

Permalink URI:

vscode://joelabenhaim.aicode-ja/chat?sessionId=cc78b5eec4194deb9942e6fbc178a854-VS-V%C3%A9rifier-finalise

<USER>

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

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

🔧  reading project: 📁 l "Racines du workspace"

🔧  scanning git: 📁 "État courant du dépôt", 📁 "Commit courant et récent", 📁 "Fichiers modifiés non idx",

📁 "Fichiers modifiés indexés"

🔧  reading project: 📁 g "Docs live session", 📁 g "Docs isomorphisme", 📁 g "Docs thinking Prompt", 📁 g "Docs verrous stream", 📁 g "Docs fallback persist", 📁 g "Docs EasyAgents", 📄 liveSession.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, 📄 easyAgents.programming-doc-readme.md

🔧  scanning git: 📁 "Inventaire du commit live", 📁 "Volume du commit live"

🔧  reading project: 📄 e "Ancien canAbortSource", 📄 e "Ancien query windows", 📄 e "Ancien type Activity", 📄 e "Ancienne identité stream", 📄 e "Tentatives Open replay", 📄 h "Contrats live exacts", 📄 h "Registre et source", 📄 h "Transport extension", 📄 h "Frontend live replay",

📄 r "LiveSessionRegistryService",

📄 r "NormalPromptLiveSessionSourceService",

📄 r "PromptLiveSessionPanelCoordinator", 📄 r "usePromptLiveSession", 📄 r "useAiStream", 📄 "Contrats shared live", 📄 "Registre et source live", 📄 "Transport et coordination", 📄 "Frontend live Prompt", 📄 e "Identité Open interne", 📄 e "Finalisation frontend", 📄 e "Génération source Prompt", 📄 e "Mode preserve local", 📄 e "Query live explicite", 📄 e "Fallback ack barrière", 📄 e "Body Slimbk parser", 📄 e "Contexte origine stream", 📄 e "Claim destruction locator", 📄 e "stopRequested runtime", 📄 SidebarStore.ts [L120-L215], 📄 MessageRegistry.ts [L1-

L180],

📄 AiStreamStartArgs.type.ts,

📄 PromptLiveSession.types.ts,

📄 PromptLiveSessionBootstrap.adapter.ts,

📄 PromptLiveSessionPanelQuery.util.ts, 📄 parseSlimbkStreamBodyError.util.ts, 📄 extTypedMessageBus.util.ts [L1-

L180], 📄 extMessageBus.util.ts [L1-L190], 📄 stream.endpoint.ts [L100-L220], 📄 useAiStream.hook.ts [L620-

L780], 📄 usePromptLiveSession.hook.ts [L300-

L430], 📄 applyHydratedPromptSessionState.util.ts, 📄 aiStopSuffix.util.ts [L180-L330], 📄 iaStreamHandlers.ts [L100-

L225], 📄 slimbkBuildHeader.ts [L1-L180], 📄 Slimbk.ts [L180-

L330], 📄 SlimbkResult.type.ts, 📄 LiveSessionOpen.service.ts, 📄 LiveSessionRegistry.service.ts [L400-

L760], 📄 PromptLiveSessionPanelCoordinator.service.ts [L110-

L461], 📄 PromptLiveSessionDetachedStop.service.ts, 📄 openPanel.util.ts [L1-

L330], 📄 SidebarActivityQuery.tsx, 📄 SidebarQueryRunningBridge.ts [L1-L390], 📄 extMessageBus.util.ts [L300-

L383], 📄 e "Client header streaming", 📄 e "Client erreur streaming", 📄 Slimbk.ts [L320-L421], 📄 e "Finalisation

outcome stream", 📄 e "Libération identité active", 📄 e "Stream completion callback", 📄 e "Start commit

send", 📄 e "Mode live explicite Prompt", 📄 e "Hydratation live désactivée", 📄 e "Auto run live

désactivée", 📄 slimbkBuildHeader.ts [L170-L442], 📄 useAiStream.hook.ts [L900-L1090], 📄 Prompt.tsx [L200-

L360], 📄 Prompt.tsx [L830-L1060], 📄 Prompt.tsx [L1160-L1445], 📄 usePromptRunCommands.hook.ts [L1-

L230], 📄 usePromptRunCommands.hook.ts [L230-L620], 📄 usePromptRunCommands.hook.ts [L620-

L820], 📄 Prompt.tsx [L360-L660], 📄 PromptQueryDoneSound.manager.ts, 📄 openPanel.util.ts [L330-

L684], 📄 e "Ouverture centrale refs", 📄 e "Démarrage son erreurs test", 📄 e "Démarrage son mock

throw", 📄 useAiStream.hook.ts [L1080-L1172], 📄 usePromptRunController.finalization.hook.ts [L1-

L260], 📄 usePromptRunController.hook.ts [L550-L710], 📄 e "Source Stop route Prompt", 📄 e "Interaction live

viewmodel", 📄 delete.endpoint.ts, 📄 EasyAgentsBatchRunRetry.service.ts [L1-

L260], 📄 PromptThreadCleanup.manager.ts [L1-L330], 📄 SidebarStore.rehydrate.manager.ts [L1-

L700], 📄 SidebarStore.queryWindow.mutations.manager.ts [L1-

L260],  [SidebarStore.queryWindow.listeners.manager.ts](#) [L1-L430],
 [SidebarStore.handlers.manager.ts](#),
 [abortSource.endpoint.ts](#),  [abort.endpoint.ts](#),  [EasyAgentsBatchRunRetry.service.ts](#) [L250-L520],  [PromptLiveSessionDetachedStop.service.test.ts](#) [L1-L220],  [SidebarStore.core.manager.ts](#),  e ["Mise à jour Sidebar sync"](#),  e ["Claim pendant Stop source"](#),  e ["Origin context construction"](#),  [extMessageBus.util.ts](#) [L250-L320],  [registerDeepLinkUriHandler.ts](#) [L110-L170],  [Prompt.tsx](#) [L1-L210],  [Prompt.tsx](#) [L650-L830],
 [usePromptSessionLocator.hook.ts](#),
 [PromptInput.tsx](#),
 [PromptLiveSessionBanner.tsx](#),
 [EasyAgentsBatchLiveSession.adapter.ts](#),  [OrchMessagesMemoryAssistantPatch.ts](#) [L1-L260],  [OrchMessagesMemoryAssistantPatch.errorSuffix.test.ts](#) [L1-L240],  e ["Normal source throw tests"](#),  e ["Persist error barrier tests"](#),  e ["Source error test names"](#),  [OrchMessagesMemoryAssistantPatch.ts](#) [L250-L380],  [NormalPromptLiveSessionSource.service.test.ts](#) [L1-L505],  [EasyAgentsBatchRunRetry.service.ts](#) [L500-L594],
 [specifActionAutoRunQuery.util.ts](#),
 [promptAutoRunQuery.util.ts](#),
 [promptAutoRunAgenticOptionsQuery.util.ts](#),
 [promptAutoRunUserPromptAugmentationQuery.util.ts](#),  e ["AutoRun neutralisé tests"](#),  e ["Terminal transport et Stop"](#),  e ["Source Stop tests normal"](#),  [usePromptLiveSession.handoff.test.tsx](#) [L350-L650],  [usePromptLiveSession.normalPrompt.test.tsx](#) [L250-L380],  e ["Readiness observable"](#),  [LiveSessionRegistry.service.ts](#) [L1-L110],  e ["Replace readiness test"](#),  e ["Normal readiness docs"](#),  e ["Baseline only normal tests"](#),  e ["Prop disableAfterClick"](#),  e ["Ancien source id absent"](#)
  scanning git: 🌀 ["Contrôle whitespace du tx"](#)
  reading
project:  [useAiStreamQueue.hook.ts](#),  [usePromptStreamDispatchGateQueue.hook.ts](#),  [PromptStreamLockStore.ts](#)
  scanning git: 🌀 ["Diff Queue A du commit"](#), 🌀 ["Diff Queue B du commit"](#)

Rapport de vérification

Joel, cette tranche est un véritable **poulpe asynchrone à 187 tentacules** 🐙 — et, honnêtement, la discipline appliquée aux identités exactes, aux barrières et aux courses concurrentes est impressionnante.

Base factuelle de l'audit

- Le dépôt est **propre** et positionné sur le commit 5818ab07c160dc20d751af638244b25ae7751fd2.
- Ce commit porte exactement le nom prévu : feat(live-session): keep prompt runs alive across panel close and reopen.
- Il modifie **187 fichiers**, avec **22 033 insertions** et **11 717 suppressions**.
- Les six programming docs du périmètre ont été lus.
- Tous les chemins planifiés dans la spécification sont représentés dans l'implémentation ; des fichiers générés et des tests de renforcement ont également été ajoutés.
- Les anciens contrats recherchés — canAbortSource, queryState.windows, SidebarActivityQueryWindowItem et streamKey — ne subsistent plus.
- Le contrôle Git du commit ne remonte pas d'erreur de whitespace.

1) La spécification entière est-elle implémentée ?

Réponse

 Pas intégralement au sens strict.

Mesure	Estimation	Verdict
Couverture structurelle des fichiers et sous-systèmes	100 %	 Tous les blocs architecturaux sont présents
Couverture fonctionnelle principale	97 %	⚠ Deux comportements ne respectent pas complètement la spécification
Couverture des scénarios de test	96 %	⚠ Très élevée, mais les deux écarts ci-dessous ne

Mesure	Estimation	Verdict
prévus		sont pas verrouillés correctement
Conformité globale stricte	97 %	✗ Pas encore 100 % conforme
La fonctionnalité principale est réellement implémentée : une génération Prompt normale peut survivre à la fermeture du panneau, apparaître dans Activité, être rouverte, rejouer son historique et converger vers un handoff.		
Deux écarts empêchent cependant de déclarer la spécification entièrement terminée :		
1. les requêtes Prompt normales de remplacement ne deviennent pas replayables dès la baseline ;		
2. l'autorité du bouton Stop peut rester visible après certaines preuves terminales.		
Conclusion de la question 1		
La tranche est presque entièrement livrée, mais la réponse stricte est non : deux corrections fonctionnelles restent nécessaires.		

2) L'implémentation fait-elle ce qu'elle est censée faire pour chaque fonctionnalité technique demandée ?

Les pourcentages ci-dessous sont des **estimations de conformité au contrat**, et non une métrique de couverture automatisée.

Bloc technique demandé	État	Conformité	
Livraison atomique de la tranche	✓	100 %	Les contrats, backend, transportés dans le même commit.
Identité exacte { locator, turnId, sourceId }	✓	100 %	LiveSessionRun.type.ts indépendante.
sourceId transporté au backend sans identifiants frontend-only	✓	100 %	MessageRegistry.ts transporté.
Union stricte des descripteurs EasyAgents/normalPrompt	✓	100 %	LiveSessionPanel.type.ts et sourceId===turnId pour la validation.
Rejet des paramètres inconnus, partiels ou dupliqués	✓	100 %	La validation est dupliquée volontairement dans PromptLiveSessionPanel.type.ts .
Enveloppe Open sélective et résultat Open complet	✓	100 %	LiveSessionOpenResult.type.ts session.
Replay	✓	100 %	Le contrat est centralisé dans LiveSession.type.ts .
modes replaceTurnAssistants / continueAssistant	✓	100 %	
Fabriques SYSTEM_ERROR/INTERRUPTED et protocole V1	✓	100 %	aiStopSuffix.util.ts pour la gestion de l'erreur.
Détection d'une erreur Slimbk dans le body après header 200	✓	100 %	parseSlimbkStreamBodyError.ts
Persistance de tout meta.error avant append	✓	100 %	NormalPromptLiveSession.type.ts forwarding.
Suffixe terminal Orch idempotent	✓	100 %	OrchMessagesMemoryAssistance.ts dupliquer le suffixe.
Réservation synchrone avant tout await	✓	100 %	LiveSessionRegistry.service.ts préparation asynchrone.
Promotion exacte et refus des réservations dupliquées	✓	100 %	Le handle opaque est consommé pour éviter de voler la réservation.
Époques, retraites et tombstones process-lifetime	✓	99 %	Les structures et transitions sont correctement gérées.
Claims destructifs et génération destructive	✓	100 %	Acquire/release invalide les OpenRequests et le claim.
Wake-ups profondément immuables et listeners isolés	✓	100 %	Les événements sont figés après la réception du résultat.
Finalisation post-forwarding en cinq résultats	✓	100 %	La priorité retiredMissing est gérée et le résultat stale sont présents.
Open live-safe, filtrage du tour et terminal snapshot	✓	99 %	LiveSessionOpen.service.ts
Lifecycle lazy Readable partagé	✓	100 %	LiveSessionSourceLifecycle.ts déterministement.
Source Prompt normale, baseline et synthèse terminale	✓	98 %	Le pipeline durable-before-append est correct mais cependant incorrecte pour les requêtes normales.
Replay disponible dès la baseline pour Prompt normal	✗	70 %	continueResponse.type.ts utilise broadcast.

Bloc technique demandé	État	Conformité
Contexte de webview émettrice exact	✓	100 %
Guard ancien panneau → panneau rouvert	✓	100 %
FIFO extension unique, protocole V1 et erreur post-header	✓	99 %
Aucun second timer Activité ni second son au replay source	✓	100 %
Son exact par runRef et <code>soundAttemptId</code>	✓	100 %
Coordinateur de panneau injecté, sans import du low-level opener	✓	100 %
Suppression synchrone du panneau avant dispose coordonné	✓	100 %
Activité Idle/open, observer/open, source/open ou closed	✓	100 %
Item conservé fermé tant que la source est active	✓	100 %
Idle si la source termine panneau ouvert	✓	100 %
Troisième ligne Stop pour source fermée	✓	100 %
Réouverture depuis Activité	✓	100 %
Stop source exact avant ou après promotion	✓	100 %
Propagation synchrone de <code>stopRequested</code>	✓	100 %
Delete sous claim et attente de la source	✓	99 %
Retry EasyAgents sous claim	✓	99 %
Bootstrap React externe pour les queries invalides	✓	100 %
Snapshot brut figé	✓	100 %
Désactivation hydratation classique et auto-run	✓	100 %
<code>start()</code> synchrone, single-flight, IDs après acceptation	✓	100 %
Plans purs puis scaffold après acceptation	✓	100 %
Guards <code>sourceId</code> + <code>streamInstanceId</code> Queue A/B	✓	99 %
Publication outcome après Queue A, Queue B et commit	✓	100 %
Terminal data puis <code>transportError</code> sans doublon	✓	100 %
Finalisation frontend même sans chunk visible	✓	100 %
<code>preserveLocal</code> sans lecture draft/attachments/nextTurnId	✓	100 %
Handoff normalPrompt direct, sans second Open	✓	100 %
Autorité Stop retirée après toute preuve terminale	⚠	78 %
Composer éditable après armement, Send verrouillé	✓	100 %
Stop unique dans le compositeur	✓	100 %
Retry banner normalPrompt sans Stop	✓	100 %
Fallback ack corrélé { <code>turnId</code> , <code>streamInstanceId</code> }	✓	100 %
Finalisation sans ack : démasquage sans perte	✓	100 %
Continuation/thinking par tentative exacte	✓	99 %

token ou raw-live supplémentaire
[extTypedMessageBus.uti](#)
sérialisé.
Les deux handlers comparent l
ou le replay.
[iaStreamHandlers.ts](#) et
métier et conservent l'ordre.
Le replay d'une source normal
supplémentaire.
[PromptQueryDoneSound.m](#)
[PromptLiveSessionPanel](#)
locator.
[openPanel.util.ts](#) retire c
L'union stricte et les transition
La fermeture transforme uniqu
projections sont retirées.
La transition se fait même pou
[SidebarActivityQuery.t](#)
same-stack.
Le RPC ouvre la session par lo
L'endpoint et le service détach
Le runtime et le store sont mut
désactivé.
[delete.endpoint.ts](#) abort
exact.
[EasyAgentsBatchRunRetr](#)
suppression durable.
[Prompt.tsx](#) ne monte pas le
[usePromptSessionLocator](#)
Toute query live explicite désa
[useAiStream.hook.ts](#) refu
génère `streamInstanceId` e
Send, SPECIF, Retry, Edit, Co
dans [usePromptRunCommand](#)
Les algorithmes de coalescenc
ont été ajoutées autour du pipe
completed ou `transportEr`
Le terminal déjà consommé re
[usePromptRunController](#)
compris les transports vides.
[applyHydratedPromptSes](#)
Le handoff attend terminal, co
Correct pour terminal réussi, n
terminal + `transportError`
Le latch `replayComposerAr`
La bannière normalPrompt n'a
Le modèle de bannière est sépa
Le clear ne se fait qu'après l'a
Texte, attachments, `nextTurn`
[preparePromptContinuat](#)

Bloc technique demandé	État	Conformité
Préservation d'EasyAgents	✓ 99 %	propre à la tentative. sourceId=turnId, observer sont conservés.
Tests et programming docs	⚠ 96 %	La couverture est très importan normalPrompt ne vérifie le ret

Variations d'implémentation acceptables

Deux différences littérales ne constituent pas des défauts :

- L'identité appelée `replayOpenAttemptId` dans la spécification est implémentée sous la forme `openAttemptRef` avec un `attemptId` frais par Open. Le comportement demandé est présent.
- Le Stop Activité n'utilise pas littéralement `disableAfterClick`; il emploie un ref et un état local same-stack. Cette solution est **plus robuste**, car elle permet de réactiver le bouton si le backend refuse la demande.

Conclusion de la question 2

Le comportement central est opérationnel et très largement conforme, mais la disponibilité de replay trop tardive et la conservation incorrecte de l'autorité Stop empêchent une conformité fonctionnelle complète.

3) Y a-t-il des bugs ?

Bug confirmé 1 — Le replay normalPrompt de remplacement attend trop longtemps

Sévérité : ⚠ moyenne

La spécification exige que Prompt normal devienne replayable **dès que sa baseline durable est disponible**.

L'implémentation applique néanmoins `baselineAndObservableChunk` à toutes les requêtes autres que `continueResponse` :

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

Copy

```
return aiQuery.type === 'continueResponse'
  ? Object.freeze({
    replayMode: 'continueAssistant',
    readiness: 'baselineOnly',
  })
  : Object.freeze({
    replayMode: 'replaceTurnAssistants',
    readiness: 'baselineAndObservableChunk',
  });
```

Or, le registre considère comme observable uniquement :

- un token texte ;
- un thinking token ;
- un `raw_live_received`.

`fallback_persisted` installe bien la baseline, mais ne rend donc pas le run replayable.

Scénario observable

1. L'utilisateur envoie un Prompt.
2. `fallback_persisted` est reçu et la baseline est installée.
3. Le modèle reste silencieux plusieurs secondes ou minutes avant son premier token.
4. Le panneau est fermé puis rouvert depuis Activité.
5. Open continue de retourner `not_ready`.
6. Le brouillon et les attachments restent verrouillés alors que la baseline est déjà sûre.

Le test dans `NormalPromptLiveSessionSource.service.test.ts` affirme actuellement ce comportement divergent au lieu de vérifier la spécification.

Correction nécessaire

- Utiliser `readiness: 'baselineOnly'` pour **toutes** les sources Prompt normales.
- Conserver `baselineAndObservableChunk` uniquement pour la politique EasyAgents historique.
- Ajouter un test prouvant qu'un Open normalPrompt devient `live` après `fallback_persisted`, même sans token, thinking token ou raw-live.

Bug confirmé 2 — Le Stop peut rester affiché après la fin certaine de la source

Sévérité : ⚠ moyenne à faible

Pour `normalPrompt`, l'effet `transportError` conserve systématiquement l'autorité `Stop` :

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

Copy

```
// Transport failure is viewer-local evidence only. The source may still be
running in the
// extension, so preserve both its Stop authority and any abort RPC already in
flight.
params.setPromptRuntimeMode(
  params.bootstrap.presentationAdapter.handoffPolicy ===
    'directAfterReplay'
    ? 'liveSessionReplayFailed'
    : 'liveSessionHandoffFailed',
);
```

C'est correct lorsqu'aucun terminal n'a été reçu : l'échec peut effectivement être uniquement local au viewer.

Ce n'est plus correct lorsque `terminalDataSeen === true` :

- le terminal a été consommé ;
- le handler backend a terminé son forwarding ;
- la source a été finalisée ou est sur le point d'être finalisée ;
- aucun abort utile ne reste possible.

L'écran passe pourtant en `liveSessionReplayFailed` avec `sourceStopAvailable=true`. Le textarea continue alors d'afficher **Stop** pour une source terminée et chaque tentative est refusée.

Un second cas existe pour une réponse `Open terminal` dont l'hydratation échoue. L'autorité est retirée seulement **après** `hydrate(...)` :

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

Copy

```
const parsed = hydrate(response, snapshotMode);
if (parsed.kind === 'terminal') {
  const latest = paramsRef.current;
  // Terminal hydration completes the live bootstrap atomically
  with composer ownership.
  setReplayComposerArmed(false);
  retireSourceStopAuthority();
  latest.setPromptRuntimeMode('normal');
  latest.setHandoffFailureMessage(undefined);
  return;
}
```

Si l'hydratation terminale lève une erreur, `retireSourceStopAuthority()` n'est jamais atteint, alors que l'enveloppe terminale a déjà prouvé la fin du backend.

Correction nécessaire

- Retirer l'autorité source `Stop` dès qu'une enveloppe `Open` exacte est classifiée `terminal`, avant l'hydratation de son contenu.
- Lors d'un `transportError`, retirer également l'autorité si `terminalDataSeen === true`.
- La conserver uniquement pour un échec viewer sans preuve terminale.
- Ajouter les deux tests `normalPrompt` correspondants.

Écarts de maintenance non bloquants

Commentaires devenus contradictoires

Dans [usePromptRunCommands.hook.ts](#), `onBeforeSubmitUi` est volontairement devenu un no-op, mais sa documentation parle toujours d'un « flip pending immédiat ».

Le comportement actuel est correct — les mutations appartiennent aux commandes après acceptation — mais le commentaire peut induire une future régression.

Callback React recréé dans la liste Activité

Dans `SidebarActivityQuery.tsx`, la cible du lien est créée inline avec une fermeture sur l'item. Ce n'est pas un bug visible, mais cela ne suit pas totalement la règle locale des callbacks React stables.

Conclusion de la question 3

J'ai identifié deux bugs fonctionnels confirmés, aucun défaut critique de perte de données, et deux points mineurs de maintenance.

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

Avis sur la spécification — 96 %

La spécification est excellente sur les sujets difficiles :

- identité logique contre identité d'exécution ;
- protection ABA ;
- exactitude de webview ;
- réservation avant `await` ;
- terminal durable avant replay ;
- séparation terminal data / erreur transport ;
- claims destructifs ;
- transitions Activité ;
- absence de duplication son/timer ;
- isomorphisme stream/replay ;
- restauration transactionnelle du compositeur.

Son principal défaut est sa taille : **160 fichiers prévus dans une tranche atomique** rendent la vérification humaine extrêmement coûteuse. Le risque de régression annoncé à 92 % était donc réaliste, et non pessimiste.

Avis sur le code source — 93 %

Points particulièrement réussis

- Les identités sont transportées jusqu'aux continuations asynchrones.
- Les claims et handles opaques réduisent fortement les opérations locator-only dangereuses.
- Le backend source, le lifecycle générique, le registre, le coordinateur et l'adaptation produit sont bien séparés.
- La persistance de `meta.error` avant `append` ferme une vraie fenêtre de corruption.
- L'erreur `Slimbk post-header` n'est plus confondue avec un chunk métier.
- La politique `preserveLocal` évite réellement la lecture des champs draft.
- La finalisation frontend couvre même les transports sans chunk.
- Les tests sont nombreux, ciblés et décrivent généralement de vraies invariants concurrentes.
- Les commentaires techniques sont détaillés et utiles.

Limites

- Plusieurs modules restent très volumineux et demandent une forte charge cognitive.
- Certaines machines à états sont distribuées entre refs synchrones, état React, registre backend et store Activité : c'est justifié, mais difficile à maintenir.
- Le test de readiness `normalPrompt` a suivi l'implémentation plutôt que le contrat, ce qui montre qu'une suite de tests abondante ne protège pas contre une mauvaise exigence encodée.
- La distinction « transport viewer-local » contre « transport avec terminal déjà observé » n'est pas suffisamment matérialisée dans le hook.

Conclusion de la question 4

L'architecture est production-grade et soigneusement défensive, mais deux décisions locales contredisent encore des invariants explicitement formulés par la spécification.

5) Résumé du rapport

Sujet	Résultat	Estimation	Conclusion
Spécification entièrement implémentée		97 %	Deux corrections fonctionnelles restent
Architecture générale liveSession		99 %	Très robuste
Fermeture et continuation de source		99 %	Fonctionnement principal présent
Réouverture et replay exact		95 %	Replay trop tardif pour les requêtes replace
Exactitude des identités et protections ABA		99 %	Très bonne

Sujet	Résultat	Estimation	Conclusion
Persistance terminale et protocole Slimbk	✓	100 %	Conforme
Activité et Stop détaché	✓	98 %	Projection correcte
Autorité Stop dans le panneau reconnecté	⚠	90 %	Peut survivre à une preuve terminale
Delete et Retry destructifs	✓	98 %	Claims et barrières correctement placés
Fallback persist et conservation du compositeur	✓	100 %	Conforme
Continuation et thinking	✓	99 %	Conforme
Préservation EasyAgents	✓	99 %	Comportement historique maintenu
Tests	⚠	96 %	Très complets, mais deux scénarios manquent ou affirment le mauvais contrat
Documentation	✓	98 %	Riche et cohérente
Bugs critiques	✓	0 identifié	Aucun défaut manifeste de perte ou duplication durable
Bugs fonctionnels confirmés	⚠	2	Readiness et autorité Stop
Défauts mineurs de maintenance	●	2	Commentaire et callback React

6) Conclusion en une phrase

La tranche liveSession Prompt normale est techniquement solide et presque complète, mais elle ne doit pas être considérée comme totalement conforme avant de rendre toutes les sources Prompt replayables dès leur baseline et de retirer systématiquement le Stop lorsqu'une fin terminale est certaine.

7) Ensembles de corrections proposés

Ensemble A — Readiness normalPrompt prioritaire

- Passer toutes les requêtes Prompt normales à `baselineOnly`.
- Conserver la readiness historique EasyAgents inchangée.
- Corriger le test de mapping.
- Ajouter un test de réouverture après baseline, avant tout premier token.

Ensemble B — Autorité Stop terminale

- Retirer l'autorité Stop avant l'hydratation d'une branche Open `terminal`.
- La retirer après `terminalDataSeen` + `transportError`.
- Conserver l'autorité uniquement pour les erreurs viewer sans terminal.
- Ajouter les tests de hook et de modèle visuel correspondants.

Ensemble C — Polissage de maintenance

- Aligner les commentaires de `onBeforeSubmitUi` avec son comportement transactionnel réel.
- Supprimer ou renommer le seam devenu trompeur si son retrait reste local.
- Extraire la ligne Activité dans un sous-composant afin de stabiliser son callback d'ouverture.

Ensemble D — Tous les correctifs

- Appliquer les ensembles A, B et C.
- Conserver tous les contrats EasyAgents, Queue A/B, son, Activity et fallback existants.
- Ajouter les régressions ciblées sans élargir le périmètre fonctionnel.



Coder le correctif readiness baseline



Coder le correctif autorité Stop



Coder le polissage maintenance



Coder tous les correctifs

</ASSISTANT>

<USER>

[Coder tous les correctifs]

</USER>

<ASSISTANT>

I'm thinking...

</ASSISTANT>

<ASSISTANT>

  reading project:  [liveSession.programming-doc-readme.md](#),
 [NormalPromptLiveSessionSource.service.ts](#),  [NormalPromptLiveSessionSource.service.test.ts](#),  [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#),
 [usePromptLiveSession.hook.ts](#),
 [usePromptLiveSession.normalPrompt.test.tsx](#),
 [usePromptLiveSession.handoff.test.tsx](#),
 [usePromptRunCommands.hook.ts](#),
 [Prompt.tsx](#),
 [SidebarActivityQuery.tsx](#),  [SidebarActivityQuery.test.tsx](#),  [SidebarActivityIndex.tsx](#),  [VscLink.tsx](#),  e ["Seam avant soumission"](#),  e ["Readiness source Prompt"](#),  e ["Autorité Stop retrait"](#),  [usePromptIndexPending.hook.tsx](#),  [usePromptRunController.hook.ts](#) [L560-L697],  e ["Callback gate index"](#),  e ["Prop avant submit type"](#),  [PromptInput.tsx](#) [L80-L190],  [PromptInput.tsx](#) [L900-L1010],  [PromptInput.tsx](#) [L1010-L1095],  e ["Policy source Prompt usages"](#),  [iaStreamHandlers.ts](#) [L55-L125],  [LiveSessionOpenResult.type.ts](#),  e ["Chunk fallback persisted"](#),  e ["Controller seam commande"](#),  e ["Appel commandes Prompt"](#),  [usePromptRunController.hook.ts](#) [L450-L515]

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

Copy

Review

Accept

PatchFull file

```
-  
- /** Map every normal Prompt query to its exact replay reconstruction policy. */  
-  
+  
+ /**  
+  * Map every normal Prompt query to its exact replay reconstruction policy.  
+  *  
+  * Normal Prompt always becomes replayable as soon as its durable baseline exists. The  
+  * observer can  
+  * safely attach before the first token because the registry retains the complete  
+  * ordered tail and  
+  * the frontend keeps Send locked until replay/handoff converges. Observable-chunk  
+  * readiness remains  
+  * an EasyAgents product policy and must not delay restoration of an interactive Prompt  
+  * composer.  
+  */  
+  
- ): NormalPromptLiveSessionReplayPolicy {  
-   return aiQuery.type === 'continueResponse'  
-     ? Object.freeze({  
-       replayMode: 'continueAssistant',  
-       readiness: 'baselineOnly',  
-     })  
-     : Object.freeze({  
-       replayMode: 'replaceTurnAssistants',  
-       readiness: 'baselineAndObservableChunk',  
-     });  
+ ): NormalPromptLiveSessionReplayPolicy {  
+   // Continuation overlays the immutable source-start messages, while every other  
+   // query replaces  
+   // assistants from the exact logical turn. Both modes are safe immediately after  
+   // baseline install.  
+   const replayMode: LiveSessionReplayMode =
```

```

+         aiQuery.type === 'continueResponse'
+         ? 'continueAssistant'
+         : 'replaceTurnAssistants';
+     return Object.freeze({ replayMode, readiness: 'baselineOnly' });
+

```

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

Copy

Review

Accept



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

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

Copy

Review

Accept

PatchFull file

```

-
-     it('maps continuation separately from every replace-style query', () => {
-
+
+     it('maps replay reconstruction separately while every normal Prompt source is
baseline-ready', () => {
+
+
+         replayMode: 'replaceTurnAssistants',
+         readiness: 'baselineAndObservableChunk',
+
+         replayMode: 'replaceTurnAssistants',
+         readiness: 'baselineOnly',
+
+
+     }
+ });
+
+     }
+ });
+
+     it('installs a replace baseline before forwarding any observable source output',
async () => {
+         const order: string[] = [];
+         const durableAssistant = new AssistantMessage(RUN.turnId, '');
+         const getThreadMessagesSnapshot = jest
+             .fn<() => AssistantMessage[]>()
+             .mockReturnValueOnce([])
+             .mockReturnValueOnce([durableAssistant]);
+         const markBaselineReady = jest.fn(() => {
+             order.push('baseline');
+             return { becameReplayReady: true };
+         });
+         const fallbackPersisted: AiStreamedChunk = {
+             turnId: RUN.turnId,
+             type: 'meta',
+             value: { metaType: 'fallback_persisted', value: true },
+         };
+         const terminal: AiStreamedChunk = {
+             turnId: RUN.turnId,
+             type: 'meta',
+             value: {
+                 metaType: 'stop_reason',
+                 value: { reason: AiStopReason.END_TURN },
+             },
+         };
+

```

```

+     const service = new NormalPromptLiveSessionSourceService({
+       liveSession: {
+         registerRun: () => {
+           order.push('register');
+           return { abortRequested: false };
+         },
+         markBaselineReady,
+         appendChunk: () => {
+           order.push('append');
+           return { becameReplayReady: false };
+         },
+       },
+       lifecycle: new LiveSessionSourceLifecycleService(),
+       orchStream: {
+         askStream: () =>
+           Readable.from([
+             encode(fallbackPersisted),
+             encode(terminal),
+           ]),
+         abortStream: () => {},
+         memory: {
+           getThreadMessagesSnapshot,
+           patchLastAssistantMeta: async () => {},
+         },
+       },
+       parseChunk: (raw) => JSON.parse(raw) as AiStreamedChunk,
+     });
+
+     await service.execute({
+       reservation: RESERVATION,
+       runRef: RUN,
+       aiCatalog: { engines: [] },
+       aiQuery: { type: 'prompt', promptValue: 'hello' },
+       aiQueryOptions: {},
+       onChunk: async () => {
+         order.push('forward');
+       },
+     });
+
+     // fallback_persisted is not observable source output, so baselineOnly must
make the run
+     // replayable before either the fallback marker or the later terminal reaches
the viewer.
+     expect(order).toEqual([
+       'register',
+       'baseline',
+       'append',
+       'forward',
+       'append',
+       'forward',
+     ]);
+     expect(markBaselineReady).toHaveBeenCalledWith({
+       runRef: RUN,
+       baselineMessages: [durableAssistant],
+     });
+   });
+
+

```

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

Copy
Review
Accept

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

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

Copy

Review

Accept

PatchFull file

```
-     ZLiveSessionOpenResultEnvelope,
-
+     ZLiveSessionOpenResultEnvelope,
+     type LiveSessionOpenResultEnvelope,
+


---


-
- /** Generic Open/replay state machine guarded by independent Open-attempt and stream
identities. */
-
+
+ /** Open branches that carry an exact source identity and one session snapshot. */
+ type LiveSessionSessionEnvelope = Extract<
+   LiveSessionOpenResultEnvelope,
+   { kind: 'live' | 'terminal' }
+ >;
+
+ /** Generic Open/replay state machine guarded by independent Open-attempt and stream
identities. */
+


---


-
-   const hydrate = useCallback(
-     (response: unknown, mode: 'hydrateFromSnapshot' | 'preserveLocal') => {
-
+
+   /** Parse one Open response and reject any session-bearing branch for a different
exact source. */
+   const parseExactOpenEnvelope = useCallback(
+     (response: unknown): LiveSessionOpenResultEnvelope => {
+


---


-       const envelope = ZLiveSessionOpenResultEnvelope.parse(response);
-       if (
-         !current.bootstrap ||
-         (envelope.kind !== 'live' && envelope.kind !== 'terminal') ||
-         envelope.threadId !==
-           current.bootstrap.runRef.locator.sessionId ||
-         envelope.turnId !== current.bootstrap.runRef.turnId ||
-         envelope.sourceId !== current.bootstrap.runRef.sourceId
-       ) {
-         throw new Error(
-           'Live-session Open returned a different exact source.',
-         );
-
+       const envelope = ZLiveSessionOpenResultEnvelope.parse(response);
+
+       // Availability-only branches carry no identity. Every snapshot-bearing
branch must match
+       // before it is allowed to retire Stop authority or mutate hydrated Prompt
state.
+       if (envelope.kind === 'live' || envelope.kind === 'terminal') {
+         if (
+           !current.bootstrap ||
+           envelope.threadId !==
+             current.bootstrap.runRef.locator.sessionId ||
+           envelope.turnId !== current.bootstrap.runRef.turnId ||
+           envelope.sourceId !== current.bootstrap.runRef.sourceId
```

```

+         ) {
+             throw new Error(
+                 'Live-session Open returned a different exact source.',
+             );
+         }
+
-     }
-     const parsed =
-
+     }
+     return envelope;
+ },
+ [],
+ );
+
+ const hydrate = useCallback(
+     (
+         response: unknown,
+         envelope: LiveSessionSessionEnvelope,
+         mode: 'hydrateFromSnapshot' | 'preserveLocal',
+     ) => {
+         const current = paramsRef.current;
+         const parsed =
+
-     ) {
-         // A transport or parsing failure does not prove that the backend
source disappeared.
-         // Keep source-Stop authority intact so the operator can still abort
the exact source.
-
+     ) {
+         // Failures without an authoritative terminal leave the extension
source potentially active.
+         // Terminal branches retire authority before hydration, so this path
preserves only valid Stop state.
+
-         if (!ownsOpenAttempt(attemptId)) return;
-         const envelope =
-             ZLiveSessionOpenResultEnvelope.parse(response);
-
+         if (!ownsOpenAttempt(attemptId)) return;
+         const envelope = parseExactOpenEnvelope(response);
+
-     }
-     const parsed = hydrate(response, snapshotMode);
-
+     }
+     if (envelope.kind === 'terminal') {
+         // Exact terminal identity is authoritative even if full
snapshot validation or
+         // hydration fails. Retire Stop before consuming session
content so a malformed
+         // terminal snapshot cannot leave an abort button targeting a
finished source.
+         retireSourceStopAuthority();
+     }
+     const parsed = hydrate(response, envelope, snapshotMode);
+
-         setReplayComposerArmed(false);
-         retireSourceStopAuthority();

```

```

-
+                 setReplayComposerArmed(false);
+
-
+                 ownsOpenAttempt,
-
+                 ownsOpenAttempt,
+                 parseExactOpenEnvelope,
+
-
+             );
-             // Transport failure is viewer-local evidence only. The source may still be
running in the
-             // extension, so preserve both its Stop authority and any abort RPC already in
flight.
-
+             );
+             const ownsDirectSource =
+                 params.bootstrap.presentationAdapter.handoffPolicy ===
+                 'directAfterReplay';
+             if (ownsDirectSource && params.terminalDataSeen) {
+                 // A consumed true terminal is authoritative even when a later
transport/auxiliary failure
+                 // classifies the attempt as transportError. No useful source abort can
remain after that proof.
+                 retireSourceStopAuthority();
+             }
+             // Without terminal data, transport failure remains viewer-local evidence and
the exact extension
+             // source may still be active. Preserve Stop authority only for that genuinely
uncertain branch.
+
-             // source may still be active. Preserve Stop authority only for that genuinely
uncertain branch.
-             params.setPromptRuntimeMode(
-                 params.bootstrap.presentationAdapter.handoffPolicy ===
-                 'directAfterReplay'
-
+             // source may still be active. Preserve Stop authority only for that genuinely
uncertain branch.
+             params.setPromptRuntimeMode(
+                 ownsDirectSource
+
-             );
-             }, [params]);
-
+             );
+             }, [params, retireSourceStopAuthority]);
+

```

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

Copy

Review

Accept



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

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

Copy

Review

Accept

PatchFull file

```

-     jest.fn<(path: string, args: unknown) => Promise<unknown>>());

```

```

-
+   jest.fn<(path: string, args: unknown) => Promise<unknown>>());
+   const applyHydratedPromptSessionStateMock =
+     jest.fn<(args: unknown) => void>();
+
-   ' ../../Prompt/applyHydratedPromptSessionState.util.js',
-   () => ({ applyHydratedPromptSessionState: () => {} })),
-
+   ' ../../Prompt/applyHydratedPromptSessionState.util.js',
+   () => ({
+     applyHydratedPromptSessionState: applyHydratedPromptSessionStateMock,
+   })),
+
-   callLocalApiMock.mockReset();
-
+   callLocalApiMock.mockReset();
+   applyHydratedPromptSessionStateMock.mockReset();
+   applyHydratedPromptSessionStateMock.mockImplementation(() => {});
+
-
-   it('retries not_ready with one cancellable timer and starts the exact observer only
after live hydration', async () => {
-
+
+   it('retires source Stop before an exact terminal snapshot hydration failure is
exposed', async () => {
+     applyHydratedPromptSessionStateMock.mockImplementationOnce(() => {
+       throw new Error('terminal hydration failed');
+     });
+     callLocalApiMock.mockResolvedValueOnce({
+       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,
+     });
+
+     const { result } = renderHook(() =>
+       useNormalPromptLiveSessionHarness(),
+     );
+
+     await act(async () => {
+       await Promise.resolve();
+       await Promise.resolve();
+     });
+
+     expect(result.current.mode).toBe('liveSessionReplayFailed');
+     expect(result.current.failure).toContain('terminal hydration failed');
+     expect(result.current.api.sourceStopAvailable).toBe(false);
+     expect(result.current.viewModel.transportStopAvailable).toBe(false);
+   });
+

```

```
+ it('retries not_ready with one cancellable timer and starts the exact observer only
after live hydration', async () => {
```

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

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

```
- });
-
- it('ignores a completed-without-terminal outcome from a superseded observer
identity', async () => {
-
+ });
+
+ it('retires normalPrompt source Stop after terminal data followed by transport
failure', async () => {
+   const { result, rerender } = renderHook(
+     (signals: HandoffSignals) => useHandoffHarness(signals),
+     { initialProps: buildInitialHandoffSignals() },
+   );
+   await waitFor(() =>
+     expect(result.current.mode).toBe('liveSessionStreaming'),
+   );
+   expect(result.current.api.sourceStopAvailable).toBe(true);
+
+   await act(async () => {
+     rerender({
+       terminalDataSeen: true,
+       streamOutcome: 'transportError',
+       streamInstanceId: 'stream-observer',
+       finalizedStreamInstanceId: 'stream-observer',
+     });
+   });
+
+   await waitFor(() =>
+     expect(result.current.mode).toBe('liveSessionReplayFailed'),
+   );
+   expect(result.current.api.sourceStopAvailable).toBe(false);
+   expect(result.current.api.sourceStopPending).toBe(false);
+   expect(callLocalApiMock).toHaveBeenCalledTimes(1);
+ });
+
+ it('ignores a completed-without-terminal outcome from a superseded observer
identity', async () => {
```

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

Copy

Review

Accept

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

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

Copy
Review
Accept

PatchFull file

```
-      handleRetryQuery: () => AiStreamStartResult;
-      onBeforeSubmitUi: () => void;
-
+      handleRetryQuery: () => AiStreamStartResult;
+
-
-      /** Pre-submit UI hook (flips pending immediately) */
-      const onBeforeSubmitUi = useCallback(): void => {
-        // Command handlers own the pending transition only after synchronous start
acceptance.
-      }, []);
-
-
+
-      handleRetryQuery,
-      onBeforeSubmitUi,
-
+      handleRetryQuery,
```

[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

```
-      handleRetryQuery,
-      onBeforeSubmitUi,
-
+      handleRetryQuery,
```

```
-      isRunningForInput,
-      onBeforeSubmitUi,
-
+      isRunningForInput,
```

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

Copy
Review
Accept

  patching file:  [Prompt.tsx](#)

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

Copy
Review
Accept
PatchFull file

```
-      isRunningForInput,
```

```

-         onBeforeSubmitUi,
-
+         isRunningForInput,
+
-
-
-  /**
-   * Install the indexing pending gate (buffers any run-triggering action while
indexing is running).
-   * Centralizes gating for Send/Reply/Retry/Continue and exposes force/close for the
notice UI.
-   * The UI lock is engaged immediately before a buffered action is replayed.
-
+  /**
+   * Install the indexing pending gate (buffers any run-triggering action while
indexing is running).
+   * Centralizes gating for Send/Reply/Retry/Continue and exposes force/close for the
notice UI.
+   * The gate owns only buffering/notice state; accepted command handlers own
transactional pending state.
+
-
-  */
-  const { gateAction, isIndexPending, forceNow, closeNotice } =
-    usePromptIndexPending({
-      onWillExecuteBufferedAction: onBeforeSubmitUi,
-    });
-
+  */
+  const { gateAction, isIndexPending, forceNow, closeNotice } =
+    usePromptIndexPending();
+
-
-      stopEnabled={!effectiveAbortAckPending}
-      /* Instant UI flip before delegating submit up */
-      onBeforeSubmitUi={onBeforeSubmitUi}
-
+      stopEnabled={!effectiveAbortAckPending}
+
-
-      effectiveAbortAckPending,
-      onBeforeSubmitUi,
-
+      effectiveAbortAckPending,
+

```

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

Copy

Review

Accept



patching file: [PromptInput.tsx](#)

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

Copy

Review

Accept

PatchFull file

```

-
-  /** UI hook called immediately before delegating submit to parent (for instant
flip). */
-  onBeforeSubmitUi: () => void;
-
+
+  /**
+   * Optional pre-delegation notification.

```

```

+      *
+      * This callback cannot assume that the stream will be accepted and must never arm
transactional
+      * pending state. The parent command owns all visible run mutations after
synchronous acceptance.
+      */
+      onBeforeSubmitUi?: () => void;
+
-
-      // Continue with the normal send flow.
-      onBeforeSubmitUi();
-
+
+      // Notify the optional host immediately before
delegation without assuming acceptance.
+      onBeforeSubmitUi?();
+
-
-      onBeforeSubmitUi();
-
+
+      // Notify the optional host immediately before
delegation without assuming acceptance.
+      onBeforeSubmitUi?();
+
-
-      // Request an immediate UI flip to Stop (controlled at parent level)
-      onBeforeSubmitUi();
-
+
+      // Notify the optional host; the accepted command still owns every run-
state mutation.
+      onBeforeSubmitUi?();
+

```

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

Copy

Review

Accept

  **patching file:**  [SidebarActivityQuery.tsx](#)

[AI Code/src/frontend/src/app/sidebar/sidebar-activity/SidebarActivityQuery.tsx](#)

Copy

Review

Accept

PatchFull file

```

- import {
-   Fragment,
-
+ import {
+
-
- </VscVertStackRow>
-   );
- }
-
+ </VscVertStackRow>
+   );
+ }
+

```

```

+ /** One activity projection with a stable locator callback and optional detached-source
+ Stop subtree. */
+ function SidebarActivityQueryItemRow({
+   item,
+   selected,
+ }): {
+   item: SidebarActivityQueryItem;
+   selected: boolean;
+ }): JSX.Element {
+   const running = item.status === 'running';
+   const dotStatus: VscStatusDotStatus = running ? 'warning' : 'success';
+   const timeText = running
+     ? `Elapsed: ${formatDurationHms(item.liveQueryElapsedMs)}`
+     : item.lastQueryElapsedMs === undefined
+       ? '-'
+       : `Last query: ${formatDurationHms(item.lastQueryElapsedMs)}`;
+   const displayTitle = formatUserPromptTitleForUi(item.title);
+   const locator = useMemo(
+     () => ({
+       sessionId: item.sessionId,
+       storageScopeRelPath: item.storageScopeRelPath,
+     }),
+     [item.sessionId, item.storageScopeRelPath],
+   );
+
+   /** Open the exact session activity without recreating the VscLink callback on
+   parent rerenders. */
+   const handleOpen = useCallback(() : void => {
+     void bus.call('sidebar->ext:openQueryActivity', { locator });
+   }, [locator]);
+
+   // Preserve the strict detached-source subtype through JSX so the Stop subtree
+   cannot accidentally
+   // accept an idle, observer-owned, or reopened activity.
+   const detachedSource = isDetachedSourceItem(item) ? item : undefined;
+
+   return (
+     <>
+       <VscVertStackRow>
+         <VscHorGroupLayout allowShrink>
+           <VscStatusDot status={dotStatus} />
+           <VscLabel text={running ? 'Running' : 'Idle'} />
+           <VscLink
+             text={displayTitle}
+             target={handleOpen}
+             selected={selected}
+             singleLineWithEllipsis
+             tooltipText={displayTitle}
+             tooltipOnlyWhenEllipses
+           />
+         </VscHorGroupLayout>
+       </VscVertStackRow>
+       <VscVertStackRow overrideSpacingTop="never">
+         <VscLabel text={timeText} />
+       </VscVertStackRow>
+       {detachedSource ? (
+         <DetachedSourceStop
+           key={detachedSource.runRef.sourceId}
+           item={detachedSource}
+         />
+       ) : null}
+     </>
+   );

```

```

+ }
+
-
- const openActivity = useCallback((item: SidebarActivityQueryItem): void => {
-   void bus.call('sidebar->ext:openQueryActivity', {
-     locator: {
-       sessionId: item.sessionId,
-       storageScopeRelPath: item.storageScopeRelPath,
-     },
-   });
- }, []);
-
+
-
-   const running = item.status === 'running';
-   const dotStatus: VscStatusDotStatus = running
-     ? 'warning'
-     : 'success';
-   const timeText = running
-     ? `Elapsed: ${formatDurationHms(item.liveQueryElapsedMs)}`
-     : item.lastQueryElapsedMs === undefined
-     ? '-'
-     : `Last query: ${formatDurationHms(item.lastQueryElapsedMs)}`;
-   const displayTitle = formatUserPromptTitleForUi(item.title);
-
+   const running = item.status === 'running';
+
-
-   );
-   // Preserve the strict detached-source subtype through JSX so the Stop
subtree
-   // cannot accidentally accept an idle, observer-owned, or reopened
activity.
-   const detachedSource = isDetachedSourceItem(item)
-     ? item
-     : undefined;
-
+   );
+
-
-   return (
-     <Fragment
-
+   return (
+     <SidebarActivityQueryItemRow
+
-
-   }` }
-
-   >
-     <VscVertStackRow>
-       <VscHorGroupLayout allowShrink>
-         <VscStatusDot status={dotStatus} />
-         <VscLabel text={running ? 'Running' : 'Idle'} />
-         <VscLink
-           text={displayTitle}
-           target={() => openActivity(item)}
-           selected={selected}
-           singleLineWithEllipsis
-           tooltipText={displayTitle}
-           tooltipOnlyWhenEllipsed
-         />
-       </VscHorGroupLayout>
-     </VscVertStackRow>

```

```

-         <VscVertStackRow overrideSpacingTop="never">
-             <VscLabel text={timeText} />
-         </VscVertStackRow>
-         {detachedSource ? (
-             <DetachedSourceStop
-                 key={detachedSource.runRef.sourceId}
-                 item={detachedSource}
-             />
-         ) : null}
-     </Fragment>
-
+     }}
+     item={item}
+     selected={selected}
+
+ />
+

```

[AICode/src/frontend/src/app/sidebar/sidebar-activity/SidebarActivityQuery.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

- `baselineAndObservableChunk` additionally requires token, thinking token, or raw-live evidence.
- `baselineAndObservableChunk` additionally requires token, thinking token, or raw-live evidence.
- Every normal Prompt source uses `baselineOnly` for both replay modes so panel restoration can arm the composer before the first provider token. EasyAgents alone retains `baselineAndObservableChunk` as its historical product policy.

Abort is claimed atomically for either an active observer lease or an exact source-owned reservation/record. Claiming synchronously latches `abortRequested`; the returned callback must be invoked without an intervening `await`. A refused or failed frontend abort restores the exact still-active Stop capability and releases the local abort handshake so the operator may retry. The detached Activity Stop follows the same rule with local pending state instead of an irreversible click disable. When rendering already became idle but the transport capability remains stoppable, the handshake stays armed during the RPC and closes on its accepted response because no later `isStreaming` transition exists. A normal-Prompt source Stop latch is also released when replay converges to a stable failure, because that surface must keep Retry usable after a pre-baseline source retirement. Reservation snapshots expose their real source/observer ownership so panel coordination and Activity rehydration never reinterpret an EasyAgents reservation as a detached normal Prompt source. The registry emits deeply immutable advisory wake-ups only after complete observer-inactive and destruction-claim-release mutations, and isolates listener failures. When the last observer lease disappeared before EasyAgents completed its business handoff, the handoff emits the observer-inactive wake-up again so deferred panel cleanup is retried after the final authority transition. Exact sound state is retired for every source-handler finalization, including `stale`, because the full runRef prevents cleanup from touching a successor source.

Abort is claimed atomically for either an active observer lease or an exact source-owned reservation/record. Claiming synchronously latches `abortRequested`; the returned callback must be invoked without an intervening `await`. A refused or failed frontend abort restores the exact still-active Stop capability and releases the local abort handshake so the operator may retry. The detached Activity Stop follows the same rule with local pending state instead of an irreversible click disable. When rendering already became idle but the transport capability remains stoppable, the handshake stays armed

during the RPC and closes on its accepted response because no later `isStreaming` transition exists. A normal-Prompt source Stop latch is also released when replay converges to a stable failure, because that surface must keep Retry usable after a pre-baseline source retirement. Reservation snapshots expose their real source/observer ownership so panel coordination and Activity rehydration never reinterpret an EasyAgents reservation as a detached normal Prompt source. The registry emits deeply immutable advisory wake-ups only after complete observer-inactive and destruction-claim-release mutations, and isolates listener failures. When the last observer lease disappeared before EasyAgents completed its business handoff, the handoff emits the observer-inactive wake-up again so deferred panel cleanup is retried after the final authority transition. Exact sound state is retired for every source-handler finalization, including `stale`, because the full runRef prevents cleanup from touching a successor source.

An exact terminal Open retires normal-Prompt source Stop authority before full snapshot validation or hydration. Likewise, terminal data followed by `transportError` keeps the terminal unique but retires Stop because the exact source has already provided authoritative completion evidence. Viewer transport failures without terminal data preserve Stop because the extension source may still be active.

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

Copy

Review

Accept

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

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

Copy

Review

Accept

PatchFull file

- `replaceTurnAssistants` removes only assistants from the exact logical turn;
- `continueAssistant` overlays the immutable source-start message baseline.
- `replaceTurnAssistants` removes only assistants from the exact logical turn;
- `continueAssistant` overlays the immutable source-start message baseline.
- Both normal-Prompt replay modes become openable from their durable baseline alone; the observer may wait on an empty tail before the first provider token while the composer is already armed and Send remains locked.

- Terminal data followed by transport failure remains one visible terminal with `outcome='transportError'`.
- Terminal data followed by transport failure remains one visible terminal with `outcome='transportError'`.
- Exact terminal evidence retires the source Stop route before terminal snapshot hydration, and also when a later transport failure changes only the attempt outcome. Viewer failures without terminal evidence keep Stop available for the still-possible backend source.

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

Copy

Review

Accept

  reading project:  [NormalPromptLiveSessionSource.service.ts](#) [L1-

L75],  [NormalPromptLiveSessionSource.service.test.ts](#) [L55-L190],  [usePromptLiveSession.hook.ts](#) [L1-

L285],  [usePromptLiveSession.hook.ts](#) [L390-L475],  [usePromptLiveSession.normalPrompt.test.tsx](#) [L1-

L85],  [usePromptLiveSession.normalPrompt.test.tsx](#) [L330-L465],  [usePromptLiveSession.handoff.test.tsx](#) [L150-

L280],  [usePromptRunCommands.hook.ts](#) [L90-L155],  [usePromptRunCommands.hook.ts](#) [L475-

L590],  [usePromptRunController.hook.ts](#) [L465-L500],  [usePromptRunController.hook.ts](#) [L645-

L685],  [Prompt.tsx](#) [L250-L375],  [Prompt.tsx](#) [L1215-L1305],  [PromptInput.tsx](#) [L165-L205],  [PromptInput.tsx](#) [L960-L1060],  [SidebarActivityQuery.tsx](#), [liveSession.programming-doc-readme.md](#) [L25-L75], [isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md](#) [L15-L45], [usePromptIndexPending.hook.test.tsx](#)

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

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

Grouped

([what is it?](#))

  patching file:  [usePromptIndexPending.hook.tsx](#)

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

Error

([what is it?](#))

  patching file:  [usePromptIndexPending.hook.test.tsx](#)

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

Error

([what is it?](#))

  reading project:  [usePromptIndexPending.hook.tsx](#) [L35-

L175],  [usePromptIndexPending.hook.test.tsx](#) [L35-L285],  [usePromptLiveSession.hook.ts](#) [L1-L22]

  patching file:  [usePromptIndexPending.hook.tsx](#)

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

Copy

Review

Accept

PatchFull file

```
-
- /** Hook options to notify just before executing a buffered action. */
- interface UsePromptIndexPendingOptions {
-   /** Called immediately before a buffered action is executed (auto replay or
forceNow). */
-   onWillExecuteBufferedAction?: () => void;
- }
-
-
+
+
-
- /** Hook implementation. */
- export function usePromptIndexPending(
-   opts?: UsePromptIndexPendingOptions,
- ): UsePromptIndexPendingApi {
-
+
+ /** Hook implementation. */
+ export function usePromptIndexPending(): UsePromptIndexPendingApi {
+
-
-   /** Executes the buffered action, invoking the pre-execution hook and clearing
local state consistently. */
-
+
+   /** Execute the buffered action, then clear the complete local gate window
consistently. */
+
-
+

```

```

-         if (fn) {
-             // Signal the UI controller to lock just before the replayed action runs.
-             opts?.onWillExecuteBufferedAction?.();
-
-
+         if (fn) {
+
+
-             // Clear the entire gate window after the replay completed.
-
+             // Clear the entire gate window after the replay completed.
+
-         }, [opts]);
-
-         /** Clears the notice and drops any buffered action. */
-
+         }, []);
+
+         /** Clears the notice and drops any buffered action. */
+
-         if (fn) {
-             // Lock the UI just-in-time, same behavior as auto replay.
-             opts?.onWillExecuteBufferedAction?.();
-
-
+         if (fn) {
+
+
-             // Always clear the gate window after forcing.
-
+             // Always clear the gate window after forcing.
+
-         });
-         }, [opts]);
-
+         });
+         }, []);
+

```

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

Copy

Review

Accept

  patching file: [usePromptIndexPending.hook.test.tsx](#)

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

Error

([what is it?](#))

  reading project: [usePromptIndexPending.hook.test.tsx](#) [L45-

L95], [usePromptIndexPending.hook.test.tsx](#) [L185-L240], [usePromptIndexPending.hook.test.tsx](#) [L240-

L290], [usePromptIndexPending.hook.tsx](#) [L35-L155]

  patching file: [usePromptIndexPending.hook.test.tsx](#)

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

Error

([what is it?](#))

🔧  reading project:  [usePromptIndexPending.hook.test.tsx](#) [L55-L285]

🔧  patching file:  [usePromptIndexPending.hook.test.tsx](#)

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

Copy

Review

Accept

PatchFull file

```
-
- async function renderHarness(args: {
-   onExecute: () => void;
-   onWillExecuteBufferedAction?: () => void;
- }): Promise<void> {
-
+
+ async function renderHarness(args: { onExecute: () => void }): Promise<void> {
+


---


-   function Harness(): JSX.Element {
-     // Keep the harness aligned with the real hook contract so tests can
-     // assert both replay behavior and the pre-execution callback path.
-     const api = usePromptIndexPending({
-       onWillExecuteBufferedAction: args.onWillExecuteBufferedAction,
-     });
-
+   function Harness(): JSX.Element {
+     // Keep the harness aligned with the real hook contract and exercise replay
+     // directly.
+     const api = usePromptIndexPending();
+


---


-   test('forces the buffered action immediately and clears the notice', async () => {
-     // Start from a running snapshot so the first action goes through the
-     // exact same buffering path as a live Prompt blocked by indexing.
-     sidebarStoreSnapshot = {
-       sidebarContentState: {
-         activityState: {
-           indexState: {
-             isRunning: true,
-             cycleId: 'cycle-1',
-           },
-         },
-       },
-     };
-     const onExecute = jest.fn();
-     const onWillExecuteBufferedAction = jest.fn();
-
-     await renderHarness({
-       onExecute,
-       onWillExecuteBufferedAction,
-     });
-
-     await act(async () => {
-       // Buffer the action first so forceNow has one pending replay to run.
-       fireEvent.click(screen.getByTestId('run'));
-     });
-
-     expect(onExecute).not.toHaveBeenCalled();
-     expect(screen.getByTestId('pending')).toBeInTheDocument();
-
-   }
- }
```

```

-     await act(async () => {
-         // forceNow must replay immediately instead of waiting for the cycle stop.
-         fireEvent.click(screen.getByTestId('force'));
-     });
-
-     expect(onWillExecuteBufferedAction).toHaveBeenCalledTimes(1);
-     expect(onExecute).toHaveBeenCalledTimes(1);
-     expect(screen.queryByTestId('pending')).not.toBeInTheDocument();
-
-     await act(async () => {
-         // After forceNow cleared the gate window, a later stop must stay harmless.
-         latestBusHandler?.({
-             isRunning: false,
-             cycleId: 'cycle-1',
-         });
-     });
-
-     expect(onExecute).toHaveBeenCalledTimes(1);
- });
-
+ test('forces the buffered action immediately and clears the notice', async () => {
+     // Start from a running snapshot so the first action goes through the
+     // exact same buffering path as a live Prompt blocked by indexing.
+     sidebarStoreSnapshot = {
+         sidebarContentState: {
+             activityState: {
+                 indexState: {
+                     isRunning: true,
+                     cycleId: 'cycle-1',
+                 },
+             },
+         },
+     };
+     const onExecute = jest.fn();
+
+     await renderHarness({ onExecute });
+
+     await act(async () => {
+         // Buffer the action first so forceNow has one pending replay to run.
+         fireEvent.click(screen.getByTestId('run'));
+     });
+
+     expect(onExecute).not.toHaveBeenCalled();
+     expect(screen.getByTestId('pending')).toBeInTheDocument();
+
+     await act(async () => {
+         // forceNow must replay immediately instead of waiting for the cycle stop.
+         fireEvent.click(screen.getByTestId('force'));
+     });
+
+     expect(onExecute).toHaveBeenCalledTimes(1);
+     expect(screen.queryByTestId('pending')).not.toBeInTheDocument();
+
+     await act(async () => {
+         // After forceNow cleared the gate window, a later stop must stay harmless.
+         latestBusHandler?.({
+             isRunning: false,
+             cycleId: 'cycle-1',
+         });
+     });
+
+     expect(onExecute).toHaveBeenCalledTimes(1);

```

```

+   });
+
-
-   test('closes the notice without replaying the buffered action', async () => {
-     // Mirror one already-running indexing cycle so the hook buffers first.
-     sidebarStoreSnapshot = {
-       sidebarContentState: {
-         activityState: {
-           indexState: {
-             isRunning: true,
-             cycleId: 'cycle-1',
-           },
-         },
-       },
-     };
-     const onExecute = jest.fn();
-     const onWillExecuteBufferedAction = jest.fn();
-
-     await renderHarness({
-       onExecute,
-       onWillExecuteBufferedAction,
-     });
-
-     await act(async () => {
-       // Buffer one action so closeNotice can prove it really cancels pending
work.
-       fireEvent.click(screen.getByTestId('run'));
-     });
-
-     expect(screen.getByTestId('pending')).toBeInTheDocument();
-
-     await act(async () => {
-       // closeNotice must clear the notice and discard the replay callback.
-       fireEvent.click(screen.getByTestId('close'));
-     });
-
-     expect(onWillExecuteBufferedAction).not.toHaveBeenCalled();
-     expect(onExecute).not.toHaveBeenCalled();
-     expect(screen.queryByTestId('pending')).not.toBeInTheDocument();
-
-     await act(async () => {
-       // Once the buffered action was cancelled, the stop event must not
resurrect it.
-       latestBusHandler?.({
-         isRunning: false,
-         cycleId: 'cycle-1',
-       });
-     });
-
-     expect(onExecute).not.toHaveBeenCalled();
-   });
+
+   test('closes the notice without replaying the buffered action', async () => {
+     // Mirror one already-running indexing cycle so the hook buffers first.
+     sidebarStoreSnapshot = {
+       sidebarContentState: {
+         activityState: {
+           indexState: {
+             isRunning: true,
+             cycleId: 'cycle-1',
+           },
+         },
+       },
+     },
+

```

```

+         },
+     };
+     const onExecute = jest.fn();
+
+     await renderHarness({ onExecute });
+
+     await act(async () => {
+         // Buffer one action so closeNotice can prove it really cancels pending
work.
+         fireEvent.click(screen.getByTestId('run'));
+     });
+
+     expect(screen.getByTestId('pending')).toBeInTheDocument();
+
+     await act(async () => {
+         // closeNotice must clear the notice and discard the replay callback.
+         fireEvent.click(screen.getByTestId('close'));
+     });
+
+     expect(onExecute).not.toHaveBeenCalled();
+     expect(screen.queryByTestId('pending')).not.toBeInTheDocument();
+
+     await act(async () => {
+         // Once the buffered action was cancelled, the stop event must not
resurrect it.
+         latestBusHandler?.({
+             isRunning: false,
+             cycleId: 'cycle-1',
+         });
+     });
+
+     expect(onExecute).not.toHaveBeenCalled();
+ });
+

```

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

Copy

Review

Accept

  reading project:  e ["Seam soumission restant"](#),  e ["Callback gate restant"](#),  e ["Readiness Prompt ancienne"](#),  [usePromptIndexPending.hook.tsx](#) [L35-L155],  [usePromptIndexPending.hook.test.tsx](#) [L55-L285],  [usePromptLiveSession.hook.ts](#) [L1-L18],  [.prettierrc](#),  [iaStreamHandlers.queryDoneSound.test.ts](#) [L60-L95],  e ["Readiness handler test"](#),  e ["Réservation handler assertions"](#)

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

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

Grouped

([what is it?](#))

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

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

Grouped

([what is it?](#))

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

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

Copy

Review

Accept

```
-      replayMode: 'replaceTurnAssistants',
-      readiness: 'baselineAndObservableChunk',
-
+      replayMode: 'replaceTurnAssistants',
+      readiness: 'baselineOnly',
+
```

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

Accept

 reading project: [iaStreamHandlers.queryDoneSound.test.ts](#) [L120-L245]

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

(what is it?)

  *scanning recent files*

🔧 ☒ reading project: 📄 [Prompt.autoRunPrompt.test.tsx](#) [L210-L250], 🔍 e ["Open baseline sans chunk"](#), 🔍 e ["Tests"](#)

L35], [usePromptLiveSession.hook.ts](#) [L7-L18]

1. Replay Prompt disponible dès la baseline

- Le test du handler vérifie aussi explicitement que la réservation source reçoit :

- readiness: 'baselineOnly':

- `replayMode: 'replaceTurnAssistants'.`

2. Retrait exact de l'autorité Stop terminale

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

- l'enveloppe Open est d'abord validée contre le `sessionId`, `turnId` et `sourceId` attendus ;
- une branche Open `terminal` exacte retire l'autorité Stop **avant** la validation complète ou l'hydratation du snapshot ;
- si l'hydratation terminale échoue, l'écran peut afficher l'échec de replay, mais il ne présente plus un Stop obsolète ;
- un `transportError` après `terminalDataSeen=true` retire également l'autorité Stop ;
- un échec transport sans terminal conserve l'autorité, car la source extension peut encore être active ;
- le terminal visible reste unique : aucun second terminal ni second `meta.error` n'est créé.

Régressions ajoutées dans :

- [usePromptLiveSession.normalPrompt.test.tsx](#) : échec d'hydratation d'un snapshot terminal exact ;
- [usePromptLiveSession.handoff.test.tsx](#) : terminal consommé puis `transportError`.

3. Nettoyage du seam transactionnel de soumission

La chaîne de callback no-op `onBeforeSubmitUi` a été retirée des couches de production où elle suggérait incorrectement une mutation avant l'acceptation du stream :

- [usePromptRunCommands.hook.ts](#)
- [usePromptRunController.hook.ts](#)
- [Prompt.tsx](#)

L'invariant reste désormais sans ambiguïté :

1. le plan de streaming est construit sans mutation ;
2. `start()` décide synchroniquement de l'acceptation ;
3. les états `uiPending`, `scaffold` et barrières ne sont mutés qu'après acceptation.

Le callback de [PromptInput.tsx](#) reste optionnel pour les consommateurs autonomes du composant, mais sa documentation précise maintenant qu'il ne peut jamais supposer que le stream sera accepté ni armer un état transactionnel.

Le callback devenu inutilisé dans la gate d'indexation a également été supprimé de :

- [usePromptIndexPending.hook.tsx](#)
- Ses tests ont été alignés dans : [usePromptIndexPending.hook.test.tsx](#)

4. Callback stable pour les activités de session

La projection d'une ligne d'activité a été extraite dans un sous-composant dédié dans :

- [SidebarActivityQuery.tsx](#)

Le sous-composant :

- mémorise le locator exact avec `useMemo` ;
- expose un callback d'ouverture stable avec `useCallback` ;
- ne recrée plus une closure inline pour `VscLink` à chaque rendu de la liste ;
- conserve le sous-arbre Stop strictement limité à `Running/source/closed` ;
- conserve le remount par `sourceId` lors du remplacement d'une source.

5. Programming docs alignées

Les invariants corrigés ont été documentés dans :

- [liveSession.programming-doc-readme.md](#)
- [documentation d'isomorphisme streaming/hydratation](#)

Elles précisent maintenant que :

- tous les replay modes Prompt normaux deviennent ouvrables depuis la baseline seule ;
- EasyAgents conserve sa readiness historique ;
- un terminal exact retire Stop avant l'hydratation ;
- un terminal suivi d'un `transportError` retire également Stop ;
- une erreur viewer sans terminal conserve Stop.

Résumé des changements

Bloc	Résultat	Protection ajoutée	Complétude
Readiness Prompt normale	Toutes les sources utilisent <code>baselineOnly</code>	Réouverture avant le premier token provider	✓
EasyAgents	Politique historique	Aucun changement de comportement	✓

Bloc	Résultat	Protection ajoutée	Complétude
Open terminal	inchangée Retrait de Stop avant hydratation	produit Aucun Stop fantôme après snapshot terminal invalide	✓
Terminal puis transportError	Autorité Stop retirée	Terminal unique et bouton cohérent	✓
Erreur transport sans terminal	Autorité Stop conservée	Source backend toujours abortable	✓
Soumission transactionnelle	Seam no-op retiré de la chaîne de production	Zéro mutation avant acceptation	✓
Gate d'indexation	Callback pré-exécution mort supprimé	Responsabilités simplifiées	✓
Activité Sidebar	Ligne extraite avec callback stable	Moins de closures recrées	✓
Tests de source	Mapping et baseline pré-token couverts	Contrat <code>baselineOnly</code> verrouillé	✓
Tests frontend	Échec d'hydratation et terminal+transport couverts	Autorité Stop verrouillée	✓
Documentation technique	Invariants corrigés et explicités	Maintenance future sécurisée	✓
Fichiers générés	Aucun fichier généré modifié	Règle <code>*.generated.*</code> respectée	✓
Nom de commit proposé			

text

Copy

fix(live-session): restore prompt from baseline and retire stale stop authority

🎵 Le replay repart dès la baseline 🚫 Le Stop fantôme quitte enfin la vitrine 🎵

</ASSISTANT>

<USER>

</USER>

<ASSISTANT>

🔧 ✓ scanning problems

🔧 ✓ reading project: [NormalPromptLiveSessionSource.service.test.ts](#) [L90-L190], [iaStreamHandlers.queryDoneSound.test.ts](#) [L20-L95], [iaStreamHandlers.queryDoneSound.test.ts](#) [L190-L230], [usePromptRunController.hook.ts](#) [L60-L160], [usePromptRunController.hook.ts](#) [L635-L680], [Type résultat contrôleur"](#)

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

Copy

Review

Accept

PatchFull file

```
-         .mockReturnValueOnce([durableAssistant]);
-         const markBaselineReady = jest.fn(() => {
-             order.push('baseline');
-             return { becameReplayReady: true };
-         });
-
+         .mockReturnValueOnce([durableAssistant]);
+         const markBaselineReady = jest.fn<
+             (args: {
+                 runRef: LiveSessionRunRef;
```

```
+         baselineMessages: AssistantMessage[];
+     }) => { becameReplayReady: boolean }
+     >(() => {
+         order.push('baseline');
+         return { becameReplayReady: true };
+     });
+
```

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

Copy

Review

Accept

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

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

Error

([what is it?](#))

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

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

Copy

Review

Accept

PatchFull file

```
-     isRunningForInput: boolean;
-     onBeforeSubmitUi: () => void;
-
+     isRunningForInput: boolean;
+
```

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

Copy

Review

Accept

  reading project:  [iaStreamHandlers.queryDoneSound.test.ts](#) [L1-L42]

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

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

Copy

Review

Accept

PatchFull file

```
- import type { LiveSessionSourceReservationHandle } from
'../../../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
-
+ import type { LiveSessionSourceReservationHandle } from
'../../../../aiTools/orch/liveSession/registry/LiveSessionRecord.type.js';
+ import type { LiveSessionRegistryService } from
'../../../../aiTools/orch/liveSession/registry/LiveSessionRegistry.service.js';
+
- });
- const reserveSourceMock = jest.fn<
-     () => LiveSessionSourceReservationHandle | undefined
- >(() => reservation);
-
+ });
+ const reserveSourceMock = jest.fn<LiveSessionRegistryService['reserveSource']>()
+     () => reservation,
```

```
+ );
+
AICode/src/extension/src/messageBus/handlers/ia/ test /iaStreamHandlers.queryDoneSound.test.ts
Copy
Review
Accept
```

reading project: [NormalPromptLiveSessionSource.service.test.ts](#) [L100-L130], [iaStreamHandlers.queryDoneSound.test.ts](#) [L1-L38], [usePromptRunController.hook.ts](#) [L95-L112]

Les trois erreurs sont corrigées

Merci pour les diagnostics, Joel — trois erreurs nettes, trois corrections ciblées. Le compilateur avait simplement sorti son petit sifflet d'arbitre .

Diagnostic	Cause	Correction appliquée
markBaselineReady — argument refusé	Jest avait inféré un mock sans paramètres depuis <code>jest.fn(() => ...)</code>	Le mock utilise maintenant une fonctionnelle stricte décrivant <code>runRef</code> et <code>baseline</code>
reserveSourceMock — argument refusé	Le mock était explicitement typé <code>() => result</code>	Le mock reprend désormais du type réel <code>LiveSessionRegistry.eserveSource'</code>]
onBeforeSubmitUi manquant	La propriété avait été retirée du retour, mais restait obligatoire dans <code>UsePromptRunControllerResult</code>	La propriété obsolète a été retirée public

Fichiers corrigés

- [NormalPromptLiveSessionSource.service.test.ts](#)
- [iaStreamHandlers.queryDoneSound.test.ts](#)
- [usePromptRunController.hook.ts](#)

Les mocks restent **strictement typés**, sans `any`, sans cast et sans contournement ESLint.

Nom de commit proposé

```
text
Copy
fix(live-session): align regression mocks and controller contract
```

Les mocks reprennent le bon contrat Et TypeScript ne proteste plus tout bas

</ASSISTANT>