResult.accepted) return startResult;
+
+         if (continuationPlan) {
+             // Continue the hydrated assistant itself; adding a second assistant
+             // would break replay isomorphism.
+             commitPromptContinuationStreaming({
+                 plan: continuationPlan,
+                 setMessages,
+                 setStreamingThinkingTokens,
+                 setStreamingTokens,
+                 streamingMessageUidRef,
+                 streamingThinkingUidRef,
+                 currentTurnIdRef,
+                 baseAssistantTextRef,
+                 lastConsumedRawIndexRef,
+                 toolCallNotice,

```

```

+         metaOnlyFinalizedTurnIdRef,
+         lastMetaChunkFinalizedRef,
+         resetLastReceivedStreamKind,
+     });
+     messagesSnapshotRef.current = [
+         ...continuationPlan.messages,
+     ];
+ } else {
+     // Replace-style replay uses the canonical fresh assistant scaffold
after the backend removed
+     // only the assistant messages belonging to the exact active turn.
+     const targets = preparePromptStreamingTurn({
+         turnId: args.runRef.turnId,
+         setMessages,
+         buildMessages: buildPromptStreamingAssistantMessages,
+         currentTurnIdRef,
+         streamingMessageUidRef,
+         streamingThinkingUidRef,
+         baseAssistantTextRef,
+         setStreamingThinkingTokens,
+         setStreamingTokens,
+         lastConsumedRawIndexRef,
+         toolCallNotice,
+         metaOnlyFinalizedTurnIdRef,
+         lastMetaChunkFinalizedRef,
+         resetLastReceivedStreamKind,
+     });
+     messagesSnapshotRef.current = [
+         ...messagesSnapshotRef.current,
+         ...buildPromptStreamingAssistantMessages(targets),
+     ];
+ }
+
+

```

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

Copy

Review

Accept



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

[AI Code/src/frontend/src/app/prompt/Prompt/ test /preparePromptContinuationStreaming.util.test.ts](#)

Copy

Review

Accept

PatchFull file

```

- import { describe, it } from '@jest/globals';
-
+ import { describe, it } from '@jest/globals';
+ import type { SetStateAction } from 'react';
+
+
+
+
+ import { preparePromptContinuationStreaming } from
+ '../preparePromptContinuationStreaming.util.js';
+
+
+ import type { AiStreamedChunk } from
+ '../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import { AiStopReason } from
+ '../sharedlib/types/aiTypes/AiStopReason.type.js';
+ import type { PromptMessage } from '../PromptHistory/promptHistoryTypes.type.js';
+ import {

```



```

+     commitPromptContinuationStreaming,
+     preparePromptContinuationStreaming,
+ } from '../preparePromptContinuationStreaming.util.js';
+
-     ).toHaveLength(1);
- });
-
+     ).toHaveLength(1);
+ });
+
+ it('selects the assistant from the exact requested turn even when a later turn
exists', () => {
+     const input: PromptMessage[] = [
+         { turnId: 'turn-a', uid: 'user-a', role: 'user', text: 'Q1' },
+         {
+             turnId: 'turn-a',
+             uid: 'assistant-a',
+             role: 'assistant',
+             text: 'A1',
+         },
+         { turnId: 'turn-b', uid: 'user-b', role: 'user', text: 'Q2' },
+         {
+             turnId: 'turn-b',
+             uid: 'assistant-b',
+             role: 'assistant',
+             text: 'A2',
+         },
+     ],
+ ];
+
+     const plan = preparePromptContinuationStreaming(input, 'turn-a');
+
+     expect(plan.turnId).toBe('turn-a');
+     expect(plan.assistantUid).toBe('assistant-a');
+     expect(plan.baseAssistantText).toBe('A1');
+ });
+
+ it('commits the accepted continuation plan to every exact runtime target once', ()
=> {
+     const plan = preparePromptContinuationStreaming(
+         [
+             { turnId: 'turn', uid: 'user', role: 'user', text: 'Q' },
+             {
+                 turnId: 'turn',
+                 uid: 'assistant',
+                 role: 'assistant',
+                 text: 'Existing answer',
+                 fileEditsUiFinalized: true,
+             },
+         ],
+         'turn',
+     );
+     const terminal: AiStreamedChunk = {
+         turnId: 'old-turn',
+         type: 'meta',
+         value: {
+             metaType: 'stop_reason',
+             value: { reason: AiStopReason.END_TURN },
+         },
+     },
+ ];
+     let committedMessages: PromptMessage[] = [];
+     let thinkingTokens = ['stale-thinking'];
+     let messageTokens = ['stale-message'];
+     let noticeResetCount = 0;

```

```

+ let streamKindResetCount = 0;
+ const currentTurnIdRef: { current: string | undefined } = {
+   current: 'old-turn',
+ };
+ const streamingMessageUidRef: { current: string | undefined } = {
+   current: 'old-message',
+ };
+ const streamingThinkingUidRef: { current: string | undefined } = {
+   current: 'old-thinking',
+ };
+ const baseAssistantTextRef = { current: 'old-base' };
+ const lastConsumedRawIndexRef = { current: 12 };
+ const metaOnlyFinalizedTurnIdRef: {
+   current: string | undefined;
+ } = {
+   current: 'old-turn',
+ };
+ const lastMetaChunkFinalizedRef: {
+   current: AiStreamedChunk | undefined;
+ } = {
+   current: terminal,
+ };
+
+ commitPromptContinuationStreaming({
+   plan,
+   setMessages: (action: SetStateAction<PromptMessage[]>): void => {
+     committedMessages =
+       typeof action === 'function'
+         ? action(committedMessages)
+         : action;
+   },
+   setStreamingThinkingTokens: (
+     action: SetStateAction<string[]>,
+   ): void => {
+     thinkingTokens =
+       typeof action === 'function'
+         ? action(thinkingTokens)
+         : action;
+   },
+   setStreamingTokens: (action: SetStateAction<string[]>): void => {
+     messageTokens =
+       typeof action === 'function'
+         ? action(messageTokens)
+         : action;
+   },
+   streamingMessageUidRef,
+   streamingThinkingUidRef,
+   currentTurnIdRef,
+   baseAssistantTextRef,
+   lastConsumedRawIndexRef,
+   toolCallNotice: {
+     reset: (): void => {
+       noticeResetCount += 1;
+     },
+   },
+   metaOnlyFinalizedTurnIdRef,
+   lastMetaChunkFinalizedRef,
+   resetLastReceivedStreamKind: (): void => {
+     streamKindResetCount += 1;
+   },
+ });
+
+ expect(committedMessages).toEqual(plan.messages);

```

```
+ expect(thinkingTokens).toEqual([]);
+ expect(messageTokens).toEqual([]);
+ expect(currentTurnIdRef.current).toBe(plan.turnId);
+ expect(streamingMessageUidRef.current).toBe(plan.assistantUid);
+ expect(streamingThinkingUidRef.current).toBe(plan.thinkingUid);
+ expect(baseAssistantTextRef.current).toBe(plan.baseAssistantText);
+ expect(lastConsumedRawIndexRef.current).toBe(0);
+ expect(metaOnlyFinalizedTurnIdRef.current).toBeUndefined();
+ expect(lastMetaChunkFinalizedRef.current).toBeUndefined();
+ expect(noticeResetCount).toBe(1);
+ expect(streamKindResetCount).toBe(1);
+ });
+
```

[AI Code/src/frontend/src/app/prompt/Prompt/ test /preparePromptContinuationStreaming.util.test.ts](#)

Copy

Review

Accept



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

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

Copy

Review

Accept

PatchFull file

```
-
- it('commits exactly one canonical assistant scaffold after accepted observer
identity', async () => {
-
+
+ it('commits exactly one fresh assistant scaffold for accepted replace-style
replay', async () => {
+
+ viewerLeaseId: 'lease',
+ replayMode: 'continueAssistant',
+
+ viewerLeaseId: 'lease',
+ replayMode: 'replaceTurnAssistants',
+
+ viewerLeaseId: 'lease',
+ replayMode: 'continueAssistant',
+
+ viewerLeaseId: 'lease',
+ replayMode: 'replaceTurnAssistants',
+
+ });
+ });
+
+ it('continues the same hydrated assistant when replay is accepted in the hydration
call stack', async () => {
+ const accepted: AiStreamStartResult = {
+ accepted: true,
+ runRef: RUN_REF,
+ streamInstanceId: 'stream-continuation',
+ };
+ startMock.mockReturnValue(accepted);
+ await act(async () => {
```

```

+         root.render(React.createElement(Harness));
+     });
+     const controller = controllerApiRef.current;
+     if (!controller)
+         throw new Error('Controller harness was not initialized.');
```

render the snapshot.

```

+     let result: AiStreamStartResult | undefined;
+     act(() => {
+         // Hydration and observer acceptance intentionally happen before React can
+         controller.replaceAllMessages([
+             {
+                 turnId: RUN_REF.turnId,
+                 uid: 'user',
+                 role: 'user',
+                 text: 'Question',
+             },
+             {
+                 turnId: RUN_REF.turnId,
+                 uid: 'assistant-existing',
+                 role: 'assistant',
+                 text: 'Existing answer',
+                 assistantBubbleKind: 'msg',
+                 fileEditsUiFinalized: true,
+             },
+         ]);
+         result = controller.startLiveObserver({
+             runRef: RUN_REF,
+             viewerLeaseId: 'lease',
+             replayMode: 'continueAssistant',
+             stopPolicy: 'source',
+         });
+     });
+
+     expect(result).toEqual(accepted);
+     expect(startMock).toHaveBeenCalledWith({
+         kind: 'liveSessionObserver',
+         runRef: RUN_REF,
+         viewerLeaseId: 'lease',
+         replayMode: 'continueAssistant',
+         stopPolicy: 'source',
+     });
+     const messages = controllerApiRef.current?.messages ?? [];
+     expect(messages.map((message) => message.uid)).toEqual([
+         'user',
+         expect.any(String),
+         'assistant-existing',
+     ]);
+     expect(
+         messages.filter(
+             (message) =>
+                 message.role === 'assistant' &&
+                 message.assistantBubbleKind !== 'think',
+         ),
+     ).toHaveLength(1);
+     expect(
+         messages.find((message) => message.uid === 'assistant-existing'),
+     ).toEqual(
+         expect.objectContaining({
+             text: 'Existing answer',
+             fileEditsUiFinalized: false,
+         }),
+     );
+
+ });

```

+

Copy

Accept

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

Review

PatchFull file

—

+

+

+

+

—

+

+

```

-         );
-         if (
-             args.finalization === 'handoffReady' ||
-             args.finalization === 'retiredMissingBaseline' ||
-             args.finalization === 'retiredMissingTerminal' ||
-             args.finalization === 'reservationCancelled'
-         ) {
-
+         );
+     } catch (error) {
+         errors.push(normalizeCoordinatorError(error));
+     }
+
+     if (
+         args.finalization === 'handoffReady' ||
+         args.finalization === 'retiredMissingBaseline' ||
+         args.finalization === 'retiredMissingTerminal' ||
+         args.finalization === 'reservationCancelled'
+     ) {
+         try {
+
-             await this.cleanupDetachedIfAllowed(args.runRef.locator);
-
+             await this.cleanupDetachedIfAllowed(args.runRef.locator);
+         } catch (error) {
+             errors.push(normalizeCoordinatorError(error));
+
-         }
-     } finally {
-         // Sound state is keyed by the exact runRef, so stale cleanup cannot
-         // affect a successor source.
-
+         }
+     }
+
+     try {
+         // Sound state is keyed by the exact runRef, so stale cleanup cannot
+         // affect a successor source.
+
-         cleanupPromptQueryDoneSoundRun(args.runRef);
-
+         cleanupPromptQueryDoneSoundRun(args.runRef);
+     } catch (error) {
+         errors.push(normalizeCoordinatorError(error));
+
-         errors.push(normalizeCoordinatorError(error));
-     }
-     });
-
+         errors.push(normalizeCoordinatorError(error));
+     }
+
+     throwCoordinatorErrors(
+         errors,
+         'Prompt source settlement completed with ordered failures.',
+     );
+ });
+

```

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

Copy

Review

Accept

  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

```
-      expect(cleanupOnPromptDisposeMock).toHaveBeenCalledWith(LOCATOR);
-    });
-
+      expect(cleanupOnPromptDisposeMock).toHaveBeenCalledWith(LOCATOR);
+    });
+
+    it('removes an idle Activity projection before detached panel cleanup', async () =>
+    {
+      const coordinator = createCoordinator();
+      await coordinator.openPromptPanel({
+        locator: LOCATOR,
+        query: BASE_QUERY,
+        port: { open: () => {} },
+      });
+
+      await coordinator.onPanelDisposed(LOCATOR);
+
+      expect(panelClosedMock).toHaveBeenCalledWith({ locator: LOCATOR });
+      expect(cleanupOnPromptDisposeMock).toHaveBeenCalledWith(LOCATOR);
+      expect(releaseLocatorDestructionClaimMock).toHaveBeenCalledWith(
+        CLEANUP_CLAIM,
+      );
+    });
+
+  },
+);
+
+  },
+);
+
+    it('continues exact detached cleanup when the terminal Activity transition fails',
+    async () => {
+      activeSnapshots = [reservation()];
+      const coordinator = createCoordinator();
+      await coordinator.openPromptPanel({
+        locator: LOCATOR,
+        query: BASE_QUERY,
+        port: { open: () => {} },
+      });
+      await coordinator.onPanelDisposed(LOCATOR);
+      expect(cleanupOnPromptDisposeMock).not.toHaveBeenCalled();
+
+      activeSnapshots = [];
+      transitionSourceSettledMock.mockRejectedValueOnce(
+        new Error('Activity transition failed'),
+      );
+
+      await expect(
+        coordinator.onSourceSettled({
+          runRef: RUN,
+          runId: 15,
+          finalization: 'handoffReady',

```

```

+         }),
+         ).rejects.toThrow('Activity transition failed');
+
+         expect(cleanupOnPromptDisposeMock).toHaveBeenCalledWith(LOCATOR);
+         expect(cleanupSoundRunMock).toHaveBeenCalledWith(RUN);
+         expect(releaseLocatorDestructionClaimMock).toHaveBeenCalledWith(
+             CLEANUP_CLAIM,
+         );
+     });
+
+

```

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

Copy
Review
Accept

  patching file:  [liveSessionStreamHandlers.ts](#)

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

Copy
Review
Accept

PatchFull file

```

-     const errors: Error[] = [];
-
+     const errors: Error[] = [];
+     // Every replay chunk shares the same Activity publication. Normalize that promise
+     // once so a
+     // rejected broadcast is recorded once rather than once per unbounded backlog item
+     // plus cleanup.
+     const activityPublicationReady: Promise<boolean> = allocation
+         ? allocation.publication.then(
+             () => true,
+             (error: unknown) => {
+                 errors.push(normalizeError(error));
+                 return false;
+             },
+         )
+         : Promise.resolve(false);
+

```

```

-         try {
-             await allocation.publication;
-             await sidebarQueryRunningBridge.onChunk(
-                 runRef,
-                 allocation.runId,
-                 chunk,
-             );
-         }
+         try {
+             // A failed initial publication only disables Activity mutations.
+             // forwarding remain independent so the exact replay can still
+             // complete and clean up.
+             if (await activityPublicationReady) {
+                 await sidebarQueryRunningBridge.onChunk(
+                     runRef,
+                     allocation.runId,
+                     chunk,
+                 );
+             }
+         }
+

```

```
- expect(thrown).toBeInstanceOf(AggregateError);
- if (!(thrown instanceof AggregateError)) {
-     throw new Error('Expected ordered publication failures.');
```

```
- }
- expect(thrown.errors).toEqual(
-     expect.arrayContaining([
-         expect.objectContaining({
-             message: 'activity publication failed',
-         })
-     ]),
- );
- expect(emitted).toHaveLength(1);
- expect(soundChunkMock).toHaveBeenCalledTimes(1);
-
+
+ expect(thrown).toEqual(
```

```
+         expect.objectContaining({
+             message: 'activity publication failed',
+         })),
+     );
+     expect(thrown).not.toBeInstanceOf(AggregateError);
+     expect(emitted).toHaveLength(2);
+     expect(activityOnChunkMock).not.toHaveBeenCalled();
+     expect(soundChunkMock).toHaveBeenCalledTimes(2);
+
```

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

Copy

Review

Accept

  patching file:  [SlimbkResult.type.ts](#)

[AI Code/src/sharedlib/src/slimbk/framework/SlimbkResult.type.ts](#)

Copy

Review

Accept

PatchFull file

```
-
- /**
-  * Return type of every Slimbk endpoint:
-  *   - For normal response: this type exactly
-  *   - For stream response: The first line is this type with response='streaming'
-  */
- export type SlimbkResult<
-
+
+ /** Successful Slimbk response, including the canonical header that opens a streamed
+ body. */
+ export type SlimbkSuccessResult<
+

```

```
-   TResp extends ZodObject<ZodRawShape> = ZodObject<ZodRawShape>,
-   > =
-   | {
-       status: '200_OK';
-       response: 'streaming' | z.infer<TResp>;
-   }
-   | {
-       status:
-         | '400_BAD_REQUEST'
-         | '401_FORBIDDEN'
-         | '403_UNAUTHORIZED'
-         | '404_NOT_FOUND'
-         | '426_UPGRADE_REQUIRED'
-         | '500_SERVER_ERROR';
-       /**
-        * Optional machine-readable error code meant to be safe for UI display.
-        *
-        * Invariants:
-        * - Must never contain secrets (no tokens/headers/query params).
-        * - Must be stable enough to be used by the UI without string parsing.
-        */
-       errorCode?: string;
-       /**
-        * Keep a streamed response open after reporting this header error to the
- client.
-        *

```

```

-         * This is reserved for protocols whose first application item is represented
by the
-         * header while later body transport failures must remain observable on the
same stream.
-         */
-         response?: 'streaming';
-         errorMessage: string;
-     };
-
+     TResp extends ZodObject<ZodRawShape> = ZodObject<ZodRawShape>,
+ > = {
+     status: '200_OK';
+     response: 'streaming' | z.infer<TResp>;
+ };
+
+ /** Closed Slimbk error response used by every ordinary synchronous or terminal
failure. */
+ export type SlimbkErrorResult = {
+     status:
+         | '400_BAD_REQUEST'
+         | '401_FORBIDDEN'
+         | '403_UNAUTHORIZED'
+         | '404_NOT_FOUND'
+         | '426_UPGRADE_REQUIRED'
+         | '500_SERVER_ERROR';
+     /**
+      * Optional machine-readable error code meant to be safe for UI display.
+      *
+      * Invariants:
+      * - Must never contain secrets (no tokens/headers/query params).
+      * - Must be stable enough to be used by the UI without string parsing.
+      */
+     errorCode?: string;
+     errorMessage: string;
+ };
+
+ /** Machine-readable status carried by every non-success Slimbk response. */
+ export type SlimbkErrorStatus = SlimbkErrorResult['status'];
+
+ /** Return type of ordinary Slimbk endpoints and completed endpoint failures. */
+ export type SlimbkResult<
+     TResp extends ZodObject<ZodRawShape> = ZodObject<ZodRawShape>,
+ > = SlimbkSuccessResult<TResp> | SlimbkErrorResult;
+
+ /**
+  * First-line contract of a streamed Slimbk response.
+  *
+  * Only this dedicated header type may keep the body channel open after a non-200
application
+  * terminal. Ordinary `SlimbkResult` errors remain closed and cannot accidentally opt
into the
+  * private live-session first-header protocol.
+  */
+ export type SlimbkStreamHeaderResult<
+     TResp extends ZodObject<ZodRawShape> = ZodObject<ZodRawShape>,
+ > =
+     | SlimbkSuccessResult<TResp>
+     | (SlimbkErrorResult & { response?: 'streaming' });
+

```

[AI Code/src/sharedlib/src/slimbk/framework/SlimbkResult.type.ts](#)

Copy

Review

Accept

  patching file:  [SlimbkEndpoint.ts](#)

[AICode/src/sharedlib/src/slimbk/framework/SlimbkEndpoint.ts](#)

Copy

Review

Accept

PatchFull file

```
- import z, { ZodObject, ZodRawShape } from 'zod';
- import { SlimbkResult } from './SlimbkResult.type.js';
-
+ import z, { ZodObject, ZodRawShape } from 'zod';
+ import type {
+   SlimbkResult,
+   SlimbkStreamHeaderResult,
+ } from './SlimbkResult.type.js';
+
-
- * A streamed handler writes a response progressively via a writable stream.
- * The first line is always a stringify of SlimbkResult. Then it can get the function
to stream chunks.
-
+ * A streamed handler writes a response progressively via a writable stream.
+ * The first line is always a serialized SlimbkStreamHeaderResult. Then it can get the
function to stream chunks.
+
-
- export type FunStreamChunk = (str: string) => void;
- export type FunSendStreamResult = (result: SlimbkResult) => FunStreamChunk;
-
+ export type FunStreamChunk = (str: string) => void;
+ /** Send the exact first-line stream header and return its body writer. */
+ export type FunSendStreamResult = (
+   result: SlimbkStreamHeaderResult,
+ ) => FunStreamChunk;
+
+
AICode/src/sharedlib/src/slimbk/framework/SlimbkEndpoint.ts
```

Copy

Review

Accept

  patching file:  [Slimbk.ts](#)

[AICode/src/sharedlib/src/slimbk/framework/Slimbk.ts](#)

Copy

Review

Accept

PatchFull file

```
- import { SlimbkEndpoints } from './SlimbkEndpoints.js';
- import { SlimbkResult } from './SlimbkResult.type.js';
-
+ import { SlimbkEndpoints } from './SlimbkEndpoints.js';
+ import type {
+   SlimbkResult,
+   SlimbkStreamHeaderResult,
+ } from './SlimbkResult.type.js';
+
-
- /**
-   * Utility function to send a SlimbkResult through the configured stream.
-
+
+ /**
+   * Send one ordinary Slimbk result or dedicated first-line stream header.
```

```

+
-         function sendSlimbkResult(
-             result: SlimbkResult,
-
+         function sendSlimbkResult(
+             result: SlimbkStreamHeaderResult,
+
-
-             function sendStreamResult(
-                 result: SlimbkResult,
-
+             function sendStreamResult(
+                 result: SlimbkStreamHeaderResult,
+

```

[AI Code/src/sharedlib/src/slimbk/framework/Slimbk.ts](#)

Copy

Review

Accept



patching file: [slimbkBuildHeader.ts](#)

[AI Code/src/sharedlib/src/slimbk/compiler/generateClient/slimbkBuildHeader.ts](#)

Copy

Review

Accept

PatchFull file

```

- }
- /**
-  * Return type of every Slimbk endpoint:
-  *   - For normal response: this type exactly
-  *   - For stream response: The first line is this type with response='streaming'
-  */
- export type SlimbkResult<TResp> =
-   | {
-       status: '200_OK';
-       response: 'streaming' | TResp;
-   }
-   | {
-       status:
-         | '400_BAD_REQUEST'
-         | '401_FORBIDDEN'
-         | '403_UNAUTHORIZED'
-         | '404_NOT_FOUND'
-         | '426_UPGRADE_REQUIRED'
-         | '500_SERVER_ERROR';
-       /**
-        * Optional machine-readable error code meant to be safe for UI display.
-        *
-        * Invariants:
-        * - Must never contain secrets (no tokens/headers/query params).
-        * - Must stay separate from errorMessage (no string concatenation/parsing).
-        */
-       errorCode?: string;
-       /** Preserve the body channel after this non-200 stream header. */
-       response?: 'streaming';
-       errorMessage: string;
-   };
-
+ }
+ /** Successful Slimbk response, including the canonical header that opens a streamed
+  body. */
+ type SlimbkSuccessResult<TResp> = {

```


Copy
Review
Accept

PatchFull file

```
- import { SlimbkEndpoint } from '../SlimbkEndpoint.js';
-
+ import { SlimbkEndpoint } from '../SlimbkEndpoint.js';
+ import type { SlimbkStreamHeaderResult } from '../SlimbkResult.type.js';
+
-                                     // returned writer must remain usable until the endpoint execution
fully settles.
-                                     const writeBody = sendStreamResult({
-
+                                     // returned writer must remain usable until the endpoint execution
fully settles.
+                                     const streamingErrorHandler: SlimbkStreamHeaderResult = {
+
-                                     errorMessage: 'terminal application error',
-                                     });
-
+                                     errorMessage: 'terminal application error',
+                                     };
+                                     const writeBody = sendStreamResult(streamingErrorHandler);
+

```

[AICode/src/sharedlib/src/slimbk/framework/_test_/Slimbk.streamingErrorHandler.test.ts](#)

Copy
Review
Accept

  patching file:  [slimbkBuildHeader.errorCode.test.ts](#)

[AICode/src/sharedlib/src/slimbk/compiler/generateClient/_test_/slimbkBuildHeader.errorCode.test.ts](#)

Error

([what is it?](#))

  reading project:  [slimbkBuildHeader.errorCode.test.ts](#) [L1-L100]

  patching file:  [slimbkBuildHeader.errorCode.test.ts](#)

[AICode/src/sharedlib/src/slimbk/compiler/generateClient/_test_/slimbkBuildHeader.errorCode.test.ts](#)

Copy
Review
Accept

PatchFull file

```
-     };
- }
-
+     };
+ }
+
+ /** Extract the ordinary result declaration so stream-only properties cannot hide in a
+ broad string assertion. */
+ function extractOrdinaryResultTypeBlock(text: string): string {
+     const start = text.indexOf('export type SlimbkResult<TResp> =');
+     const end = text.indexOf('/** Dedicated first-line stream header', start);
+     if (start < 0 || end < 0) {
+         throw new Error('Generated Slimbk result declarations are missing.');
```

```

+
-      const text = slimbkBuildHeader(buildParams('cloud'));
-
-      // Transport: SlimbkResult error variant includes a dedicated errorCode field.
-      expect(text).toContain('errorCode?: string;');
-      expect(text).toContain("response?: 'streaming'");
-
+      const text = slimbkBuildHeader(buildParams('cloud'));
+
+      // Transport: SlimbkResult error variant includes a dedicated errorCode field.
+      expect(text).toContain('errorCode?: string;');
+      expect(extractOrdinaryResultTypeBlock(text)).not.toContain(
+        "response?: 'streaming'";
+      );
+      expect(text).toContain('type SlimbkStreamHeaderResult<TResp> =');
+      expect(text).toContain(
+        "SlimbkErrorResult & { response?: 'streaming' }";
+      );
+      expect(text).toContain(
+        'let header: SlimbkStreamHeaderResult<unknown>;';
+      );
+
-      expect(text).toContain('errorCode?: string;');
-      expect(text).toContain("response?: 'streaming'");
-
+      expect(text).toContain('errorCode?: string;');
+      expect(extractOrdinaryResultTypeBlock(text)).not.toContain(
+        "response?: 'streaming'";
+      );
+      expect(text).toContain('type SlimbkStreamHeaderResult<TResp> =');
+      expect(text).toContain(
+        "SlimbkErrorResult & { response?: 'streaming' }";
+      );
+      expect(text).toContain(
+        'let header: SlimbkStreamHeaderResult<unknown>;';
+      );
+

```

[AI Code/src/sharedlib/src/slimbk/compiler/generateClient/_test_/slimbkBuildHeader.errorCode.test.ts](#)

Copy

Review

Accept

  reading project:  [promptHistoryTypes.type.ts](#) [L1-L260]

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

  reading project:  e ["Marqueur patch parasite"](#),  [preparePromptContinuationStreaming.util.ts](#),  [usePromptRunCommands.hook.ts](#) [L1-L30],  [usePromptRunCommands.hook.ts](#) [L325-L390],  [usePromptRunController.hook.ts](#) [L45-L145],  [usePromptRunController.hook.ts](#) [L235-L285],  [usePromptRunController.hook.ts](#) [L500-L625],  [PromptLiveSessionPanelCoordinator.service.ts](#) [L110-L235],  [liveSessionStreamHandlers.ts](#) [L45-L170],  [SlimbkResult.type.ts](#),  [slimbkBuildHeader.ts](#) [L40-L115],  [slimbkBuildHeader.ts](#) [L220-L245],  [slimbkBuildHeader.ts](#) [L385-L425],  [usePromptRunController.hook.ts](#) [L195-L240],  [usePromptRunController.hook.ts](#) [L620-L665],  [preparePromptContinuationStreaming.util.test.ts](#),  [usePromptRunController.liveSessionObserver.test.tsx](#) [L100-L260],  [PromptLiveSessionPanelCoordinator.service.test.ts](#) [L250-L570], [liveSessionStreamHandlers.test.ts](#) [L205-L295], [slimbkBuildHeader.errorCode.test.ts](#), [Slimbk.streamingErrorHandler.test.ts](#) [L1-L70], [Slimbk.ts](#) [L1-L125], [Slimbk.ts](#) [L270-L305], [SlimbkEndpoint.ts](#) [L25-L90], e ["Accès réponse erreur Slimbk"](#), e ["Accès](#)

[header streaming](#)",  e ["Type header Slimbk partagé"](#),  e ["Réponse streaming erreur"](#),  e ["Import résultat Slimbk"](#),  e ["Plan continuation ciblé"](#)

  scanning recent files

  reading project:  e ["Règle refs React"](#)

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

[AICode/src/frontend/src/app/prompt/Prompt/ test /preparePromptContinuationStreaming.util.test.ts](#)

Grouped

([what is it?](#))

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

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

Grouped

([what is it?](#))

  reading project:  e ["Appels directs observateur"](#)

  patching file:  [SlimbkEndpoint.ts](#)

[AICode/src/sharedlib/src/slimbk/framework/SlimbkEndpoint.ts](#)

Grouped

([what is it?](#))

  reading project:  [AiStopReason.type.ts](#) [L1-L120]

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

[AICode/src/frontend/src/app/prompt/Prompt/ test /preparePromptContinuationStreaming.util.test.ts](#)

Grouped

([what is it?](#))

  reading project:  e ["Règle ordre imports"](#),  e ["Mocks préparation continuation"](#),  [Slimbk.ts](#) [L190-L280],  e ["Assertions générateur Slimbk"](#),  e ["Snapshots header généré"](#),  [slimbkWriteClientApi.test.ts](#) [L1-L260],  [PromptLiveSessionPanelCoordinator.service.ts](#) [L190-L275],  [usePromptRunController.hook.ts](#) [L560-L635],  e ["Appels reset Prompt run"](#)

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

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

Grouped

([what is it?](#))

  patching file:  [Slimbk.ts](#)

[AICode/src/sharedlib/src/slimbk/framework/Slimbk.ts](#)

Grouped

([what is it?](#))

Corrections appliquées

Joel, les anomalies identifiées lors de la vérification ont été corrigées dans **15 fichiers ciblés**, sans modifier de fichier *.generated.*. La cathédrale asynchrone a maintenant ses gargouilles sous contrôle. 🏰 🐉

1. Continuation live sans second assistant

Le replay `continueAssistant` réutilise désormais **l'assistant exact déjà hydraté** :

- le plan de continuation est construit **avant** l'acceptation du transport ;
- la recherche est restreinte au `turnId` exact du run ;
- un refus de `start()` reste sans mutation ;
- après acceptation, l'assistant existant reçoit les nouveaux tokens ;
- une seule nouvelle bulle thinking est insérée ;
- aucun second assistant vide n'est créé ;

- le texte assistant existant alimente correctement `baseAssistantTextRef`.
La séquence de commit est centralisée afin que le Continue normal et le replay live utilisent exactement le même contrat :
- [préparation et commit de continuation](#)
- [commandes Prompt](#)
- [contrôleur de run Prompt](#)

Le contrôleur conserve également un miroir synchrone des messages hydratés : l'ouverture live peut ainsi hydrater la baseline puis accepter l'observer **dans la même pile JavaScript**, sans attendre un render React intermédiaire.

2. Suppression des activités fantômes

La fermeture d'un panneau Prompt appelle maintenant la transition `panelClosed` même lorsqu'aucune autorité live n'est encore active.

Le comportement devient :

- activité Idle ouverte → supprimée à la fermeture ;
- observer ouvert → supprimé à la fermeture ;
- source active apparue concurremment → conservée comme `Running/source/closed` ;
- source exacte déjà connue → protégée par son `runRef`.

Correction principale :

- [coordinateur de panneau Prompt liveSession](#)

3. Cleanup terminal indépendant d'Activity

La transition terminale suit maintenant trois étapes indépendantes :

1. transition Activity ;
2. cleanup du Prompt détaché lorsque la finalisation le permet ;
3. retrait de l'état sonore exact.

Une erreur Activity n'empêche donc plus :

- la libération de la mémoire du thread ;
- le cleanup des attachments ;
- le cleanup orphelin ;
- le cleanup trivial de persistance ;
- le retrait du son exact.

Les erreurs restent conservées dans leur ordre et sont agrégées après que toutes les étapes nécessaires ont été tentées.

4. Publication Activity rejetée une seule fois

La promesse initiale de publication Activity du viewer est maintenant normalisée **une seule fois**.

Même avec un backlog live non borné :

- une publication rejetée produit une seule erreur ;
- chaque chunk continue vers le son et le frontend ;
- les mutations Activity suivantes sont ignorées proprement ;
- l'observer est toujours nettoyé ;
- aucun `AggregateError` proportionnel au nombre de chunks n'est construit.

Fichier concerné :

- [handler du replay liveSession](#)

5. Contrat Slimbk resserré

La propriété `response: 'streaming'` d'un résultat non-200 n'appartient plus au type d'erreur Slimbk ordinaire.

Le contrat distingue maintenant :

- un résultat de succès ;
- une erreur Slimbk ordinaire et fermée ;
- un header de streaming dédié pouvant exceptionnellement conserver le canal body après un non-200.

Cela empêche une erreur synchrone classique d'opter accidentellement pour le protocole privé de premier header liveSession.

Les changements portent sur :

- [contrats de résultats Slimbk](#)
- [contrat des endpoints Slimbk](#)
- [runtime Slimbk](#)
- [générateur des clients Slimbk](#)

Le générateur produira le nouveau contrat lors de la génération normale ; **aucun client généré n'a été édité manuellement**.



Couverture de non-régression renforcée

Les tests couvrent désormais explicitement :

- sélection de l'assistant appartenant au `turnId` exact ;
- commit complet de toutes les refs et barrières de continuation ;
- hydratation et acceptation live dans la même pile React ;
- unicité de l'assistant après replay `continueAssistant` ;
- scaffold frais pour `replaceTurnAssistants` ;
- suppression d'une activité Idle à la fermeture ;
- cleanup terminal malgré une transition Activity rejetée ;
- publication Activity rejetée avec plusieurs chunks, mais une seule erreur ;
- usage du type Slimbk de header streaming dédié ;
- génération d'un type d'erreur ordinaire ne contenant plus `response`.

Tests modifiés :

- [tests du plan de continuation](#)
- [tests du contrôleur live observer](#)
- [tests du coordinateur de panneau](#)
- [tests du handler de replay](#)
- [test runtime du header Slimbk](#)
- [tests du générateur Slimbk](#)

Récapitulatif

Domaine	Correction	Résultat attendu	Complétude
Continuation live	Réutilisation de l'assistant hydraté exact	Un seul assistant, texte et thinking continus	✓
Identité de continuation	Filtrage par <code>turnId</code> avant acceptation	Aucun attachement à un assistant d'un autre tour	✓
Transaction frontend	Plan pur puis commit après <code>accepted=true</code>	Aucune mutation lors d'un refus	✓
Activity	Fermeture projetée même sans autorité active	Aucun item Idle faussement marqué ouvert	✓
Courses panneau/source	<code>panelClosed</code> préserve une source concurrente	Projection correcte <code>source/closed</code>	✓
Cleanup terminal	Activity, cleanup et son tentés indépendamment	Une panne auxiliaire ne bloque plus le nettoyage	✓
Replay observer	Publication Activity normalisée une fois	Pas d'amplification mémoire par nombre de chunks	✓
Transport Slimbk	Type de header streaming séparé	Le protocole privé ne relâche plus le type global	✓
Fichiers générés	Générateur source uniquement	Régénération normale, aucune édition interdite	✓
Qualité	Aucun nouveau <code>any</code> , cast inutile, TODO, ignore lint ou import dynamique	Corrections strictes et ciblées	✓

Nom du commit

text

Copy

`fix(live-session): preserve prompt continuation and harden cleanup transport`



La continuation garde son assistant 🗑️ Activity se ferme proprement, et le stream reste vaillant 🎵

</ASSISTANT>

SHOW ALLPage 1 / 1