Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=fb9b8756c922424ca55a49b1190ff95f-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*
🔧 ✅ *reading project*: 🟨 g *"Docs live session"*, 🟨 g *"Docs EasyAgents"*, 🟨 g *"Docs Prompt streaming"*, 🟨 g *"Docs flux Orch"*, 🟨 g *"Docs bus frontend"*
🔧 ✅ *scanning git*: ♈ *"État Git actuel"*, ♈ *"Fichiers implémentés"*, ♈ *"Volume implémentation"*
🔧 ✅ *reading project*: 📄 *liveSession.programming-doc-readme.md*, 📄 *easyAgents.programming-doc-readme.md*, 📄 *isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md*, 📄 *thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md*, 📄 *prompt-stream-dispatch-lock.programming-doc-readme.md*, 📄 *fallback-persist-and-title.programming-doc-readme.md*
🔧 ✅ *scanning git*: ♈ *"Historique récent"*, ♈ *"Métadonnées du commit"*, ♈ *"Inventaire du commit"*, ♈ *"Statistiques du commit"*, ❌ ♈ *"Validité du diff"*
🔧 ✅ *reading project*: 🔎 h *"Identité live source"*, 🔎 h *"Registre réservations"*, 🔎 h *"Source Prompt normale"*, 🔎 h *"Origine webview exacte"*, 🔎 h *"Coordination panneau"*, 🔎 h *"Démarrage transactionnel"*, 🔎 h *"Replay Prompt normal"*, 🔎 h *"Activité détachée"*, 🔎 h *"Erreur Slimbk body"*, 🔎 h *"Queues identité tentative"*,
📄 *LiveSessionRun.type.ts*,
📄 *LiveSessionPanel.type.ts*,
📄 *LiveSessionOpenResult.type.ts*, 📄 *LiveSessionReplay.type.ts*, 📄 *SidebarStore.ts* [L100-L240], 📄 *SidebarUiMessages.ts* [L1-L180], 📄 *MessageRegistry.ts* [L1-L240], 📄 *parseSlimbkStreamBodyError.util.ts*, 📄 *aiStopSuffix.util.ts* [L1-L260], 📄 *LiveSessionRecord.type.ts*, 📄 *LiveSessionRegistry.service.ts* [L1-L460], 📄 *LiveSessionRegistry.service.ts* [L631-L910], 📄 *LiveSessionRegistry.service.ts* [L973-L1135],
📄 *LiveSessionSourceLifecycle.service.ts*,
📄 *NormalPromptLiveSessionSource.service.ts*,
📄 *LiveSessionService.ts*,
📄 *LiveSessionOpen.service.ts*, 📄 *extTypedMessageBus.util.ts*, 📄 *extMessageBus.util.ts* [L260-L330],
📄 *iaStreamHandlers.ts*,
📄 *liveSessionStreamHandlers.ts*,
📄 *stream.endpoint.ts*,
📄 *abortSource.endpoint.ts*, 📄 *abort.endpoint.ts*, 📄 *open.endpoint.ts*, 📄 *stream.endpoint.ts*, 🔎 e *"Nettoyage réservation"*, 🔎 h *"Échec avant source"*, 🔎 h *"Résultat reservationCancelled"*, 🔎 e *"Prise réservation"*, 📄 *iaStreamHandlers.queryDoneSound.test.ts* [L1-L230], 📄 *NormalPromptLiveSessionSource.service.test.ts* [L180-L230], 📄 *PromptLiveSessionPanelCoordinator.service.ts* [L100-L190],
📄 *PromptLiveSessionPanelQuery.util.ts*,
📄 *PromptLiveSessionPanelCoordinator.service.ts*,
📄 *PromptLiveSessionDetachedStop.service.ts*, 📄 *openPanel.util.ts* [L540-L690],
📄 *PromptLiveSessionBootstrap.adapter.ts*,
📄 *NormalPromptLiveSession.adapter.ts*, 📄 *PromptLiveSession.types.ts*, 📄 *Prompt.tsx* [L1-L220], 📄 *Prompt.tsx* [L940-L1260],
📄 *usePromptLiveSession.hook.ts*,
📄 *applyHydratedPromptSessionState.util.ts*, 📄 *AiStreamStartArgs.type.ts*, 📄 *useAiStream.hook.ts* [L1-L380], 📄 *useAiStream.hook.ts* [L540-

L1010], useAiStreamQueue.hook.ts, usePromptStreamDispatchGateQueue.hook.ts [L1-L430], h "Double Stop frontend", e "État Stop transport", usePromptCancelController.hook.ts, usePromptCancelController.stopCore.test.tsx [L1-L260], useAiStream.hook.promptSourceIdentity.test.ts [L1-L280], Prompt.tsx [L600-L760],
usePromptRunCommands.hook.ts,
preparePromptStreamingTurn.util.ts,
preparePromptContinuationStreaming.util.ts, usePromptRunCommands.startAcceptance.test.tsx [L1-L240], Prompt.fallbackPersistAck.clearAttachments.test.tsx [L1-L120], Prompt.fallbackPersistAck.clearInput.test.tsx [L1-L120]
scanning *git*: "Régression test pièces", "Régression test saisie"
reading *project*:
Prompt.liveSessionBootstrap.test.tsx,
useAiStream.hook.liveSessionObserver.test.ts,
applyHydratedPromptSessionState.util.test.ts,
usePromptRunController.liveSessionObserver.test.tsx,
SidebarQueryRunningBridge.ts,
SidebarStore.queryWindow.listeners.manager.ts,
SidebarStore.queryWindow.mutations.manager.ts,
SidebarStore.rehydrate.manager.ts,
SidebarActivityQuery.tsx, SidebarStore.handlers.manager.ts, delete.endpoint.ts, h "Ordre openedAt rehydrate", e "Réinitialisation activités", SidebarStore.rehydrate.manager.test.ts [L1-L260], PromptThreadCleanup.manager.ts, PromptQueryDoneSound.manager.ts, openPanel.util.ts [L1-L180], openPanel.util.ts [L180-L540], r "cleanupOnPromptDispose", r "openCentralPanel", r "registerSessionThread", e "Appels cleanup Prompt", e "Appels ouverture centrale", e "Enregistrements thread", LiveSessionRegistry.service.ts [L850-L1030], LiveSessionRegistry.service.ts [L400-L660], EasyAgentsBatchLiveSessionAdapter.service.ts, EasyAgentsBatchRunChildExecution.service.ts [L1-L300], EasyAgentsBatchRunRetry.service.ts [L1-L360], EasyAgentsBatchRunManager.service.ts [L1-L260], EasyAgentsBatchRunExecution.service.ts [L1-L220], EasyAgentsBatchRunRetry.service.ts [L300-L582], EasyAgentsBatchRunManager.service.ts [L240-L360], e "RunRef EasyAgents", h "Identité EasyAgents", usePromptSessionLocator.hook.ts [L1-L220], usePromptSessionLocator.hook.ts [L220-L303], e "Locator Prompt runtime", e "Assignation locator", Prompt.tsx [L215-L310], ChatSessionLocator.type.ts [L1-L140], PromptInput.tsx [L1-L200], e "Surface live PromptInput", PromptInput.tsx [L800-L1100], PromptLiveSessionBanner.tsx, PromptInput.module.css, PromptInput.tsx [L320-L460], e "Rendu bannière Prompt", e "Rendu composer Prompt", PromptInput.tsx [L1930-L2090], h "Attachments avant replay", PromptInput.batchLiveViewerMode.test.tsx, usePromptLiveSession.normalPrompt.test.tsx, e "Verrou historique live", e "Usage verrou historique", e "Rendu PromptHistory", Prompt.tsx [L1280-L1425], Prompt.tsx [L1425-L1456], e "Calcul interactions", Prompt.tsx [L430-L530], e "Usage cleanupOwned", e "Usage isViewerMode", e "Usage transport policy", e "Usage send policy", PromptInput.tsx [L1500-L1660],
closeSelf.util.ts,
useInitPage.hook.ts, frontMessageBus.util.ts, page.tsx, registerDeepLinkUriHandler.ts [L100-L175], e "Auto-run live query", Prompt.tsx [L740-L940], useQueryString.hook.ts, OrchMessagesMemoryAssistantPatch.ts [L1-L260], OrchMessagesMemoryAssistantPatch.errorSuffix.test.ts, aiStopSuffix.util.test.ts [L1-L260], parseSlimbkStreamBodyError.util.test.ts, e "Décodage V1 usages", h "Stack protocole V1"
scanning *git*: "Diff protocole V1"
reading *project*: e "Erreur client Slimbk", callLocalApi.generated.ts [L1-L240], h "Callback erreur stream", e "Restauration focus transaction", Prompt.tsx [L520-L600], usePromptThinkingLifecycle.hook.ts, usePromptToolUse.hook.ts [L1-

L260], usePromptRunController.finalization.hook.ts [L1-L260], usePromptRunController.hook.ts [L240-L540], usePromptRunController.hook.ts [L540-L649]

🔧 ✅ *scanning git*: "Diff Queue A", "Diff Queue B", "Diff tests Queue"

🔧 ✅ *reading project*: 🔎 e "Suffixe erreur mémoire", 🔎 h "Ajout suffixe erreur", OrchMessagesMemoryAssistantPatch.ts [L260-L390], OrchMessagesMemoryAssistantPatch.ts [L390-L430]

🔧 ✅ *scanning git*: ❌ "Interdits ajoutés", "Casts doubles ajoutés", "Types any ajoutés", ❌ "Espaces du commit", "Cast commandes Prompt", "Interdits simples", "Imports dynamiques"

🔧 ✅ *reading project*: usePromptLiveSession.normalPromptIsomorphism.test.tsx, usePromptLiveSession.handoff.test.tsx [L1-L260], usePromptLiveSession.hydration.test.tsx [L1-L180], PromptLiveSessionBootstrap.adapter.test.ts, NormalPromptLiveSession.adapter.test.ts, useAiStream.hook.ts [L360-L560], SidebarStore.core.manager.ts [L1-L180], 🔎 e "Transmission origine stream", extMessageBus.util.ts [L35-L90], useStreamListener.hook.ts

Audit de vérification

Joel, la spécification est **remarquablement rigoureuse** — avec 160 fichiers prévus, ce n'est plus une tranche, c'est un mille-feuille asynchrone avec verrous, époques et claims destructifs 🥐🔐.

J'ai vérifié le dépôt propre au commit `11f24c130c896401607bed487237f1efcd0e3afb`, dont le message correspond au nom demandé. J'ai également lu les documents d'architecture applicables, notamment :

- architecture générique liveSession
- architecture EasyAgents
- invariants d'isomorphisme streaming/hydratation
- invariants thinking/hourglass
- invariants Queue B
- invariants fallback persist

Le document personnel modifié dans le commit n'a pas été chargé.

---

1) La spécification entière est-elle implémentée ?

Non.

Estimation

Axe	Estimation	Verdict
Présence des 160 chemins explicitement demandés	**100 %**	✅
Implémentation fonctionnelle des invariants	**≈ 84 %**	⚠
Couverture comportementale conforme à la spécification	**≈ 55 %**	❌
Conformité globale de la tranche	**≈ 78 %**	❌

Les **160 fichiers listés dans la spécification** sont tous présents dans le commit comme créations ou modifications. Le commit modifie toutefois **174 fichiers**, soit **14 fichiers supplémentaires**.

Fichiers supplémentaires justifiés par les guards de tentative

Ces cinq changements ne figuraient pas dans la liste exacte, mais ils matérialisent bien l'invariant demandé pour `sourceId + streamInstanceId` sans changer les algorithmes fondamentaux des queues :

- `useAiStreamQueue.hook.ts`
- `usePromptStreamDispatchGateQueue.hook.ts`
- `AiStreamMergedQueue.test.ts`
- `PromptStreamDispatchGateQueue.test.ts`
- `useAiStream.hook.rawLiveSideband.test.ts`

Ce dépassement est **techniquement cohérent** avec la spécification, même si la liste exacte aurait dû les annoncer.

Test EasyAgents supplémentaire

- `EasyAgentsBatchRunRetry.live-restart.service.test.ts`

Ce fichier supplémentaire est pertinent.

Sorties générées supplémentaires

Les sept changements suivants sont cohérents avec une régénération Slimbk rendue nécessaire par les nouveaux contrats et endpoint :

- `allEndpoints.generated.ts`
- `callLocalApi.ai.orch.liveSession.generated.ts`

- callLocalApi.ai.orch.query.ask.generated.ts
- callLocalApi.generated.ts
- ProxyLocalApiMessages.ai.orch.liveSession.generated.ts
- ProxyLocalApiMessages.ai.orch.query.ask.generated.ts
- ProxyLocalApiMessages.generated.ts

Leur contenu est cohérent avec les sources. Rien dans l'audit ne permet de conclure à une modification manuelle.

Fichier personnel hors périmètre

Le commit contient également une modification de fichier personnel hors périmètre.

Cette modification viole l'invariant de périmètre de la spécification, même si elle ne change pas le comportement du produit.

Conclusion de la question 1 : la couverture structurelle est complète, mais plusieurs invariants fonctionnels et surtout la couverture de tests ne le sont pas ; la réponse est donc ❌ **non**.

---

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

J'ai regroupé les 160 changements en blocs fonctionnels cohérents.

Domaine technique demandé	État	Estimation	Résultat vérifié
Identité exacte `{ locator, turnId, sourceId }`	✅	**100 %**	Contrats stricts, run key structurelle et distinction logique/exécution correctement implémentés dans `LiveSessionRun.type.ts`.
Descripteurs live partagés et union adaptateur	✅	**100 %**	EasyAgents impose `sourceId===turnId`, sa ligne et `stopOnLastViewerClose`; normal Prompt impose `continueOnPanelClose` et interdit la ligne dans `LiveSessionPanel.type.ts`.
Résultats Open, enveloppe opaque et `replayMode`	✅	**98 %**	Branches strictes et propriétés présentes à `undefined` correctement contrôlées dans `LiveSessionOpenResult.type.ts`.
Réservations, promotion, époques, retraites, tombstones	✅	**96 %**	Le cœur du registre est bien structuré dans `LiveSessionRegistry.service.ts`. Le d vient de son intégration par le handler source, pas du lui-même.
Claims destructifs et génération destructive	✅	**96 %**	Acquisition, release, handle exact, éviction sous clai invalidation Open sont présents.
Wake-ups observer/claim immuables	✅	**95 %**	Émission après mutation complète et isolation des li présentes.
Lifecycle générique du `Readable`	✅	**95 %**	Priming avant callback synchrone, consommation, destruction, `return()` et agrégation déterministe dans `LiveSessionSourceLifecycle.service`
Source Prompt normale et persistance avant append	⚠	**82 %**	Le chemin nominal est correct dans `NormalPromptLiveSessionSource.serv` mais les échecs précédant la promotion peuvent aba une réservation active.
Suffixes SYSTEM_ERROR/INTERRUPTED persistés	✅	**95 %**	Ajout idempotent et persistance du snapshot implém dans `OrchMessagesMemoryAssistantPatch.t`
Protocole privé V1 du premier header	❌	**68 %**	L'encodeur est strict, mais le décodeur fabrique `status` et `errorCode` pour une chaîne ou une `Error`, ce qui relâche le contrat demandé.
Enveloppe Slimbk non-200 dans le body	✅	**100 %**	Détection préalable et séparation du protocole V1 co dans `parseSlimbkStreamBodyError.util.ts`
Origine extension-only avec webview exacte	✅	**100 %**	Le contexte est transmis hors payload par `extTypedMessageBus.util.ts` et vérifié av réservation.
FIFO extension, finalisation après	⚠	**88 %**	L'ordre nominal est correct dans `iaStreamHandle`

Domaine technique demandé	État	Estimation	Résultat vérifié
forwarding et erreurs auxiliaires			mais le nettoyage de réservation pré-promotion est i
Query extension, coordinateur et ouverture async	✅	**94 %**	Rejet avant low-level opener, injection normal Prom sérialisation par locator et dispose après suppression synchrone du panneau sont présents.
Frontière React des queries invalides	❌	**45 %**	La frontière existe, mais elle appelle `closeSelf()` toute initialisation du bus frontend.
Activité session-oriented et Stop détaché	✅	**92 %**	Union stricte, troisième ligne et Stop exact uniquem pour `Running/source/closed` sont bien rendus dans `SidebarActivityQuery.tsx`.
Propagation synchrone de `stopRequested`	✅	**95 %**	Le runtime et le store sont mutés avant l'abort ; la pr détachée récupère bien l'état.
Rehydrate Activity	⚠	**65 %**	Les projections principales sont reconstruites, mais l hiérarchie demandée `runtime → store → panne persistance → défaut` pour titre et `openedAt` implémentée.
Son exact par run et tentative	✅	**95 %**	Déduplication terminale, `soundAttemptId` et cons pendant la source/replay sont présentes dans `PromptQueryDoneSound.manager.ts`.
Delete durable sous claim	✅	**90 %**	Claim avant fermeture, éviction après suppression d réconciliation sur échec sont présents.
Retry EasyAgents sous claim	⚠	**78 %**	Claim, éviction et release en `finally` sont présents précheck production ne consulte pas l'autorité du registre `liveSession`.
Préservation EasyAgents historique	✅	**92 %**	`sourceId=turnId`, lifecycle partagé, ownership o handoff métier sont conservés.
`start()` synchrone et transactionnel	✅	**98 %**	Aucun ID ni état n'est créé avant l'acceptation, et ur démarrage dans la même pile est refusé dans `useAiStream.hook.ts`.
Guards `sourceId + streamInstanceId` dans Queue A/B	✅	**95 %**	Les algorithmes existants restent reconnaissables ; le sorties, coalescences et continuations sont correctem corrélées.
Commandes sans mutation lors d'un refus	✅	**95 %**	Send, SPECIF, Continue, Retry, Reply, Edit et Comp appellent `start()` avant leur scaffold dans `usePromptRunCommands.hook.ts`.
Replay Open, `replayOpenAttemptId`, retries annulables	✅	**93 %**	Une seule requête Open en vol, timer annulable, stal attempts et unmount correctement gardés dans `usePromptLiveSession.hook.ts`.
`preserveLocal` sans lecture du brouillon	✅	**98 %**	Les champs brouillon, attachments et `nextTurnId` non lus ; les getters piégés sont effectivement suppo dans `applyHydratedPromptSessionState.ut`
Handoff direct sans second Open	✅	**95 %**	Handoff uniquement après terminal, outcome `completed` et `finalizedStreamInsta` act ; aucun second Open.
Composer visible, Send verrouillé, Stop unique	⚠	**78 %**	Le textarea et le Stop unique sont corrects, mais les commandes d'attachments restent actives avant l'ins de la baseline.
Barrière `fallback_persisted`	⚠	**85 % comportement / 15 % preuve de test**	Le code de production corrèle bien `{turnId, streamInstanceId}` et démasque sans effacer lor finalisation sans ack, mais les tests réels ont été rem des assertions d'objets sans montage de Prompt.

Domaine technique demandé	État	Estimation	Résultat vérifié
Continue et thinking par `streamInstanceId`	⚠	**88 % comportement / 25 % isomorphisme testé**	Le plan de continuation et le thinking sont bien struc mais le test dit « isomorphisme » ne compare pas ré les chemins streaming/replay complets demandés.
Programming docs	✅	**100 %**	Les six documents génériques attendus ont été mis à
Tests exhaustifs des courses demandées	❌	**≈ 55 %**	Plusieurs nouveaux tests importants sont réels et util plusieurs anciens tests comportementaux ont été vid remplacés par des tests de forme.

Conclusion de la question 2 : le chemin nominal et l'architecture centrale sont largement implémentés, mais plusieurs chemins d'échec, le verrouillage pré-baseline des attachments, l'autorité Retry et une partie substantielle des preuves comportementales restent incomplets.

---

3) Y a-t-il des bugs ?
Oui : au moins cinq défauts fonctionnels vérifiés, deux défauts importants de validation, et un écart de périmètre.

🔥 **B1 — Réservation source abandonnée avant promotion**
Sévérité : critique.
Le handler réserve et stage la source avant d'appeler l'endpoint. Après le transport, il retire seulement le handle de la map de staging, puis finalise le registre :

AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts
Copy

```
    // The endpoint normally consumes this capability; remove it here only when transport
failed earlier.
    unstageNormalPromptLiveSessionReservation({ runRef, reservation });

    try {
        await fifo;
    } catch (error) {
        transportError = normalizeError(error);
        fifo = Promise.resolve();
    }
    // ...
    const finalization = liveSession.finalizeSourceAfterForwarding(runRef);
```

Aucun `cancelSourceReservation(reservation)` n'est exécuté sur ce chemin.
Cela se produit notamment si :

- le transport échoue avant d'atteindre l'endpoint ;
- la persistance de préférence SPECIF échoue ;
- le catalogue échoue avant `takeNormalPromptLiveSessionReservation(...)` ;
- la validation de baseline Continue échoue ;
- `askStream(...)` ou le priming échoue avant `registerRun(...)`.

Dans ces situations :

1. la réservation reste active ;
2. le pointeur du locator reste revendiqué ;
3. une nouvelle source est refusée ;
4. Open peut rester éternellement `not_ready` ;
5. le nettoyage du panneau reste bloqué ;
6. `finalizeSourceAfterForwarding(...)` retourne `stale` au lieu de `reservationCancelled`.

Le registre possède déjà le bon mécanisme d'annulation dans `LiveSessionRegistry.service.ts` ; le handler ne l'utilise pas après transport.

🔥 **B2 — Une query live frontend invalide ne peut pas fermer son panneau**
Sévérité : élevée.
Dans `Prompt.tsx`, la query invalide monte `InvalidPromptBootstrapBoundary` sans monter `PromptRuntime`.

Or `useInitPage('prompt')`, qui initialise le bus frontend, n'est appelé que dans `PromptRuntime`. La frontière invalide appelle ensuite `closeSelf()`, dont l'implémentation passe obligatoirement par le bus dans `closeSelf.util.ts`.

Conséquence vérifiée :

- le dialogue peut s'afficher ;
- `closeSelf()` rencontre un bus non initialisé ;
- le panneau ne se ferme pas comme demandé ;
- la promesse de l'effet n'est pas capturée.

Le test `Prompt.liveSessionBootstrap.test.tsx` ne monte plus Prompt et ne peut pas détecter ce défaut.

🔥 **B3 — Les attachments sont modifiables avant l'installation de la baseline**

Sévérité : élevée.

L'adaptateur normal Prompt indique correctement :

- `showComposer: true` pendant l'ouverture ;
- `composerInteractive: false` avant le replay ;
- `composerInteractive: true` après l'armement.

Mais dans `PromptInput.tsx` :

- le textarea respecte `composerInteractive` ;
- `AttachPopupMenu` reçoit toujours `enabled={true}` ;
- les chemins picker, paste et drag vérifient surtout `isViewerMode` ;
- normal Prompt utilise `isViewerMode: false`, même avant la baseline.

L'utilisateur peut donc ajouter ou modifier des attachments pendant `liveSessionOpening`. La première hydratation complète peut ensuite remplacer ces attachments locaux par ceux du snapshot, avec risque de :

- perte visuelle de l'attachment ajouté ;
- fichier stage-only devenu orphelin ;
- divergence entre état local et état disque.

Le test `PromptInput.batchLiveViewerMode.test.tsx` ne rend plus le composant ; il vérifie seulement des objets de view model.

⚠ **B4 — Le décodeur V1 relâche le contrat strict `status + errorCode`**

Sévérité : moyenne à élevée.

Le décodeur de `aiStopSuffix.util.ts` transforme une `Error` en simple message, puis fabrique un faux envelope pour toute chaîne non JSON :

AICode/src/sharedlib/src/types/aiTypes/aiStopSuffix.util.ts

Copy

```
function normalizeSourceErrorEnvelope(input: unknown): unknown {
    if (input instanceof Error) {
        return normalizeSourceErrorEnvelope(input.message);
    }
    if (typeof input !== 'string') {
        return input;
    }
    // ...
    return {
        status: '500_SERVER_ERROR',
        errorCode: LIVE_SESSION_SOURCE_ERROR_V1_CODE,
        errorMessage: input,
    };
}
```

Une `Error` ordinaire ou une chaîne commençant par le préfixe réservé peut donc être reconnue comme protocole V1 **sans posséder réellement** :

- `status='500_SERVER_ERROR'` ;
- une propriété propre `errorCode` ;
- le code V1 exact transporté par le client Slimbk.

Cela contredit explicitement la spécification. Le chemin `Error` ignore également sa stack au lieu d'en extraire la ligne utile demandée.
Le test `aiStopSuffix.util.test.ts` entérine actuellement ce comportement relâché en attendant qu'une simple `new Error(JSON.stringify(encoded))` soit acceptée.

🔥 **B5 — Retry EasyAgents ne consulte pas l'autorité du registre**
Sévérité : élevée dans une course destructive.
La production
configure `isManualQueryRunningForLocator` dans `EasyAgentsBatchRunManager.service.ts` uniquement à partir des items Activity.
L'adaptateur `EasyAgentsBatchLiveSessionAdapter.service.ts` n'expose pas l'autorité active source/réservation demandée par la spécification.
Si la projection Activity est absente ou stale alors qu'une source manuelle est réservée ou active :

1. le précheck Retry peut autoriser la suite ;
2. le claim destructif peut être acquis ;
3. l'éviction peut tuer cette source ;
4. le chat peut être supprimé sous une requête manuelle encore active.

Le claim protège l'atomicité de la destruction, mais il ne remplace pas le précheck métier qui devait **refuser** Retry face à une source manuelle active.

⚠ **B6 — `openedAt` et le titre ne suivent pas la hiérarchie demandée au rehydrate**
Sévérité : moyenne.
Dans `SidebarStore.rehydrate.manager.ts` :

- la liste Activity est d'abord vidée ;
- les panneaux ouverts reçoivent `openedAt: Date.now()` ;
- les sources reconstruites sans item antérieur reçoivent également un nouvel instant ;
- aucune récupération de l'ancien store ou de la persistance n'est appliquée pour `openedAt`.

La hiérarchie demandée `runtime/store/panneau/persistance/défaut` n'est donc pas présente. Cela peut modifier l'ordre d'affichage lors d'une réhydratation du Sidebar.

❌ **B7 — Plusieurs tests critiques ont été remplacés par des tests qui n'exercent plus le code**
Sévérité de qualité : élevée, bloquante pour considérer la tranche finalisée.
Exemples vérifiés :

Test	Avant → après	Problème
barrière attachments	**513 → 17 lignes**	Ne monte plus Prompt ; compare seulement deux objets.
barrière input/focus	**412 → 29 lignes**	Simule un objet local ; ne teste ni React, ni le masque, ni le focus.
bootstrap live Prompt	**test composant → 23 lignes**	Teste uniquement le dispatcher ; ne détecte pas le bus non initialisé de la frontière.
contrôleur observer	**test hook → objet typé**	Ne monte plus le contrôleur.
observer useAiStream	**test hook → objet typé**	Ne teste ni callbacks, ni outcome, ni Queue B, ni Stop.
surface PromptInput	**test rendu → objets de view model**	Ne teste plus l'existence unique du Stop ni le verrouillage des contrôles.
isomorphisme normal Prompt	**77 lignes**	Ne compare pas le streaming réel au replay ; vérifie uniquement un texte terminal hydraté.

Le test d'isomorphisme demandé devait couvrir notamment tous les types de query, thinking, tools, usage, edits, erreurs, EOF et transport après terminal. Ce n'est pas le cas.

⚠ **B8 — Modification d'un fichier personnel hors périmètre**
Sévérité produit : nulle ; sévérité de processus : réelle.
Le commit modifie fichier personnel interdit alors que la spécification et les règles projet excluent cet arbre.

🟡 **B9 — Plusieurs politiques du view model sont déclarées mais non consommées**
Sévérité : faible.

`historyInteractionsEnabled` et `cleanupOwned` sont définis dans `PromptLiveSession.types.ts` et renseignés par les adaptateurs, mais Prompt dérive directement ces décisions depuis `promptRuntimeMode`.
Le comportement actuel reste généralement équivalent, mais le contrat « politiques adaptateur explicites » n'est pas réellement respecté et ces champs deviennent trompeurs pour la maintenance.
Conclusion de la question 3 : oui, il existe des bugs réels, dont un blocage permanent potentiel du locator, une frontière invalide incapable de se fermer, une course de perte d'attachments et une course destructive EasyAgents.

---

4) Quel est mon avis sur le code source et la spécification ?
Avis sur la spécification
Excellente : 9,5/10. ✅
Ses points forts :

- identités exactes clairement séparées ;
- invariants de conservation explicites ;
- races et ordres de finalisation décrits avec précision ;
- distinction nette entre source, observer, réservation, record, retraite, tombstone et claim ;
- politique frontend/backend bien délimitée ;
- critères de tests exceptionnellement détaillés.

Sa seule faiblesse est son volume : certaines obligations ont été perdues pendant l'implémentation malgré la précision du document.
Avis sur le code source
Architecture : 8,5/10.
Finition fonctionnelle : 7/10.
Qualité des preuves de tests : 4,5/10.
Les meilleures parties sont :

- le registre `liveSession` ;
- l'identité `sourceId + streamInstanceId` ;
- le lifecycle partagé du `Readable` ;
- les guards Queue A/Queue B ;
- le contexte exact de webview ;
- `preserveLocal` sans lecture des champs brouillon ;
- le handoff direct ;
- la modularisation du coordinateur, de la query et du Stop détaché ;
- la documentation de maintenance.

Les parties les plus faibles sont :

- les jonctions entre réservation, endpoint et handler ;
- le bootstrap frontend invalide ;
- la frontière entre `composerInteractive` et les contrôles d'attachments ;
- le précheck destructif EasyAgents ;
- la réhydratation Activity ;
- la suppression de tests comportementaux au profit d'assertions statiques qui ne protègent plus le produit.

La base architecturale est bonne. Le principal problème n'est pas une mauvaise conception générale : ce sont des **coutures d'intégration oubliées**, exactement là où cette spécification demandait le plus de discipline.
Conclusion de la question 4 : la spécification est excellente et le cœur du code est solide, mais la tranche souffre de plusieurs oublis d'intégration et d'une dégradation trop importante de la qualité des tests.

---

5) Résumé du rapport dans un tableau

Question	Verdict	Estimation	Résumé
Toute la spécification est-elle implémentée ?	❌	**≈ 78 % global**	Tous les chemins sont présents, mais plusieurs invariants ne le sont pas.
Le chemin nominal fonctionne-t-il ?	✅ / ⚠️	**≈ 90 %**	Fermeture, replay, handoff, Stop, Activity et source identity sont largement en place.
Les chemins d'échec sont-ils sûrs ?	❌	**≈ 70 %**	Réservation pré-promotion, query invalide, Retry et attachments présentent des défauts.
Les contrats partagés sont-ils stricts ?	✅ / ⚠️	**≈ 93 %**	Très bons sauf le décodeur V1 trop permissif.
Le registre est-il solide ?	✅	**≈ 96 %**	Très bonne implémentation ; son caller omet une

Question	Verdict	Estimation	Résumé
			annulation indispensable.
Le frontend transactionnel est-il correct ?	✅ / ⚠️	**≈ 91 %**	Start/queues/handoff corrects ; contrôle attachments incomplet.
EasyAgents est-il préservé ?	✅ / ⚠️	**≈ 88 %**	UX et lifecycle préservés ; autorité registre absente du précheck Retry.
L'Activity est-elle complète ?	⚠	**≈ 84 %**	États et Stop corrects ; rehydrate et timestamps incomplets.
Les tests prouvent-ils la spécification ?	❌	**≈ 55 %**	Plusieurs tests critiques sont devenus non comportementaux.
Les programming docs sont-elles à jour ?	✅	**100 %**	Les documents génériques demandés ont été mis à jour.
La tranche est-elle finalisée ?	❌	**Non**	Les bugs P0/P1 et les tests doivent être corrigés.

TL-DR

Le commit met en place l'essentiel de l'architecture finale et contient de très bonnes fondations, mais il reste **cinq défauts fonctionnels importants**, une réhydratation Activity incomplète, un décodeur V1 non strict et surtout plusieurs tests critiques remplacés par des assertions qui ne testent plus le produit.

---

6) Conclusion en une phrase

La tranche est largement codée mais elle ne peut pas être considérée finalisée tant que la fuite de réservation, le bootstrap invalide, la course des attachments, l'autorité Retry et les tests comportementaux ne sont pas corrigés.

---

7) Quelles corrections sont nécessaires ?

🔥 **Lot A — Bugs runtime prioritaires**

- Annuler systématiquement la réservation exacte encore active après le transport et avant la finalisation.
- Ajouter les tests d'échec avant `take`, avant `registerRun`, pendant le catalogue, la baseline et le priming.
- Initialiser le bus minimal de la frontière live invalide et garantir la fermeture en `finally`.
- Bloquer picker, paste, drag, suppression et préparation d'attachments tant que `composerInteractive=false`.
- Ajouter l'autorité registre source/réservation au précheck Retry EasyAgents.

🧪 **Lot B — Restaurer les preuves comportementales**

- Restaurer de vrais tests React pour les deux barrières `fallback_persisted`.
- Monter la frontière Prompt invalide et vérifier dialogue + fermeture + absence de runtime.
- Monter PromptInput et vérifier exactement un Stop, Send verrouillé et attachments verrouillés avant baseline.
- Restaurer les tests réels de `useAiStream` et du contrôleur observer.
- Construire un vrai test d'isomorphisme streaming/replay couvrant query types, thinking, tools, usage, erreurs, EOF et terminal suivi d'une erreur transport.

🧭 **Lot C — Conformité et finition**

- Rendre le décodeur V1 strict sur les propriétés réelles `status` et `errorCode`, y compris pour `Error`.
- Extraire la ligne de stack utile demandée.
- Implémenter la hiérarchie `runtime → store → panneau → persistance → défaut` pour Activity.
- Préserver un `openedAt` stable et calculer l'elapsed final depuis `sourceStartedAtMs`.
- Consommer réellement les politiques `historyInteractionsEnabled` et `cleanupOwned`, ou supprimer ces champs trompeurs.
- Aligner la liste exacte des fichiers avec les cinq fichiers Queue A/B nécessaires.
- Le fichier personnel hors périmètre restera exclu de toute modification de ma part.

🛠️ **Lot D — Toutes les corrections**

Ce lot regroupe les bugs runtime, le protocole V1, Activity, les politiques frontend et la reconstruction complète des tests comportementaux. Le fichier personnel interdit ne sera ni chargé ni modifié.

🔥 Corriger les bugs runtime prioritaires

🧪 Restaurer les tests comportementaux

🧭 Corriger la conformité secondaire

🛠️ Corriger toutes les anomalies
</ASSISTANT>
<USER>
[Corriger toutes les anomalies]
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *running code interpreter*: *details*
🔧 ✅ *reading project*: *liveSession.programming-doc-readme.md*, *easyAgents.programming-doc-readme.md*, *isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md*, *thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md*, *prompt-stream-dispatch-lock.programming-doc-readme.md*, *fallback-persist-and-title.programming-doc-readme.md*
🔧 ✅ *scanning git*: *"État avant corrections"*
🔧 ✅ *reading project*: 🔎 h *"Réservation handler Prompt"*, 🔎 h *"Autorité Retry EasyAgents"*, 🔎 h *"Frontière bootstrap invalide"*, 🔎 h *"Verrou attachments composer"*, 🔎 h *"Réhydratation activités"*, 🔎 h *"Politiques view model"*, 🔎 h *"Décodeur V1 strict"*, 🔎 h *"Tests barrières Prompt"*,
iaStreamHandlers.ts,
iaStreamHandlers.queryDoneSound.test.ts,
iaStreamHandlers.runIdPropagation.test.ts, *NormalPromptLiveSessionSource.service.ts* [L1-L145], *EasyAgentsBatchLiveSessionAdapter.service.ts*, *EasyAgentsBatchRunManager.service.ts* [L220-L350],
EasyAgentsBatchRunRetry.service.ts,
EasyAgentsBatchRunRetry.testHarness.ts,
EasyAgentsBatchRunRetry.guards.service.test.ts, *EasyAgentsBatchRunRetry.live-restart.service.test.ts*, *aiStopSuffix.util.ts*, *aiStopSuffix.util.test.ts*, 🔎 e *"Callback erreurs générées"*, 🔎 e *"Header erreur généré"*, *callLocalApi.generated.ts* [L150-L220], 🔎 e *"Classe erreur Slimbk"*, *callLocalApi.generated.ts* [L35-L80], 🔎 e *"Ligne stack utile"*, *useInitBus.hook.ts*, 🔎 e *"Initialisation bus"*, *frontMessageBus.util.ts* [L360-L415], *useInitPage.hook.ts*, 🔎 e *"Initialisation page Prompt"*, *Prompt.tsx* [L1-L340], *Prompt.tsx* [L430-L600], *Prompt.tsx* [L720-L820], *Prompt.tsx* [L1000-L1160], *Prompt.tsx* [L1280-L1456], *PromptInput.tsx* [L100-L440], *PromptInput.tsx* [L740-L920], *PromptInput.tsx* [L1080-L1660], *PromptInput.tsx* [L1800-L2090],
AttachPopupMenu.tsx,
PromptLiveSession.types.ts,
NormalPromptLiveSession.adapter.ts, *EasyAgentsBatchLiveSession.adapter.ts* [L90-L170], *PromptInput.tsx* [L1600-L1810],
SidebarStore.rehydrate.manager.ts,
SidebarStore.queryWindow.listeners.manager.ts,
SidebarStore.queryWindow.mutations.manager.ts,
SidebarQueryRunningBridge.ts,
SidebarStore.rehydrate.manager.test.ts,
SidebarQueryRunningBridge.usage.test.ts, *SidebarQueryRunningBridge.race.test.ts*, *SidebarStore.ts* [L180-L330], *LiveSessionRecord.type.ts*, *OpenedPanelsRegistry.util.ts* [L1-L180], *buildSidebarChatListItem.util.ts* [L1-L240], *ChatPersistenceReader.service.ts* [L1-L240], *SidebarStore.ts* [L1-L180],
Prompt.fallbackPersistAck.clearAttachments.test.tsx,
Prompt.fallbackPersistAck.clearInput.test.tsx,
Prompt.liveSessionBootstrap.test.tsx,
PromptInput.batchLiveViewerMode.test.tsx,
useAiStream.hook.liveSessionObserver.test.ts,

usePromptRunController.liveSessionObserver.test.tsx,
usePromptLiveSession.normalPromptIsomorphism.test.tsx,
useAiStream.hook.promptSourceIdentity.test.ts, usePromptRunController.code_interpreter.status_and_deltas.test.ts,
PromptInput.attachments.docs.attachFile.test.tsx [L1-L220], Prompt.promptDraft.hydration.test.tsx [L1-L220], usePromptLiveSession.handoff.test.tsx [L1-L225]

scanning git: "Ancien test fallback PJ", "Ancien test fallback sais"

reading project: e "Demande édition message", Prompt.tsx [L930-L1005], Prompt.tsx [L1160-L1290], useAiStream.hook.ts [L900-L1008], useAiStream.hook.ts [L630-L780], e "Ref disponibilité Stop", e "Mutations disponibilité Stop", useAiStream.hook.ts [L160-L240], useAiStream.hook.ts [L570-L650]

scanning git: "Sites de mutation Stop"

reading project: usePromptRunController.hook.ts [L1-L180], usePromptRunController.hook.ts [L510-L620], h "Tests isomorphisme Prompt", preparePromptStreamingTurn.util.ts, EasyAgentsBatchLiveSessionAdapter.service.test.ts, e "Fabrique erreur système", PromptLiveSessionBanner.tsx [L1-L120], e "TestId textarea stop", e "VscTextarea testId", h "Test ids suppression pièces"

scanning git: "Exports locator Prompt"

reading project: usePromptSessionLocator.hook.ts [L1-L55], Prompt.tsx [L600-L740], useAiStream.hook.ts [L820-L950], e "Construction Retry", EasyAgentsBatchRunManager.service.ts [L130-L190], EasyAgentsBatchRunManager.service.ts [L60-L130]

patching file: iaStreamHandlers.ts
AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts
Copy
Review
Accept
PatchFull file

```
-
-     // The endpoint normally consumes this capability; remove it here only when
transport failed earlier.
-
+
+     // Remove any still-staged capability, then cancel the reservation if the endpoint
consumed it
+     // but failed before promotion. Promotion makes cancellation a strict no-op, so
this single
+     // cleanup path safely covers pre-endpoint, pre-register, and normal registered
executions.
+
```

```
-     // cleanup path safely covers pre-endpoint, pre-register, and normal registered
executions.
-     unstageNormalPromptLiveSessionReservation({ runRef, reservation });
-
+     // cleanup path safely covers pre-endpoint, pre-register, and normal registered
executions.
+     unstageNormalPromptLiveSessionReservation({ runRef, reservation });
+     liveSession.cancelSourceReservation(reservation);
+
```

AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts
Copy
Review
Accept

patching file: iaStreamHandlers.queryDoneSound.test.ts
AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts
Copy
Review

Accept
PatchFull file

```
- ) => Promise<void>;
-
+ ) => Promise<void>;
+
+ type StreamCallbacks = {
+     onChunk: (chunk: unknown) => void;
+     onError: (error: unknown) => void;
+ };
+
```

```
-     return false;
- });
-
+     return false;
+ });
+ const callLocalApiMock = jest.fn<
+     (
+         url: string,
+         args: unknown,
+         callbacks: StreamCallbacks,
+     ) => Promise<void>
+ >();
+
```

```
-     () => ({
-         callLocalApi: async (
-             _url: string,
-             _args: unknown,
-             callbacks: { onChunk: (chunk: unknown) => void },
-         ) => {
-             order.push('transport');
-             callbacks.onChunk({
-                 turnId: 'turn',
-                 type: 'meta',
-                 value: { metaType: 'fallback_persisted', value: true },
-             });
-             callbacks.onChunk({
-                 turnId: 'turn',
-                 type: 'meta',
-                 value: {
-                     metaType: 'error',
-                     value: {
-                         reason: 'system_error',
-                         message: 'system error',
-                     },
-                 },
-             });
-         },
-
+     () => ({
+         callLocalApi: callLocalApiMock,
+
```

```
-         });
-         cancelSourceReservationMock.mockReturnValue(true);
-
+         });
+         // The nominal endpoint promotes the reservation, so post-transport
cancellation is a no-op.
+         cancelSourceReservationMock.mockImplementation(() => {
+             order.push('cancel');
+             return false;
```

```
+         });
+
```

```
-         });
-     });
-
+         });
+         callLocalApiMock.mockImplementation(
+             async (_url, _args, callbacks): Promise<void> => {
+                 order.push('transport');
+                 callbacks.onChunk({
+                     turnId: 'turn',
+                     type: 'meta',
+                     value: { metaType: 'fallback_persisted', value: true },
+                 });
+                 callbacks.onChunk({
+                     turnId: 'turn',
+                     type: 'meta',
+                     value: {
+                         metaType: 'error',
+                         value: {
+                             reason: 'system_error',
+                             message: 'system error',
+                         },
+                     },
+                 });
+             },
+         );
+     });
+
```

```
-             order.lastIndexOf('emit'),
-         );
-
+             order.lastIndexOf('emit'),
+         );
+         expect(order.indexOf('cancel')).toBeGreaterThan(
+             order.indexOf('transport'),
+         );
+         expect(order.indexOf('cancel')).toBeLessThan(
+             order.indexOf('finalize'),
+         );
+
```

```
-         expect(order.at(-1)).toBe('coordinator');
-     });
-
+         expect(order.at(-1)).toBe('coordinator');
+     });
+
+     it('cancels a consumed reservation when transport fails before source promotion',
async () => {
+         registerIaStreamHandlers();
+         if (!handler) throw new Error('Expected registered stream handler.');
+
+         // Model the exact failure window where the local endpoint consumed the staged
handle
+         // but never promoted it to a record. The handler must retire that reservation
before
+         // post-forwarding finalization so the locator cannot remain permanently not-
ready.
+         unstageReservationMock.mockReturnValue(false);
+         cancelSourceReservationMock.mockImplementation(() => {
+             order.push('cancel');
+             return true;
```

```
+         });
+         finalizeSourceAfterForwardingMock.mockImplementation(() => {
+             order.push('finalize');
+             return 'reservationCancelled';
+         });
+         callLocalApiMock.mockImplementationOnce(
+             async (_url, _args, callbacks): Promise<void> => {
+                 order.push('transport');
+                 callbacks.onError(new Error('failed before promotion'));
+             },
+         );
+
+         await expect(
+             handler(
+                 {
+                     turnId: 'turn',
+                     sourceId: 'source',
+                     threadId: 'a'.repeat(32),
+                     storageScopeRelPath: '',
+                     aiQuery: { type: 'prompt', promptValue: 'hello' },
+                     aiQueryOptions: {},
+                 },
+                 () => {
+                     order.push('emit');
+                 },
+                 { htmlPage: 'prompt', webview },
+             ),
+         ).rejects.toThrow('failed before promotion');
+
+         expect(order).toEqual(
+             expect.arrayContaining([
+                 'transport',
+                 'unstage',
+                 'cancel',
+                 'finalize',
+                 'soundStop',
+                 'coordinator',
+             ]),
+         );
+         expect(order.indexOf('cancel')).toBeLessThan(
+             order.indexOf('finalize'),
+         );
+         expect(order).not.toContain('emit');
+     });
+
```

AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts
Copy
Review
Accept

🔧 ✅ *patching file*: *EasyAgentsBatchLiveSessionAdapter.service.ts*
AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/EasyAgentsBatchLiveSessionAdapter.service.ts
Copy
Review
Accept
PatchFull file

```
-         | 'hasViewerForGroup'
-         | 'claimViewerAbort'
-
+         | 'hasViewerForGroup'
+         | 'claimViewerAbort'
```

```
+         | 'listActiveSnapshots'
+
```

```
-             | 'claimViewerAbort'
-
+             | 'claimViewerAbort'
+             | 'listActiveSnapshots'
+
```

```
-         return this.liveSession.claimViewerAbort(args);
-     }
-
+         return this.liveSession.claimViewerAbort(args);
+     }
+
+     /** Return whether one exact child locator still owns an active normal-Prompt
source or reservation. */
+     public hasActiveSourceForLocator(locator: ChatSessionLocator): boolean {
+         // Sweep first so expired opening leases and terminal retention cannot make the
retry guard stale.
+         this.liveSession.sweep();
+
+         // Only source ownership blocks destructive Retry. EasyAgents observer records
are governed by
+         // the Batch runtime itself and must not be reinterpreted as manual Prompt
source authority.
+         return this.liveSession.listActiveSnapshots().some((snapshot) => {
+             if (
+                 snapshot.ownership !== 'source' ||
+                 snapshot.runRef.locator.sessionId !== locator.sessionId ||
+                 snapshot.runRef.locator.storageScopeRelPath !==
+                     locator.storageScopeRelPath
+             ) {
+                 return false;
+             }
+
+             return (
+                 snapshot.kind === 'reservation' ||
+                 (!snapshot.sourceSettled && !snapshot.handoffReady)
+             );
+         });
+     }
+
```

AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/EasyAgentsBatchLiveSessionAdapter.service.ts
Copy
Review
Accept

🔧 ✅ *patching file*: *EasyAgentsBatchLiveSessionAdapter.service.test.ts*
AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/__test__/EasyAgentsBatchLiveSessionAdapter.service.test.ts
Copy
Review
Accept
PatchFull file

```
-         claimViewerAbort: jest.fn<LiveSessionService['claimViewerAbort']>(),
-
+         claimViewerAbort: jest.fn<LiveSessionService['claimViewerAbort']>(),
+         listActiveSnapshots: jest.fn<
+             LiveSessionService['listActiveSnapshots']
+         >(() => []),
+
```

```
-         );
-     });
-
+         );
+     });
+
+     it('uses registry source authority even when Activity has no projection', () => {
+         const { adapter, liveSession } = createHarness();
+         liveSession.listActiveSnapshots.mockReturnValue([
+             {
+                 kind: 'reservation',
+                 runRef: {
+                     locator: RUN.locator,
+                     turnId: 'manual-turn',
+                     sourceId: 'manual-source',
+                 },
+                 viewerGroupId: 'normal-prompt',
+                 ownership: 'source',
+                 replayMode: 'replaceTurnAssistants',
+                 sourceStartedAtMs: 1,
+                 abortRequested: false,
+                 locatorEpoch: 1,
+             },
+         ]);
+
+         expect(adapter.hasActiveSourceForLocator(RUN.locator)).toBe(true);
+         expect(
+             adapter.hasActiveSourceForLocator({
+                 sessionId: 'd'.repeat(32),
+                 storageScopeRelPath: RUN.locator.storageScopeRelPath,
+             }),
+         ).toBe(false);
+         expect(liveSession.sweep).toHaveBeenCalledTimes(2);
+     });
+
```

AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/__test__/EasyAgentsBatchLiveSessionAdapter.service.test.ts

Copy
Review
Accept

patching file: *EasyAgentsBatchRunRetry.service.ts*

AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunRetry.service.ts

Copy
Review
Accept
PatchFull file

```
-     private readonly liveSessionAdapter: Pick<
-         EasyAgentsBatchLiveSessionAdapterService,
-
+     private readonly liveSessionAdapter: Pick<
+         EasyAgentsBatchLiveSessionAdapterService,
+         | 'hasActiveSourceForLocator'
+
```

```
-             EasyAgentsBatchLiveSessionAdapterService,
-
+             EasyAgentsBatchLiveSessionAdapterService,
+             | 'hasActiveSourceForLocator'
+
```

```
-
-             if (
-                 this.isManualQueryRunningForLocator(
-                     targetRow.childSessionLocator,
-                 )
-             ) {
-
+
+             if (
+                 this.liveSessionAdapter.hasActiveSourceForLocator(
+                     targetRow.childSessionLocator,
+                 ) ||
+                 this.isManualQueryRunningForLocator(
+                     targetRow.childSessionLocator,
+                 )
+             ) {
+
```

AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunRetry.service.ts
Copy
Review
Accept

patching file: *EasyAgentsBatchRunRetry.testHarness.ts*
AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.testHarness.ts
Copy
Review
Accept
PatchFull file

```
-     destructionClaimAvailable?: boolean;
-
+     destructionClaimAvailable?: boolean;
+     hasActiveLiveSessionSource?: boolean;
+
```

```
-         (claim: LiveSessionLocatorDestructionClaim) => boolean
-     >;
-
+         (claim: LiveSessionLocatorDestructionClaim) => boolean
+     >;
+     hasActiveSourceForLocatorMock: jest.MockedFunction<
+         (locator: ChatSessionLocator) => boolean
+     >;
+
```

```
-     >(() => {});
-
+     >(() => {});
+     const hasActiveSourceForLocatorMock = jest.fn<
+         (locator: ChatSessionLocator) => boolean
+     >(() => args?.hasActiveLiveSessionSource ?? false);
+
```

```
-         liveSessionAdapter: {
-
+         liveSessionAdapter: {
+             hasActiveSourceForLocator: hasActiveSourceForLocatorMock,
+
```

```
-         releaseLocatorDestructionClaimMock,
-
+         releaseLocatorDestructionClaimMock,
+         hasActiveSourceForLocatorMock,
+
```

AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.testHarness.ts
Copy
Review
Accept

patching file: *EasyAgentsBatchRunRetry.guards.service.test.ts*
AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.guards.service.test.ts
Copy
Review
Accept
PatchFull file

```
- import { EasyAgentsError } from '../../../shared/EasyAgentsError.js';
-
+ import { EasyAgentsError } from '../../../shared/EasyAgentsError.js';
+ import { EASY_AGENTS_BATCH_RETRY_CHILD_RUNNING_BLOCKED_REASON } from
'../EasyAgentsBatchRunRetryMessages.const.js';
+
```

```
-
-     it('allows the retry to restart live when a sibling pending line still exists in
the same parent batch', async () => {
-
+
+     it('refuses destructive retry from registry authority when the Activity projection
is missing', async () => {
+         const {
+             service,
+             hasActiveSourceForLocatorMock,
+             deleteChatByLocatorMock,
+             resolveAuthoritativeStateMock,
+         } = createRetryHarness({
+             hasActiveLiveSessionSource: true,
+             isManualQueryRunningForLocator: false,
+         });
+         resolveAuthoritativeStateMock.mockResolvedValue(
+             buildColdState({
+                 state: 'done',
+                 lineStatus: 'error',
+                 updatedAtMs: 10,
+                 includeChildLocator: true,
+             }),
+         );
+
+         // The registry is the durable runtime authority for active sources and
reservations.
+         // A missing/stale Sidebar projection must therefore never allow the old child
chat to be deleted.
+         await expect(
+             service.retryLine({
+                 batchParentLocator: PARENT_LOCATOR,
+                 lineNumber: 2,
+             }),
+         ).rejects.toThrow(
+             EASY_AGENTS_BATCH_RETRY_CHILD_RUNNING_BLOCKED_REASON,
+         );
+
+         expect(hasActiveSourceForLocatorMock).toHaveBeenCalledWith(
+             CHILD_LOCATOR,
+         );
+         expect(deleteChatByLocatorMock).not.toHaveBeenCalled();
+     });
```

```
+
+     it('allows the retry to restart live when a sibling pending line still exists in
the same parent batch', async () => {
+
```
AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.guards.service.test.ts
Copy
Review
Accept

patching file: *aiStopSuffix.util.ts*
AICode/src/sharedlib/src/types/aiTypes/aiStopSuffix.util.ts
Copy
Review
Accept
PatchFull file

```
-
- /** Normalize unknown diagnostic input into stable optional details. */
-
+
+ /** Extract one concise actionable stack frame without persisting an entire
environment-specific stack. */
+ function extractUsefulErrorStackLine(error: Error): string {
+     const stackLines = error.stack
+         ?.split(/\r?\n/u)
+         .map((line) => line.trim())
+         .filter((line) => line.length > 0);
+     const firstFrame = stackLines?.find((line) => line.startsWith('at '));
+     return firstFrame ?? error.message;
+ }
+
+ /** Normalize unknown diagnostic input into stable optional details. */
+
```

```
-     if (value instanceof Error) {
-         return value.stack ?? value.message;
-
+     if (value instanceof Error) {
+         return extractUsefulErrorStackLine(value);
+
```

```
-
- /** Return whether an object owns a property, including an explicitly undefined
property. */
- function hasOwn(value: object, key: PropertyKey): boolean {
-
+
+ /** Return whether an object owns a property while preserving the property's unknown
value type. */
+ function hasOwn<K extends PropertyKey>(
+     value: object,
+     key: K,
+ ): value is object & Record<K, unknown> {
+
```

```
-
- /** Normalize supported decoder inputs into an unknown structural envelope. */
- function normalizeSourceErrorEnvelope(input: unknown): unknown {
-     if (input instanceof Error) {
-         return normalizeSourceErrorEnvelope(input.message);
-     }
-     if (typeof input !== 'string') {
-         return input;
```

```
-     }
-
+
+ /** Normalize supported decoder inputs without inventing missing transport status or
error-code fields. */
+ function normalizeSourceErrorEnvelope(input: unknown): object | undefined {
+     if (typeof input === 'object' && input !== null) return input;
+     if (typeof input !== 'string') return undefined;
+
+
```

```
-     try {
-         return JSON.parse(input);
-
+     try {
+         const parsed: unknown = JSON.parse(input);
+         return typeof parsed === 'object' && parsed !== null
+             ? parsed
+             : undefined;
+
```

```
-     } catch {
-         return {
-             status: '500_SERVER_ERROR',
-             errorCode: LIVE_SESSION_SOURCE_ERROR_V1_CODE,
-             errorMessage: input,
-         };
-
+     } catch {
+         return undefined;
+
```

```
-     const normalized = normalizeSourceErrorEnvelope(input);
-     if (typeof normalized !== 'object' || normalized === null) {
-         return undefined;
-     }
-     const envelope = normalized as {
-         status?: unknown;
-         errorCode?: unknown;
-         errorMessage?: unknown;
-     };
-
+     const normalized = normalizeSourceErrorEnvelope(input);
+
```

```
-     if (
-         envelope.status !== '500_SERVER_ERROR' ||
-         !hasOwn(envelope, 'errorCode') ||
-         envelope.errorCode !== LIVE_SESSION_SOURCE_ERROR_V1_CODE ||
-         typeof envelope.errorMessage !== 'string'
-
+     if (
+         !normalized ||
+         !hasOwn(normalized, 'status') ||
+         normalized.status !== '500_SERVER_ERROR' ||
+         !hasOwn(normalized, 'errorCode') ||
+         normalized.errorCode !== LIVE_SESSION_SOURCE_ERROR_V1_CODE
+
```

```
-     }
-     const prefix = `${LIVE_SESSION_SOURCE_ERROR_V1_CODE}:`;
-
+     }
+
```

```
+     // Generated Slimbk errors expose the remote message through Error.message, while
serialized
+     // and structural envelopes expose errorMessage. Both still require real own
status/errorCode fields.
+     const errorMessage =
+         normalized instanceof Error
+             ? normalized.message
+             : hasOwn(normalized, 'errorMessage')
+               ? normalized.errorMessage
+               : undefined;
+     if (typeof errorMessage !== 'string') return undefined;
+
+     const prefix = `${LIVE_SESSION_SOURCE_ERROR_V1_CODE}:`;
+
```

```
-     const prefix = `${LIVE_SESSION_SOURCE_ERROR_V1_CODE}:`;
-     if (!envelope.errorMessage.startsWith(prefix)) {
-
+     const prefix = `${LIVE_SESSION_SOURCE_ERROR_V1_CODE}:`;
+     if (!errorMessage.startsWith(prefix)) {
+
```

```
-         const payload = ZLiveSessionSourceErrorV1Payload.parse(
-             JSON.parse(envelope.errorMessage.slice(prefix.length)),
-
+         const payload = ZLiveSessionSourceErrorV1Payload.parse(
+             JSON.parse(errorMessage.slice(prefix.length)),
+
```

AICode/src/sharedlib/src/types/aiTypes/aiStopSuffix.util.ts
Copy
Review
Accept

patching file: *aiStopSuffix.util.test.ts*
AICode/src/sharedlib/src/types/aiTypes/__test__/aiStopSuffix.util.test.ts
Copy
Review
Accept
PatchFull file

```
-         const error = new Error('boom');
-
+         const error = new Error('boom');
+         error.stack = [
+             'Error: boom',
+             '    at source.ts:10:2',
+             '    at caller.ts:20:4',
+         ].join('\n');
+
```

```
-             message: 'system error',
-             details: error.stack,
-
+             message: 'system error',
+             details: 'at source.ts:10:2',
+
```

```
-         );
-         expect(
-             decodeLiveSessionSourceErrorV1(new Error(JSON.stringify(encoded))),
-         ).toEqual(value);
-
+         );
+
```

```
+         // The generated client throws an Error subclass with own status/errorCode
fields and the
+         // exact remote errorMessage as Error.message. The decoder must accept that
real structure
+         // without treating an unrelated plain Error as the private protocol.
+         const structuredError = new Error(encoded.errorMessage);
+         Object.defineProperties(structuredError, {
+             status: {
+                 configurable: true,
+                 enumerable: true,
+                 value: encoded.status,
+             },
+             errorCode: {
+                 configurable: true,
+                 enumerable: true,
+                 value: encoded.errorCode,
+             },
+         });
+         expect(decodeLiveSessionSourceErrorV1(structuredError)).toEqual(value);
+         expect(
+             decodeLiveSessionSourceErrorV1(new Error(encoded.errorMessage)),
+         ).toBeUndefined();
+         expect(
+             decodeLiveSessionSourceErrorV1(encoded.errorMessage),
+         ).toBeUndefined();
+
```

AICode/src/sharedlib/src/types/aiTypes/__test__/aiStopSuffix.util.test.ts

Copy
Review
Accept

reading project: *EasyAgentsBatchRunRetry.service.ts* [L70-L180], *EasyAgentsBatchRunRetry.service.ts* [L245-L295], *aiStopSuffix.util.ts* [L80-L245], *iaStreamHandlers.queryDoneSound.test.ts* [L1-L260], *EasyAgentsBatchRunRetry.testHarness.ts* [L80-L210], *EasyAgentsBatchRunRetry.testHarness.ts* [L250-L420]

patching file: *Prompt.tsx*

AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx

Copy
Review
Accept
PatchFull file

```
-     PromptLiveSessionBootstrap,
-     PromptLiveSessionViewModel,
-
+     PromptLiveSessionBootstrap,
+
```

```
-
- /** Stable short-lived presentation used while an invalid live descriptor is explained
and closed. */
- const INVALID_LIVE_SESSION_VIEW_MODEL: PromptLiveSessionViewModel = {
-     isViewerMode: true,
-     showComposer: false,
-     composerInteractive: false,
-     sendEnabled: false,
-     historyInteractionsEnabled: false,
-     cleanupOwned: false,
-     transportStopAvailable: false,
-     banner: {
-         statusLabel: 'Opening live session...',
-         showStopAction: false,
```

```
-         stopEnabled: false,
-         showRetryAction: false,
-         retryEnabled: false,
-     },
- };
-
+
+ /** Bootstrap variants allowed to cross the external invalid-query boundary into Prompt
runtime. */
+ type PromptRuntimeDispatch = Exclude<
+     PromptBootstrapDispatch,
+     { kind: 'invalid' }
+ >;
+
```

```
- export function Prompt(): JSX.Element {
-
+ export function Prompt(): JSX.Element {
+     // Initialize the page before the bootstrap branch is selected. The invalid live-
session boundary
+     // still needs the typed bus so its explanatory dialog can always close the exact
webview panel.
+     useInitPage('prompt');
+
+
```

```
-         void (async () => {
-             await showVscAlertDialog({
-                 icon: 'error',
-                 title: 'Live session bootstrap failed',
-                 labelMessage: message,
-             });
-             await closeSelf();
-
+         void (async () => {
+             // Dialog rendering is best-effort. A popup failure must never strand an
invalid panel.
+             try {
+                 await showVscAlertDialog({
+                     icon: 'error',
+                     title: 'Live session bootstrap failed',
+                     labelMessage: message,
+                 });
+             } catch (error) {
+                 console.error(
+                     '[Prompt] Failed to show the invalid liveSession dialog:',
+                     error,
+                 );
+             }
+
+             // Closing is attempted independently because the descriptor is not allowed
to mount Prompt runtime.
+             try {
+                 await closeSelf();
+             } catch (error) {
+                 console.error(
+                     '[Prompt] Failed to close an invalid liveSession panel:',
+                     error,
+                 );
+             }
+
```

```
- }: {
-     dispatch: PromptBootstrapDispatch;
-
```

```
+ }: {
+     dispatch: PromptRuntimeDispatch;
+
```

```
- }): JSX.Element {
-     useInitPage('prompt');
-
-
+ }): JSX.Element {
+
```

```
-     const liveSessionBootstrap = useMemo<PromptLiveSessionBootstrap>(() => {
-         // Stabilize the derived descriptor so downstream effects cannot restart merely
because
-         // a defensive inactive/invalid object was reconstructed during an unrelated
render.
-
+     const liveSessionBootstrap = useMemo<PromptLiveSessionBootstrap>(() => {
+         // The external boundary already rejected invalid descriptors, so runtime owns
only classic or ready state.
+
```

```
-             ? dispatch.bootstrap
-             : dispatch.kind === 'invalid'
-               ? {
-                     kind: 'invalid',
-                     explicitLiveQuery: true,
-                     dialogTitle: 'Live session bootstrap failed',
-                     message: dispatch.message,
-                 }
-               : {
-                     kind: 'inactive',
-                     explicitLiveQuery: false,
-                 };
-
+             ? dispatch.bootstrap
+             : {
+                   kind: 'inactive',
+                   explicitLiveQuery: false,
+               };
+
```

```
-     }, [liveSessionBootstrap]);
-     const invalidLiveSessionDialogShownRef = useRef<boolean>(false);
-     useEffect(() => {
-         if (
-             liveSessionBootstrap.kind !== 'invalid' ||
-             invalidLiveSessionDialogShownRef.current
-         ) {
-             return;
-         }
-         invalidLiveSessionDialogShownRef.current = true;
-         void (async () => {
-             try {
-                 await showVscAlertDialog({
-                     icon: 'error',
-                     title: liveSessionBootstrap.dialogTitle,
-                     labelMessage: liveSessionBootstrap.message,
-                 });
-             } catch (error) {
-                 console.error(
-                     '[Prompt] Failed to show the invalid liveSession dialog:',
-                     error,
-                 );
-             }
```

```
-             try {
-                 await closeSelf();
-             } catch (error) {
-                 console.error(
-                     '[Prompt] Failed to close an invalid liveSession panel:',
-                     error,
-                 );
-             }
-         })();
-     }, [liveSessionBootstrap]);
-
-
+     }, [liveSessionBootstrap]);
+
--------------------------------------------------------------------------------
-
-     // Live replay may allow draft editing after baseline installation, but every Send
hotkey remains locked.
-     const runSubmissionLocked = finalIsLocked || promptRuntimeMode !== 'normal';
-
+
+     /**
+      * Resolve adapter-owned interaction policies before wiring hotkeys and cleanup
registration.
+      *
+      * The final banner model is built later with live Stop state, but these three
policies are intentionally
+      * independent from transport availability. Reading them here prevents Prompt from
silently bypassing the
+      * adapter contract by deriving behavior only from the coarse runtime-mode
discriminator.
+      */
+     const promptInteractionPolicy = useMemo(() => {
+         if (!liveSessionReadyBootstrap) {
+             const classicPromptEnabled = !isLiveSessionBootstrap;
+             return {
+                 sendEnabled: classicPromptEnabled,
+                 historyInteractionsEnabled: classicPromptEnabled,
+                 cleanupOwned: classicPromptEnabled,
+             };
+         }
+
+         const policyViewModel =
+             liveSessionReadyBootstrap.presentationAdapter.buildViewModel({
+                 mode: promptRuntimeMode,
+                 transportStopAvailable: false,
+                 stopEnabled: false,
+             });
+         return {
+             sendEnabled: policyViewModel.sendEnabled,
+             historyInteractionsEnabled:
+                 policyViewModel.historyInteractionsEnabled,
+             cleanupOwned: policyViewModel.cleanupOwned,
+         };
+     }, [
+         isLiveSessionBootstrap,
+         liveSessionReadyBootstrap,
+         promptRuntimeMode,
+     ]);
+
+     // Live replay may allow draft editing after baseline installation, but every Send
path remains policy-locked.
+     const runSubmissionLocked =
+         finalIsLocked || !promptInteractionPolicy.sendEnabled;
```

```
+
```

```
-
-     const interactionsEnabled = promptRuntimeMode === 'normal';
-
+
+     const interactionsEnabled =
+         promptInteractionPolicy.historyInteractionsEnabled;
+
```

```
-     const ownsThreadForCleanup =
-         interactionsEnabled && typeof openThreadId === 'string';
-
+     const ownsThreadForCleanup =
+         promptInteractionPolicy.cleanupOwned &&
+         typeof openThreadId === 'string';
+
```

```
-     const liveSessionViewModel = useMemo(() => {
-         if (!liveSessionReadyBootstrap) {
-             return isLiveSessionBootstrap
-                 ? INVALID_LIVE_SESSION_VIEW_MODEL
-                 : undefined;
-         }
-
+     const liveSessionViewModel = useMemo(() => {
+         if (!liveSessionReadyBootstrap) return undefined;
+
```

```
-         liveSessionHandoffFailureMessage,
-         isLiveSessionBootstrap,
-
+         liveSessionHandoffFailureMessage,
+
```

AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx
Copy
Review
Accept

🔧 ❌ *patching file*: *useAiStream.hook.ts*
AICode/src/frontend/src/messageBus/useAiStream.hook.ts
Error
(what is it?)

🔧 ❌ *patching file*: *SidebarStore.queryWindow.listeners.manager.ts*
AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts
Error
(what is it?)

🔧 ✅ *patching file*: *SidebarQueryRunningBridge.ts*
AICode/src/extension/src/messageBus/handlers/ia/SidebarQueryRunningBridge.ts
Copy
Review
Accept
PatchFull file

```
- import type { AiStreamedChunk } from
'../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
-
```

```
+ import type { AiStreamedChunk } from
'../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import type { SidebarActivityQueryItem } from
'../../../sharedlib/types/busTypes/ui/SidebarStore.js';
+
```

```
-         if (!runtime || runtime.ownership !== 'source') return;
-
+         if (!runtime || runtime.ownership !== 'source') return;
+
+         // Freeze the exact wall-clock duration before removing the runtime so a sub-
second final
+         // interval does not leave Activity with the previous timer tick.
+         const finalElapsedMs = Math.max(
+             0,
+             Date.now() - runtime.sourceStartedAtMs,
+         );
+
```

```
-         this.runtimeByLocator.delete(this.locatorKey(runRef.locator));
-         await sourceSettled(runRef);
-
+         this.runtimeByLocator.delete(this.locatorKey(runRef.locator));
+         await sourceSettled(runRef, finalElapsedMs);
+
```

```
-         if (!runtime || runtime.ownership !== 'observer') return;
-
+         if (!runtime || runtime.ownership !== 'observer') return;
+         const finalElapsedMs = Math.max(
+             0,
+             Date.now() - runtime.sourceStartedAtMs,
+         );
+
```

```
-         this.runtimeByLocator.delete(this.locatorKey(runRef.locator));
-         await observerSettled(runRef);
-
+         this.runtimeByLocator.delete(this.locatorKey(runRef.locator));
+         await observerSettled(runRef, finalElapsedMs);
+
```

```
-                 panelState: 'open' | 'closed';
-
+                 panelState: 'open' | 'closed';
+                 title?: string;
+                 openedAt?: number;
+                 liveQueryElapsedMs: number;
+                 lastQueryElapsedMs?: number;
+                 usage?: SidebarActivityQueryItem['usage'];
+
```

```
-             panelState: item?.status === 'running' ? item.panelState : 'closed',
-
+             panelState: item?.status === 'running' ? item.panelState : 'closed',
+             title: item?.title,
+             openedAt: item?.openedAt,
+             liveQueryElapsedMs:
+                 item?.status === 'running'
+                     ? item.liveQueryElapsedMs
+                     : Math.max(0, Date.now() - runtime.sourceStartedAtMs),
+             lastQueryElapsedMs: item?.lastQueryElapsedMs,
+             usage: item?.usage,
+
```

AICode/src/extension/src/messageBus/handlers/ia/SidebarQueryRunningBridge.ts

Copy

Review
Accept

reading project: *useAiStream.hook.ts* [L190-L225], *useAiStream.hook.ts* [L585-L730], *useAiStream.hook.ts* [L840-L945], *SidebarStore.queryWindow.listeners.manager.ts*, *Prompt.tsx* [L80-L170], *Prompt.tsx* [L480-L570], *SidebarQueryRunningBridge.ts* [L130-L245]

patching file: *useAiStream.hook.ts*

AICode/src/frontend/src/messageBus/useAiStream.hook.ts

Copy
Review
Accept
PatchFull file

```
-     const [transportStopAvailable, setTransportStopAvailable] = useState(false);
-
+     const [transportStopAvailable, setTransportStopAvailable] = useState(false);
+     const transportStopAvailableRef = useRef(false);
+
```

```
-         undefined,
-     );
-
+         undefined,
+     );
+
+     /** Update Stop availability in React state and in the synchronous same-stack authority. */
+     const updateTransportStopAvailable = useCallback(
+         (available: boolean): void => {
+             // The ref closes the double-click window before React commits the matching visual state.
+             transportStopAvailableRef.current = available;
+             setTransportStopAvailable(available);
+         },
+         [],
+     );
+
```

```
-         terminalDataSeenRef.current = false;
-         setTransportStopAvailable(false);
-
+         terminalDataSeenRef.current = false;
+         updateTransportStopAvailable(false);
+
```

```
-         syntaxErrorTagNoAttributesSanitizer,
-         syntaxErrorTagOpenerSanitizer,
-
+         syntaxErrorTagNoAttributesSanitizer,
+         syntaxErrorTagOpenerSanitizer,
+         updateTransportStopAvailable,
+
```

```
-             terminalDataSeenRef.current = false;
-             setTransportStopAvailable(true);
-
+             terminalDataSeenRef.current = false;
+             updateTransportStopAvailable(true);
+
```

```
-             threadId,
-             dispatchGateQueue,
-
+             threadId,
```

```
+             dispatchGateQueue,
+             updateTransportStopAvailable,
+
```

```
-         try {
-             if (!transportStopAvailable) return;
-             setTransportStopAvailable(false);
-
+         try {
+             if (!transportStopAvailableRef.current) return;
+
+             // Consume the exact transport capability synchronously before the first
await. A same-stack
+             // double click therefore cannot send a second source/viewer abort request.
+             updateTransportStopAvailable(false);
+
```

```
-         // The stream listener will set isStreaming=false through meta/complete.
-     }, [activeStream, threadId, transportStopAvailable]);
-
+         // The stream listener will set isStreaming=false through meta/complete.
+     }, [activeStream, threadId, updateTransportStopAvailable]);
+
```

```
-                             setOutcome('completed');
-                             setTransportStopAvailable(false);
-
+                             setOutcome('completed');
+                             updateTransportStopAvailable(false);
+
```

```
-                             setOutcome('transportError');
-                             setTransportStopAvailable(false);
-
+                             setOutcome('transportError');
+                             updateTransportStopAvailable(false);
+
```

```
-             streamQueue,
-
+             streamQueue,
+             updateTransportStopAvailable,
+
```

AICode/src/frontend/src/messageBus/useAiStream.hook.ts
Copy
Review
Accept

🔧 ❌ *patching file*: *SidebarStore.queryWindow.listeners.manager.ts*
AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts
Error
(what is it?)

🔧 ✅ *reading project*: *SidebarStore.queryWindow.listeners.manager.ts* [L35-L290]
🔧 ❌ *patching file*: *SidebarStore.queryWindow.listeners.manager.ts*
AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts
Error
(what is it?)

🔧 ✅ *patching file*: *SidebarStore.queryWindow.listeners.manager.ts*

AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts
Copy
Review
Accept
PatchFull file

```
-     }
- }
-
+     }
+ }
+
+ /** Normalize one optional elapsed timestamp while rejecting NaN, infinity, and
negative values. */
+ function normalizeElapsedMs(value: number | undefined): number | undefined {
+     return typeof value === 'number' && Number.isFinite(value) && value >= 0
+         ? Math.floor(value)
+         : undefined;
+ }
+
```

```
-     openedAt: number;
-
+     openedAt: number;
+     lastQueryElapsedMs?: number;
+     usage?: SidebarActivityQueryItem['usage'];
+
```

```
-         args.title?.trim() || getVisibleChatTitleOrUntitled(args.locator);
-
+         args.title?.trim() || getVisibleChatTitleOrUntitled(args.locator);
+     const lastQueryElapsedMs = normalizeElapsedMs(args.lastQueryElapsedMs);
+
```

```
-             storageScopeRelPath: args.locator.storageScopeRelPath,
-             title,
-
+             storageScopeRelPath: args.locator.storageScopeRelPath,
+             title,
+             ...(lastQueryElapsedMs === undefined
+                 ? {}
+                 : { lastQueryElapsedMs }),
+             ...(args.usage === undefined ? {} : { usage: args.usage }),
+
```

```
-     stopRequested: boolean;
-
+     stopRequested: boolean;
+     openedAt?: number;
+     liveQueryElapsedMs?: number;
+     lastQueryElapsedMs?: number;
+     usage?: SidebarActivityQueryItem['usage'];
+
```

```
-     const now = Date.now();
-
+     const now = Date.now();
+     const restoredLiveQueryElapsedMs = normalizeElapsedMs(
+         args.liveQueryElapsedMs,
+     );
+     const restoredLastQueryElapsedMs = normalizeElapsedMs(
+         args.lastQueryElapsedMs,
+     );
+
```

```
-         );
```

```
-         const openedAt = index >= 0 ? items[index].openedAt : now;
-
+         );
+         const openedAt =
+             index >= 0
+                 ? items[index].openedAt
+                 : normalizeElapsedMs(args.openedAt) ?? now;
+
```

```
-             runRef: args.runRef,
-             liveQueryElapsedMs: Math.max(0, now - args.sourceStartedAtMs),
-
+             runRef: args.runRef,
+             liveQueryElapsedMs:
+                 restoredLiveQueryElapsedMs ??
+                 Math.max(0, now - args.sourceStartedAtMs),
+             ...(restoredLastQueryElapsedMs === undefined
+                 ? {}
+                 : { lastQueryElapsedMs: restoredLastQueryElapsedMs }),
+             ...(args.usage === undefined ? {} : { usage: args.usage }),
+
```

```
-
- /** Final non-stale source transition: open panels become idle, closed activities
disappear. */
- export async function sourceSettled(runRef: LiveSessionRunRef): Promise<void> {
-     await updateSidebarStore((store) => {
-         const items = store.sidebarContentState.activityState.queryState.items;
-         const index = items.findIndex(
-             (candidate) =>
-                 candidate.status === 'running' &&
-                 candidate.ownership === 'source' &&
-                 candidate.runRef.turnId === runRef.turnId &&
-                 candidate.runRef.sourceId === runRef.sourceId &&
-                 isSameQueryActivityLocator(candidate, runRef.locator),
-         );
-         if (index < 0) return;
-         const current = items[index];
-         if (current.status !== 'running' || current.ownership !== 'source')
-             return;
-         if (current.panelState === 'closed') {
-             items.splice(index, 1);
-             return;
-         }
-         items[index] = {
-             status: 'idle',
-             panelState: 'open',
-             sessionId: current.sessionId,
-             storageScopeRelPath: current.storageScopeRelPath,
-             title: current.title,
-             lastQueryElapsedMs: current.liveQueryElapsedMs,
-             usage: current.usage,
-             updatedAt: Date.now(),
-             openedAt: current.openedAt,
-         };
-     });
-     emitSafely(runningListeners, {
-         locator: runRef.locator,
-         running: false,
-     });
- }
-
+
```

```
+ /** Final non-stale source transition: open panels become idle, closed activities
disappear. */
+ export async function sourceSettled(
+     runRef: LiveSessionRunRef,
+     finalElapsedMs?: number,
+ ): Promise<void> {
+     const normalizedFinalElapsed = normalizeElapsedMs(finalElapsedMs);
+     await updateSidebarStore((store) => {
+         const items = store.sidebarContentState.activityState.queryState.items;
+         const index = items.findIndex(
+             (candidate) =>
+                 candidate.status === 'running' &&
+                 candidate.ownership === 'source' &&
+                 candidate.runRef.turnId === runRef.turnId &&
+                 candidate.runRef.sourceId === runRef.sourceId &&
+                 isSameQueryActivityLocator(candidate, runRef.locator),
+         );
+         if (index < 0) return;
+         const current = items[index];
+         if (current.status !== 'running' || current.ownership !== 'source')
+             return;
+         if (current.panelState === 'closed') {
+             items.splice(index, 1);
+             return;
+         }
+         items[index] = {
+             status: 'idle',
+             panelState: 'open',
+             sessionId: current.sessionId,
+             storageScopeRelPath: current.storageScopeRelPath,
+             title: current.title,
+             lastQueryElapsedMs:
+                 normalizedFinalElapsed ?? current.liveQueryElapsedMs,
+             usage: current.usage,
+             updatedAt: Date.now(),
+             openedAt: current.openedAt,
+         };
+     });
+     emitSafely(runningListeners, {
+         locator: runRef.locator,
+         running: false,
+     });
+ }
+
```

```
-
- /** Final non-stale observer transition: its still-open Prompt panel becomes Idle. */
- export async function observerSettled(
-     runRef: LiveSessionRunRef,
- ): Promise<void> {
-     await updateSidebarStore((store) => {
-         const items = store.sidebarContentState.activityState.queryState.items;
-         const index = items.findIndex(
-             (candidate) =>
-                 candidate.status === 'running' &&
-                 candidate.ownership === 'observer' &&
-                 candidate.runRef.turnId === runRef.turnId &&
-                 candidate.runRef.sourceId === runRef.sourceId &&
-                 isSameQueryActivityLocator(candidate, runRef.locator),
-         );
-         if (index < 0) return;
-         const current = items[index];
-         if (current.status !== 'running' || current.ownership !== 'observer')
-             return;
```

```
-
-         // Observer activities can exist only while their panel is open, so settlement
always projects Idle/open.
-         items[index] = {
-             status: 'idle',
-             panelState: 'open',
-             sessionId: current.sessionId,
-             storageScopeRelPath: current.storageScopeRelPath,
-             title: current.title,
-             lastQueryElapsedMs: current.liveQueryElapsedMs,
-             usage: current.usage,
-             updatedAt: Date.now(),
-             openedAt: current.openedAt,
-         };
-     });
-     emitSafely(runningListeners, {
-         locator: runRef.locator,
-         running: false,
-     });
- }
-
+
+ /** Final non-stale observer transition: its still-open Prompt panel becomes Idle. */
+ export async function observerSettled(
+     runRef: LiveSessionRunRef,
+     finalElapsedMs?: number,
+ ): Promise<void> {
+     const normalizedFinalElapsed = normalizeElapsedMs(finalElapsedMs);
+     await updateSidebarStore((store) => {
+         const items = store.sidebarContentState.activityState.queryState.items;
+         const index = items.findIndex(
+             (candidate) =>
+                 candidate.status === 'running' &&
+                 candidate.ownership === 'observer' &&
+                 candidate.runRef.turnId === runRef.turnId &&
+                 candidate.runRef.sourceId === runRef.sourceId &&
+                 isSameQueryActivityLocator(candidate, runRef.locator),
+         );
+         if (index < 0) return;
+         const current = items[index];
+         if (current.status !== 'running' || current.ownership !== 'observer')
+             return;
+
+         // Observer activities can exist only while their panel is open, so settlement
always projects Idle/open.
+         items[index] = {
+             status: 'idle',
+             panelState: 'open',
+             sessionId: current.sessionId,
+             storageScopeRelPath: current.storageScopeRelPath,
+             title: current.title,
+             lastQueryElapsedMs:
+                 normalizedFinalElapsed ?? current.liveQueryElapsedMs,
+             usage: current.usage,
+             updatedAt: Date.now(),
+             openedAt: current.openedAt,
+         };
+     });
+     emitSafely(runningListeners, {
+         locator: runRef.locator,
+         running: false,
+     });
+ }
```

+
AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts
Copy
Review
Accept

🔧 ❌ *patching file*: *PromptInput.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Error
(what is it?)

🔧 ✅ *reading project*: *PromptInput.tsx* [L1-L460], *PromptInput.tsx* [L1080-L1760], *PromptInput.tsx* [L1990-L2085]
🔧 ✅ *patching file*: *PromptInput.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Copy
Review
Accept
PatchFull file

```
-     useEffect,
-
+     useEffect,
+     useLayoutEffect,
+
```

```
-             liveSessionViewModel ?? PROMPT_INPUT_NORMAL_LIVE_SESSION_VIEW_MODEL;
-
+             liveSessionViewModel ?? PROMPT_INPUT_NORMAL_LIVE_SESSION_VIEW_MODEL;
+         const attachmentsInteractive =
+             viewerMode.showComposer && viewerMode.composerInteractive;
+         const attachmentsInteractiveRef = useRef(attachmentsInteractive);
+
+         /** Keep asynchronous attachment callbacks guarded by the latest live-session
composer policy. */
+         useLayoutEffect(() => {
+             // Layout timing closes the render-to-event window before a disabled
composer can receive input.
+             attachmentsInteractiveRef.current = attachmentsInteractive;
+         }, [attachmentsInteractive]);
+
```

```
-             (nextAttachments: readonly PromptAttachment[]): void => {
-
+             (nextAttachments: readonly PromptAttachment[]): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-             (computeNext: PromptAttachmentUpdater): void => {
-
+             (computeNext: PromptAttachmentUpdater): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-             [applyReplacedAttachments, attachments, onUpdateAttachments],
-         );
-
+             [applyReplacedAttachments, attachments, onUpdateAttachments],
+         );
+
+         /** Apply one popup-produced attachment only while the current composer policy
still allows edits. */
```

```
+         const handleAddAttachmentIfInteractive = useCallback(
+             (attachment: PromptAttachment): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+                 onAddAttachment(attachment);
+             },
+             [onAddAttachment],
+         );
+
──────────────────────────────────────────────────────────────────────────────
-         } = usePendingDocQueue(pendingDocQueueArgs);
-
+         } = usePendingDocQueue(pendingDocQueueArgs);
+
+         /** Remove one pending image only while attachment editing remains authorized.
*/
+         const handleRemovePendingImage = useCallback(
+             (clientId: string): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+                 removePendingImage(clientId);
+             },
+             [removePendingImage],
+         );
+
+         /** Remove one pending document only while attachment editing remains
authorized. */
+         const handleRemovePendingDoc = useCallback(
+             (clientId: string): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+                 removePendingDoc(clientId);
+             },
+             [removePendingDoc],
+         );
+
──────────────────────────────────────────────────────────────────────────────
-             ): Promise<void> => {
-                 if (viewerMode.isViewerMode) return;
-
+             ): Promise<void> => {
+                 if (!attachmentsInteractiveRef.current) return;
+
──────────────────────────────────────────────────────────────────────────────
-                 enqueuePendingImageFiles,
-                 indexPendingVisible,
-                 viewerMode.isViewerMode,
-
+                 enqueuePendingImageFiles,
+                 indexPendingVisible,
+
──────────────────────────────────────────────────────────────────────────────
-             async (imageId: string): Promise<void> => {
-
+             async (imageId: string): Promise<void> => {
+                 if (!attachmentsInteractiveRef.current) return;
+
──────────────────────────────────────────────────────────────────────────────
-             async (docIdRaw: string): Promise<void> => {
-
+             async (docIdRaw: string): Promise<void> => {
+                 if (!attachmentsInteractiveRef.current) return;
+
──────────────────────────────────────────────────────────────────────────────
-             (kind: PromptAttachment['kind']): void => {
-
+             (kind: PromptAttachment['kind']): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-             (source: AiDocRef['source'], files: readonly File[]): void => {
-                 if (viewerMode.isViewerMode) return;
-
+             (source: AiDocRef['source'], files: readonly File[]): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
-                 indexPendingVisible,
-                 viewerMode.isViewerMode,
-
+                 indexPendingVisible,
+
-         const handleClearAllAttachments = useCallback((): void => {
-
+         const handleClearAllAttachments = useCallback((): void => {
+             if (!attachmentsInteractiveRef.current) return;
+
-             > => {
-
+             > => {
+                 if (!attachmentsInteractiveRef.current) {
+                     throw new Error(
+                         'Prompt attachments are locked until live replay is armed.',
+                     );
+                 }
+
+
-             (e: React.ClipboardEvent<HTMLDivElement>): void => {
-                 if (viewerMode.isViewerMode) return;
-
+             (e: React.ClipboardEvent<HTMLDivElement>): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
-                 handleIngestImageFiles,
-                 indexPendingVisible,
-                 supportsVision,
-                 viewerMode.isViewerMode,
-
+                 handleIngestImageFiles,
+                 indexPendingVisible,
+                 supportsVision,
+
-             async (files: readonly File[]): Promise<void> => {
-                 if (viewerMode.isViewerMode) return;
-
+             async (files: readonly File[]): Promise<void> => {
+                 if (!attachmentsInteractiveRef.current) return;
+
-                 handleIngestImageFiles,
-                 indexPendingVisible,
-                 splitFilesByMime,
-                 supportsVision,
-                 viewerMode.isViewerMode,
-
+                 handleIngestImageFiles,
+                 indexPendingVisible,
+                 splitFilesByMime,
+                 supportsVision,
+
```

```
-         const handleDragEnter = useCallback(
-             (e: React.DragEvent<HTMLDivElement>): void => {
-                 if (viewerMode.isViewerMode) return;
-
+         const handleDragEnter = useCallback(
+             (e: React.DragEvent<HTMLDivElement>): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-                 isImageDragActiveRef,
-                 supportsVision,
-                 viewerMode.isViewerMode,
-
+                 isImageDragActiveRef,
+                 supportsVision,
+
```

```
-         const handleDragOver = useCallback(
-             (e: React.DragEvent<HTMLDivElement>): void => {
-                 if (viewerMode.isViewerMode) return;
-
+         const handleDragOver = useCallback(
+             (e: React.DragEvent<HTMLDivElement>): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-                 indexPendingVisible,
-                 supportsVision,
-                 viewerMode.isViewerMode,
-
+                 indexPendingVisible,
+                 supportsVision,
+
```

```
-             (e: React.DragEvent<HTMLDivElement>): void => {
-                 if (viewerMode.isViewerMode) return;
-
+             (e: React.DragEvent<HTMLDivElement>): void => {
+                 if (!attachmentsInteractiveRef.current) return;
+
```

```
-                 supportsVision,
-                 viewerMode.isViewerMode,
-
+                 supportsVision,
+
```

```
-                                     <AttachPopupMenu
-                                         enabled={true}
-
+                                     <AttachPopupMenu
+                                         enabled={attachmentsInteractive}
+
```

```
-                                         attachments={attachments}
-                                         onAddAttachment={onAddAttachment}
-
+                                         attachments={attachments}
+                                         onAddAttachment={
+                                             handleAddAttachmentIfInteractive
+                                         }
+
```

```
-                                     <div className={styles.attachListSlot}>
-                                         <AttachList
-                                             sessionLocator={sessionLocator}
```

```
-                                                turnId={
-                                                    effectiveAttachmentsTurnId
-                                                }
-                                                attachments={attachments}
-                                                pendingImages={pendingImages}
-                                                pendingDocs={pendingDocs}
-                                                thumbsByKey={thumbsByKey}
-                                                onRemoveAttachment={
-                                                    handleRemoveAttachment
-                                                }
-                                                onRemoveImageAttachment={
-                                                    handleRemoveImageAttachment
-                                                }
-                                                onRemoveDocAttachment={
-                                                    handleRemoveDocAttachment
-                                                }
-                                                onRemovePendingImageAttachment={
-                                                    removePendingImage
-                                                }
-                                                onRemovePendingDocAttachment={
-                                                    removePendingDoc
-                                                }
-                                            />
-
+                                            <div className={styles.attachListSlot}>
+                                                {attachmentsInteractive ? (
+                                                    <AttachList
+                                                        sessionLocator={
+                                                            sessionLocator
+                                                        }
+                                                        turnId={
+                                                            effectiveAttachmentsTurnId
+                                                        }
+                                                        attachments={attachments}
+                                                        pendingImages={
+                                                            pendingImages
+                                                        }
+                                                        pendingDocs={pendingDocs}
+                                                        thumbsByKey={thumbsByKey}
+                                                        onRemoveAttachment={
+                                                            handleRemoveAttachment
+                                                        }
+                                                        onRemoveImageAttachment={
+                                                            handleRemoveImageAttachment
+                                                        }
+                                                        onRemoveDocAttachment={
+                                                            handleRemoveDocAttachment
+                                                        }
+                                                        onRemovePendingImageAttachment={
+                                                            handleRemovePendingImage
+                                                        }
+                                                        onRemovePendingDocAttachment={
+                                                            handleRemovePendingDoc
+                                                        }
+                                                    />
+                                                ) : (
+                                                    <AttachList
+                                                        sessionLocator={
+                                                            sessionLocator
+                                                        }
+                                                        turnId={
+                                                            effectiveAttachmentsTurnId
+                                                        }
```

```
+                                                    attachments={attachments}
+                                                    thumbsByKey={thumbsByKey}
+                                                    removeEnabled={false}
+                                                />
+                                            )}
+
```

AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Copy
Review
Accept

patching file: *SidebarStore.rehydrate.manager.ts*
AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts
Copy
Review
Accept
PatchFull file

```
- import { sidebarQueryRunningBridge } from
'../../../messageBus/handlers/ia/SidebarQueryRunningBridge.js';
- import type { ChatSessionLocator } from
'../../../sharedlib/types/chatTypes/ChatSessionLocator.type.js';
-
+ import { sidebarQueryRunningBridge } from
'../../../messageBus/handlers/ia/SidebarQueryRunningBridge.js';
+ import type { SidebarActivityQueryItem } from
'../../../sharedlib/types/busTypes/ui/SidebarStore.js';
+ import type { ChatSessionLocator } from
'../../../sharedlib/types/chatTypes/ChatSessionLocator.type.js';
+ import type { LiveSessionRunRef } from
'../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
+
```

```
-     return JSON.stringify([locator.sessionId, locator.storageScopeRelPath]);
- }
-
+     return JSON.stringify([locator.sessionId, locator.storageScopeRelPath]);
+ }
+
+ /** Persisted fallback fields used only when runtime, store, and panel metadata are
unavailable. */
+ interface PersistedActivityMetadata {
+     readonly title?: string;
+     readonly openedAt?: number;
+ }
+
+ /** Fully resolved presentation metadata applied after the Activity list is rebuilt
atomically. */
+ interface ResolvedActivityMetadata {
+     readonly title: string;
+     readonly openedAt: number;
+     readonly liveQueryElapsedMs?: number;
+     readonly lastQueryElapsedMs?: number;
+     readonly usage?: SidebarActivityQueryItem['usage'];
+ }
+
+ /** Return one finite non-negative millisecond value or undefined. */
+ function normalizeActivityTimestamp(
+     value: number | undefined,
+ ): number | undefined {
+     return typeof value === 'number' && Number.isFinite(value) && value >= 0
+         ? Math.floor(value)
```

```
+         : undefined;
+ }
+
+ /** Compare full logical and execution identity without falling back to the active
locator pointer. */
+ function isSameRunRef(
+     left: LiveSessionRunRef | undefined,
+     right: LiveSessionRunRef | undefined,
+ ): boolean {
+     return (
+         left !== undefined &&
+         right !== undefined &&
+         left.turnId === right.turnId &&
+         left.sourceId === right.sourceId &&
+         isSameQueryActivityLocator(left.locator, right.locator)
+     );
+ }
+
+ /** Read one persisted title/open timestamp best-effort without making persistence an
Activity authority. */
+ async function readPersistedActivityMetadata(
+     persistence: ChatPersistence,
+     locator: ChatSessionLocator,
+ ): Promise<PersistedActivityMetadata> {
+     try {
+         const session = await persistence.reader.loadSessionFile(locator);
+         if (!session) return {};
+         const listItem = buildSidebarChatListItem({
+             session,
+             storageScopeRelPath: locator.storageScopeRelPath,
+         });
+         return {
+             title: listItem.title,
+             openedAt: normalizeActivityTimestamp(session.meta.createdAt),
+         };
+     } catch {
+         // Rehydration remains available for ephemeral and temporarily unreadable
chats.
+         return {};
+     }
+ }
+
-----------------------------------------------------------------------------------------
-         string,
-         { locator: ChatSessionLocator; title: string; openedAt: number }
-
+         string,
+         { locator: ChatSessionLocator; title: string }
+
-----------------------------------------------------------------------------------------
-             title,
-             openedAt: Date.now(),
-
+             title,
+
-----------------------------------------------------------------------------------------
-             title,
-         });
-     });
-
+             title,
+         });
+     });
+
```

```
+     // Snapshot every authority before clearing the projection. Runtime getSnapshot
intentionally reads
+     // the current item to preserve stable title, openedAt, usage, and elapsed values
across Sidebar reloads.
+     const snapshots = getLiveSessionServiceForProd().listActiveSnapshots();
+     const previousItemsByLocatorKey = new Map(
+         sidebarStore.sidebarContentState.activityState.queryState.items.map(
+             (item) => [locatorKey(item), item] as const,
+         ),
+     );
+     const projectionTargets = new Map<
+         string,
+         { locator: ChatSessionLocator; runRef?: LiveSessionRunRef }
+     >();
+     for (const open of openLocators.values()) {
+         projectionTargets.set(locatorKey(open.locator), {
+             locator: open.locator,
+         });
+     }
+     for (const snapshot of snapshots) {
+         projectionTargets.set(locatorKey(snapshot.runRef.locator), {
+             locator: snapshot.runRef.locator,
+             runRef: snapshot.runRef,
+         });
+     }
+
+     const rehydratedAtMs = Date.now();
+     const persistence = new ChatPersistence();
+     const metadataByLocatorKey = new Map<
+         string,
+         ResolvedActivityMetadata
+     >();
+
+     // Resolve the explicit priority chain before mutating the store:
+     // runtime → previous store → open panel → persistence → deterministic default.
+     await Promise.all(
+         [...projectionTargets.entries()].map(
+             async ([key, target]): Promise<void> => {
+                 const runtime = sidebarQueryRunningBridge.getSnapshot(
+                     target.locator,
+                 );
+                 const previous = previousItemsByLocatorKey.get(key);
+                 const open = openLocators.get(key);
+                 const persisted = await readPersistedActivityMetadata(
+                     persistence,
+                     target.locator,
+                 );
+                 const runtimeMatchesTarget = isSameRunRef(
+                     runtime?.runRef,
+                     target.runRef,
+                 );
+                 const previousRunRef =
+                     previous?.status === 'running'
+                         ? previous.runRef
+                         : undefined;
+                 const previousMatchesTarget = isSameRunRef(
+                     previousRunRef,
+                     target.runRef,
+                 );
+
+                 const title =
+                     runtime?.title?.trim() ||
+                     previous?.title.trim() ||
```

```
+                    open?.title.trim() ||
+                    persisted.title?.trim() ||
+                    getVisibleChatTitleOrUntitled(target.locator);
+                const openedAt =
+                    normalizeActivityTimestamp(runtime?.openedAt) ??
+                    normalizeActivityTimestamp(previous?.openedAt) ??
+                    normalizeActivityTimestamp(persisted.openedAt) ??
+                    rehydratedAtMs;
+
+                const usage = target.runRef
+                    ? runtimeMatchesTarget
+                        ? runtime?.usage
+                        : previousMatchesTarget
+                          ? previous?.usage
+                          : undefined
+                    : runtime?.usage ?? previous?.usage;
+                const liveQueryElapsedMs = target.runRef
+                    ? runtimeMatchesTarget
+                        ? runtime?.liveQueryElapsedMs
+                        : previousMatchesTarget &&
+                            previous?.status === 'running'
+                          ? previous.liveQueryElapsedMs
+                          : undefined
+                    : undefined;
+                const lastQueryElapsedMs = target.runRef
+                    ? runtimeMatchesTarget
+                        ? runtime?.lastQueryElapsedMs
+                        : previousMatchesTarget
+                          ? previous?.lastQueryElapsedMs
+                          : undefined
+                    : runtime?.lastQueryElapsedMs ??
+                      previous?.lastQueryElapsedMs ??
+                      (previous?.status === 'running'
+                          ? previous.liveQueryElapsedMs
+                          : undefined);
+
+                metadataByLocatorKey.set(key, {
+                    title,
+                    openedAt,
+                    ...(liveQueryElapsedMs === undefined
+                        ? {}
+                        : { liveQueryElapsedMs }),
+                    ...(lastQueryElapsedMs === undefined
+                        ? {}
+                        : { lastQueryElapsedMs }),
+                    ...(usage === undefined ? {} : { usage }),
+                });
+            },
+        ),
+    );
+
-    const projected = new Set<string>();
-    for (const snapshot of getLiveSessionServiceForProd().listActiveSnapshots()) {
-
+    const projected = new Set<string>();
+    for (const snapshot of snapshots) {
+

-            snapshot.runRef.locator,
-        );
-

+            snapshot.runRef.locator,
+        );
```

```
+         const metadata = metadataByLocatorKey.get(key);
+         const runtimeMatchesSnapshot = isSameRunRef(
+             runtime?.runRef,
+             snapshot.runRef,
+         );
+
```

```
-             sourceStartedAtMs: snapshot.sourceStartedAtMs,
-             title: open?.title,
-
+             sourceStartedAtMs: snapshot.sourceStartedAtMs,
+             title: metadata?.title,
+
```

```
-             stopRequested:
-                 snapshot.abortRequested || runtime?.stopRequested === true,
-
+             stopRequested:
+                 snapshot.abortRequested ||
+                 (runtimeMatchesSnapshot && runtime?.stopRequested === true),
+             openedAt: metadata?.openedAt,
+             liveQueryElapsedMs: metadata?.liveQueryElapsedMs,
+             lastQueryElapsedMs: metadata?.lastQueryElapsedMs,
+             usage: metadata?.usage,
+
```

```
-         if (projected.has(key)) continue;
-         await panelOpened(open);
-
+         if (projected.has(key)) continue;
+         const metadata = metadataByLocatorKey.get(key);
+         await panelOpened({
+             locator: open.locator,
+             title: metadata?.title,
+             openedAt: metadata?.openedAt ?? rehydratedAtMs,
+             lastQueryElapsedMs: metadata?.lastQueryElapsedMs,
+             usage: metadata?.usage,
+         });
+
```

AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts
Copy
Review
Accept

reading project: *useAiStream.hook.ts* [L195-L230], *useAiStream.hook.ts* [L590-L735], *useAiStream.hook.ts* [L850-L955], *PromptInput.tsx* [L1-L30], *PromptInput.tsx* [L360-L500], *PromptInput.tsx* [L1120-L1760], *PromptInput.tsx* [L2030-L2085], *SidebarStore.queryWindow.listeners.manager.ts* [L55-L310], *SidebarStore.rehydrate.manager.ts* [L1-L300]

patching file: *SidebarStore.rehydrate.manager.test.ts*
AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts
Copy
Review
Accept
PatchFull file

```
- import type * as vscode from 'vscode';
-
+ import type * as vscode from 'vscode';
+ import type { ChatSessionLocator } from
'../../../../sharedlib/types/chatTypes/ChatSessionLocator.type.js';
```

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

```
- const listActiveSnapshotsMock = jest.fn(() => [] as unknown[]);
-
+ const listActiveSnapshotsMock = jest.fn(() => [] as unknown[]);
+ const loadSessionFileMock = jest.fn<(locator: ChatSessionLocator) => Promise<
+     | {
+           meta: {
+               formatVersionNumber: number;
+               sessionId: string;
+               title: string;
+               createdAt: number;
+               updatedAt: number;
+               firstSendState: 'preSend' | 'postSend' | 'notApplicable';
+           };
+           messages: [];
+       }
+     | null
+ >>();
+ type RuntimeSnapshot = Readonly<{
+     runRef: LiveSessionRunRef;
+     ownership: 'source' | 'observer';
+     runId: number;
+     sourceStartedAtMs: number;
+     stopRequested: boolean;
+     panelState: 'open' | 'closed';
+     title?: string;
+     openedAt?: number;
+     liveQueryElapsedMs: number;
+     lastQueryElapsedMs?: number;
+     usage?: {
+         query_time: {
+             start_ts_utc: number;
+             end_ts_utc: number;
+             duration_ms: number;
+         };
+     };
+ }>;
+ const getSnapshotMock = jest.fn<
+     (locator: ChatSessionLocator) => RuntimeSnapshot | undefined
+ >();
+
```

```
-     }),
- );
-
+     }),
+ );
+ jest.unstable_mockModule(
+     '../../../../aiTools/orch/context/persistence/ChatPersistence.service.js',
+     () => ({
+         ChatPersistence: class ChatPersistenceMock {
+             public readonly reader = {
+                 loadSessionFile: loadSessionFileMock,
+             };
+         },
+     }),
+ );
+ jest.unstable_mockModule(
+     '../../../../messageBus/handlers/ia/SidebarQueryRunningBridge.js',
+     () => ({
+         sidebarQueryRunningBridge: {
```

```
+             getSnapshot: getSnapshotMock,
+         },
+     }),
+ );
+
_____________________________________________________________________________
-         listActiveSnapshotsMock.mockReturnValue([]);
-
+         listActiveSnapshotsMock.mockReturnValue([]);
+         loadSessionFileMock.mockReset();
+         loadSessionFileMock.mockResolvedValue(null);
+         getSnapshotMock.mockReset();
+         getSnapshotMock.mockReturnValue(undefined);
+
_____________________________________________________________________________
-         ).toEqual([]);
-     });
-
+         ).toEqual([]);
+     });
+
+     it('restores metadata in runtime, store, panel, persistence, then default priority
order', async () => {
+         const runtimeLocator = {
+             sessionId: 'f'.repeat(32),
+             storageScopeRelPath: '',
+         };
+         const panelLocator = {
+             sessionId: '1'.repeat(32),
+             storageScopeRelPath: '',
+         };
+         const runtimeRunRef = {
+             locator: runtimeLocator,
+             turnId: 'turn-runtime',
+             sourceId: 'source-runtime',
+         };
+         const panelRunRef = {
+             locator: panelLocator,
+             turnId: 'turn-panel',
+             sourceId: 'source-panel',
+         };
+         const usage = {
+             query_time: {
+                 start_ts_utc: 1,
+                 end_ts_utc: 2,
+                 duration_ms: 44,
+             },
+         };
+
+         sidebarStore.sidebarContentState.activityState.queryState.items = [
+             {
+                 status: 'running',
+                 ownership: 'source',
+                 panelState: 'open',
+                 sessionId: runtimeLocator.sessionId,
+                 storageScopeRelPath: runtimeLocator.storageScopeRelPath,
+                 title: 'Store title',
+                 runRef: runtimeRunRef,
+                 liveQueryElapsedMs: 50,
+                 lastQueryElapsedMs: 44,
+                 usage,
+                 stopRequested: false,
+                 updatedAt: 60,
+                 openedAt: 40,
```

```
+            },
+        ];
+        openedPanelsRegistry.set(
+            buildPromptPanelKey(runtimeLocator),
+            panel('🤖 Panel fallback'),
+        );
+        openedPanelsRegistry.set(
+            buildPromptPanelKey(panelLocator),
+            panel('🤖 Panel title'),
+        );
+        listActiveSnapshotsMock.mockReturnValue([
+            {
+                kind: 'record',
+                runRef: runtimeRunRef,
+                ownership: 'source',
+                sourceStartedAtMs: 10,
+                abortRequested: false,
+                sourceSettled: false,
+                handoffReady: false,
+            },
+            {
+                kind: 'record',
+                runRef: panelRunRef,
+                ownership: 'source',
+                sourceStartedAtMs: 20,
+                abortRequested: false,
+                sourceSettled: false,
+                handoffReady: false,
+            },
+        ]);
+        getSnapshotMock.mockImplementation((locator) => {
+            if (locator.sessionId !== runtimeLocator.sessionId) {
+                return undefined;
+            }
+            return {
+                runRef: runtimeRunRef,
+                ownership: 'source',
+                runId: 7,
+                sourceStartedAtMs: 10,
+                stopRequested: false,
+                panelState: 'open',
+                title: 'Runtime title',
+                openedAt: 30,
+                liveQueryElapsedMs: 55,
+                lastQueryElapsedMs: 44,
+                usage,
+            };
+        });
+        loadSessionFileMock.mockImplementation(async (locator) => ({
+            meta: {
+                formatVersionNumber: 1,
+                sessionId: locator.sessionId,
+                title: 'Persisted title',
+                createdAt: 70,
+                updatedAt: 80,
+                firstSendState: 'postSend',
+            },
+            messages: [],
+        }));
+
+        await rehydrateQueryActivities();
+
+        const runtimeItem =
```

```
+             sidebarStore.sidebarContentState.activityState.queryState.items.find(
+                 (item) => item.sessionId === runtimeLocator.sessionId,
+             );
+         const panelItem =
+             sidebarStore.sidebarContentState.activityState.queryState.items.find(
+                 (item) => item.sessionId === panelLocator.sessionId,
+             );
+         expect(runtimeItem).toEqual(
+             expect.objectContaining({
+                 title: 'Runtime title',
+                 openedAt: 30,
+                 liveQueryElapsedMs: 55,
+                 lastQueryElapsedMs: 44,
+                 usage,
+             }),
+         );
+         expect(panelItem).toEqual(
+             expect.objectContaining({
+                 title: 'Panel title',
+                 openedAt: 70,
+             }),
+         );
+     });
+
```

AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts

Copy

Review

Accept

patching file: *SidebarQueryRunningBridge.race.test.ts*

AICode/src/extension/src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts

Copy

Review

Accept

PatchFull file

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

```
- >(async () => {});
- const sourceSettledMock = jest.fn<(runRef: LiveSessionRunRef) => Promise<void>>(
-     async () => {},
- );
-
+ >(async () => {});
+ const sourceSettledMock = jest.fn<
+     (runRef: LiveSessionRunRef, finalElapsedMs?: number) => Promise<void>
+ >(async () => {});
+
```

```
- const observerSettledMock = jest.fn<
-     (runRef: LiveSessionRunRef) => Promise<void>
-
+ const observerSettledMock = jest.fn<
```

```
+     (runRef: LiveSessionRunRef, finalElapsedMs?: number) => Promise<void>
+
```

```
-
-     it('allocates synchronously before auxiliary publication and ignores stale
finalization', async () => {
-
+
+     afterEach(() => {
+         jest.restoreAllMocks();
+     });
+
+     it('allocates synchronously before auxiliary publication and ignores stale
finalization', async () => {
+
```

```
-         await service.transitionSourceSettled(RUN_B, second.runId);
-         expect(sourceSettledMock).toHaveBeenCalledWith(RUN_B);
-
+         await service.transitionSourceSettled(RUN_B, second.runId);
+         expect(sourceSettledMock).toHaveBeenCalledWith(
+             RUN_B,
+             expect.any(Number),
+         );
+
```

```
-
-     it('latches exact stopRequested once before later panel-close projection', () => {
-
+
+     it('latches exact stopRequested once before later panel-close projection', async ()
=> {
+
```

```
-         const service = new SidebarQueryActivityRuntimeService();
-         service.allocateStart({
-
+         const service = new SidebarQueryActivityRuntimeService();
+         const allocation = service.allocateStart({
+
```

```
-             expect.objectContaining({ stopRequested: true }),
-         );
-
+             expect.objectContaining({ stopRequested: true }),
+         );
+         await service.transitionSourceSettled(RUN_A, allocation.runId);
+
```

```
-
-         expect(observerSettledMock).toHaveBeenCalledWith(RUN_A);
-
+
+         expect(observerSettledMock).toHaveBeenCalledWith(
+             RUN_A,
+             expect.any(Number),
+         );
+
```

```
-         expect(service.getSnapshot(LOCATOR)).toBeUndefined();
-     });
-
+         expect(service.getSnapshot(LOCATOR)).toBeUndefined();
+     });
+
```

```
+     it('publishes the exact final wall-clock elapsed value instead of the previous
timer tick', async () => {
+         const service = new SidebarQueryActivityRuntimeService();
+         const source = service.allocateStart({
+             runRef: RUN_A,
+             ownership: 'source',
+             sourceStartedAtMs: 1_000,
+             panelState: 'open',
+         });
+         jest.spyOn(Date, 'now').mockReturnValue(2_750);
+
+         await service.transitionSourceSettled(RUN_A, source.runId);
+
+         expect(sourceSettledMock).toHaveBeenCalledWith(RUN_A, 1_750);
+         expect(service.getSnapshot(LOCATOR)).toBeUndefined();
+     });
+
```

AICode/src/extension/src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts
Copy
Review
Accept

🔧 ✅ *reading project*: 📄 *VscButton.tsx* [L1-L180], 🔎 e *"Problems attachment testid"*, 🔎 e *"Textarea submit test id"*
🔧 ✅ *scanning git*: ✂ *"Testids VscTextarea"*
🔧 ✅ *reading project*: 🔎 h *"DOM bouton textarea"*
🔧 ✅ *patching file*: 📄 *PromptInput.batchLiveViewerMode.test.tsx*

AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Copy
Review
Accept
PatchFull file

```
- /** @jest-environment jsdom */
-
- import { describe, it } from '@jest/globals';
-
- import type { PromptLiveSessionViewModel } from
'../../liveSession/core/PromptLiveSession.types.js';
-
- describe('PromptInput live-session surfaces', () => {
-     it('keeps EasyAgents read-only and unlocks normalPrompt editing only after replay
arming', () => {
-         const easyAgents: PromptLiveSessionViewModel = {
-             isViewerMode: true,
-             showComposer: false,
-             composerInteractive: false,
-             sendEnabled: false,
-             historyInteractionsEnabled: false,
-             cleanupOwned: false,
-             transportStopAvailable: true,
-             banner: {
-                 showStopAction: true,
-                 stopEnabled: true,
-                 stopActionLabel: 'Stop line',
-                 showRetryAction: false,
-                 retryEnabled: false,
-             },
-         };
-         const normalPromptStreaming: PromptLiveSessionViewModel = {
-             isViewerMode: false,
-             showComposer: true,
```

```
-             composerInteractive: true,
-             sendEnabled: false,
-             historyInteractionsEnabled: false,
-             cleanupOwned: false,
-             transportStopAvailable: true,
-             banner: {
-                 showStopAction: false,
-                 stopEnabled: false,
-                 showRetryAction: false,
-                 retryEnabled: false,
-             },
-         };
-         const normalPromptReplayFailed: PromptLiveSessionViewModel = {
-             ...normalPromptStreaming,
-             composerInteractive: false,
-             banner: {
-                 showStopAction: false,
-                 stopEnabled: true,
-                 showRetryAction: true,
-                 retryEnabled: true,
-                 retryActionLabel: 'Retry replay',
-             },
-         };
-         expect(easyAgents.showComposer).toBe(false);
-         expect(normalPromptStreaming).toEqual(
-             expect.objectContaining({
-                 showComposer: true,
-                 composerInteractive: true,
-                 sendEnabled: false,
-             }),
-         );
-         expect(normalPromptReplayFailed).toEqual(
-             expect.objectContaining({ composerInteractive: false }),
-         );
-         expect(normalPromptStreaming.banner?.showStopAction).toBe(false);
-     });
- });
-
+ /** @jest-environment jsdom */
+
+ import { beforeEach, describe, it, jest } from '@jest/globals';
+ import { fireEvent, render, screen, waitFor } from '@testing-library/react';
+ import { useCallback, useMemo, useState } from 'react';
+
+ import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
+ import type { PromptViewSelection } from '../../Prompt/usePromptViewMode.hook.js';
+ import type { PromptLiveSessionViewModel } from
'../../liveSession/core/PromptLiveSession.types.js';
+ import type { AiDocRef } from
'../../../../sharedlib/types/promptAttachmentTypes/AiDocRef.type.js';
+
+ const SESSION_ID = '0123456789abcdef0123456789abcdef';
+ const TURN_ID = 'turn-live-session';
+ const VIEW_SELECTION: PromptViewSelection = ['chatView'];
+
+ const EASY_AGENTS_VIEWER: PromptLiveSessionViewModel = {
+     isViewerMode: true,
+     showComposer: false,
+     composerInteractive: false,
+     sendEnabled: false,
+     historyInteractionsEnabled: false,
+     cleanupOwned: false,
+     transportStopAvailable: true,
```

```
+     banner: {
+         showStopAction: true,
+         stopEnabled: true,
+         stopActionLabel: 'Stop line',
+         showRetryAction: false,
+         retryEnabled: false,
+     },
+ };
+
+ const NORMAL_PROMPT_OPENING: PromptLiveSessionViewModel = {
+     isViewerMode: false,
+     showComposer: true,
+     composerInteractive: false,
+     sendEnabled: false,
+     historyInteractionsEnabled: false,
+     cleanupOwned: false,
+     transportStopAvailable: true,
+     banner: {
+         showStopAction: false,
+         stopEnabled: false,
+         showRetryAction: false,
+         retryEnabled: false,
+     },
+ };
+
+ const NORMAL_PROMPT_STREAMING: PromptLiveSessionViewModel = {
+     ...NORMAL_PROMPT_OPENING,
+     composerInteractive: true,
+ };
+
+ const NORMALIZED_DOC = {
+     name: 'notes.txt',
+     byteSize: 5,
+     sourceMime: 'text/plain',
+     sha256: 'a'.repeat(64),
+     base64: 'aGVsbG8=',
+ };
+
+ const DOC_REF: AiDocRef = {
+     id: NORMALIZED_DOC.sha256,
+     sha256: NORMALIZED_DOC.sha256,
+     source: 'file',
+     name: NORMALIZED_DOC.name,
+     byteSize: NORMALIZED_DOC.byteSize,
+     sourceMime: NORMALIZED_DOC.sourceMime,
+     text: {
+         fileName: `turn-live-session-doc-${NORMALIZED_DOC.sha256}-notes.txt.txt`,
+         mime: 'text/plain',
+     },
+ };
+
+ const callLocalApiMock = jest.fn<
+     (url: string, args?: unknown) => Promise<unknown>
+ >();
+ const normalizeDocFileToIngestItemMock = jest.fn<
+     (args: { file: File }) => Promise<typeof NORMALIZED_DOC>
+ >();
+
+ jest.unstable_mockModule('../../../../messageBus/callLocalApi.util.js', () => ({
+     callLocalApi: callLocalApiMock,
+ }));
+ jest.unstable_mockModule(
+     '../../../../messageBus/useBusListener.hook.js',
```

```
+     () => ({ useBusListener: () => {} }),
+ );
+ jest.unstable_mockModule(
+     '../attachments/normalizeDocFileToIngestItem.util.js',
+     () => ({
+         normalizeDocFileToIngestItem: normalizeDocFileToIngestItemMock,
+     }),
+ );
+
+ const { PromptInput } = await import('../PromptInput.js');
+
+ /** Mount the real PromptInput with controlled attachments and one switchable live-
session policy. */
+ function Harness({
+     viewModel,
+ }: {
+     viewModel: PromptLiveSessionViewModel;
+ }) {
+     const [promptValue, setPromptValue] = useState('next prompt');
+     const [attachments, setAttachments] = useState<readonly PromptAttachment[]>(
+         [],
+     );
+     const dbSettingsExtUiPrompt = useMemo(
+         () => ({
+             autoScroll: true,
+             paginationEnabled: false,
+             queryDoneSoundEnabled: false,
+         }),
+         [],
+     );
+     const handleSend = useCallback((_text: string): void => {}, []);
+     const handleStop = useCallback(async (): Promise<void> => {}, []);
+     const handleBeforeSubmit = useCallback((): void => {}, []);
+     const handleIndexAction = useCallback((): void => {}, []);
+     const handleSettingsChange = useCallback((): void => {}, []);
+     const handleViewSelectionChange = useCallback(
+         (_next: PromptViewSelection): void => {},
+         [],
+     );
+     const handleComposerMaximizedChange = useCallback(
+         (_next: boolean): void => {},
+         [],
+     );
+     const handleAddAttachment = useCallback(
+         (attachment: PromptAttachment): void => {
+             setAttachments((current) => [...current, attachment]);
+         },
+         [],
+     );
+     const handleRemoveAttachment = useCallback(
+         (kind: PromptAttachment['kind']): void => {
+             setAttachments((current) =>
+                 current.filter((attachment) => attachment.kind !== kind),
+             );
+         },
+         [],
+     );
+     const handleClearAttachments = useCallback((): void => {
+         setAttachments([]);
+     }, []);
+     const handleReplaceAttachments = useCallback(
+         (next: readonly PromptAttachment[]): void => {
+             setAttachments([...next]);
```

```
+        },
+        [],
+    );
+    const ensureNextTurnId = useCallback((): string => TURN_ID, []);
+    const handleRetry = useCallback((): void => {}, []);
+
+    return (
+        <PromptInput
+            sessionLocator={{
+                sessionId: SESSION_ID,
+                storageScopeRelPath: '',
+            }}
+            supportsVision={true}
+            isLocked={false}
+            onSend={handleSend}
+            dbSettingsExtUiPrompt={dbSettingsExtUiPrompt}
+            onChangeDbSettingsExtUiPrompt={handleSettingsChange}
+            running={true}
+            stopEnabled={true}
+            onBeforeSubmitUi={handleBeforeSubmit}
+            onStop={handleStop}
+            indexPendingVisible={false}
+            onIndexPendingForceNow={handleIndexAction}
+            onIndexPendingClose={handleIndexAction}
+            promptValue={promptValue}
+            onChangePromptValue={setPromptValue}
+            attachments={attachments}
+            onAddAttachment={handleAddAttachment}
+            onRemoveAttachment={handleRemoveAttachment}
+            onClearAllAttachments={handleClearAttachments}
+            onReplaceAttachments={handleReplaceAttachments}
+            viewSelection={VIEW_SELECTION}
+            onChangeViewSelection={handleViewSelectionChange}
+            logsOnly={false}
+            composerMaximized={false}
+            onChangeComposerMaximized={handleComposerMaximizedChange}
+            attachmentsTurnId={TURN_ID}
+            ensureNextTurnId={ensureNextTurnId}
+            liveSessionViewModel={viewModel}
+            onRetryViewerHandoff={handleRetry}
+        />
+    );
+ }
+
+ describe('PromptInput live-session surfaces', () => {
+    beforeEach(() => {
+        callLocalApiMock.mockReset();
+        callLocalApiMock.mockImplementation(async (url) => {
+            if (url === '/api/ai/orch/attachments/draftAddDoc') {
+                return {
+                    doc: DOC_REF,
+                    draftAttachments: [{ kind: 'doc', doc: DOC_REF }],
+                };
+            }
+            return {};
+        });
+        normalizeDocFileToIngestItemMock.mockReset();
+        normalizeDocFileToIngestItemMock.mockResolvedValue(NORMALIZED_DOC);
+        window.focus = jest.fn<() => void>();
+    });
+
+    it('keeps EasyAgents read-only with one banner-owned Stop action', () => {
+        render(<Harness viewModel={EASY_AGENTS_VIEWER} />);
```

```
+
+         expect(screen.getByTestId('promptInputViewerBanner')).toBeInTheDocument();
+         expect(
+             screen.getByTestId('promptInputViewerStopButton'),
+         ).toBeInTheDocument();
+         expect(screen.queryByTestId('promptAttachButton')).toBeNull();
+     });
+
+     it('locks every attachment entry point before replay arming and unlocks them
afterwards', async () => {
+         const rendered = render(
+             <Harness viewModel={NORMAL_PROMPT_OPENING} />,
+         );
+         const attachButton = screen.getByTestId('promptAttachButton');
+         const fileInput = screen.getByTestId('promptAttachFileInput');
+         const textarea = document.querySelector('textarea');
+         if (!textarea) throw new Error('Expected the Prompt textarea.');
+
+         expect(textarea).toBeDisabled();
+         expect(attachButton).toBeDisabled();
+         expect(screen.queryByTestId('promptInputViewerStopButton')).toBeNull();
+
+         // A programmatic file-input change must remain inert even though the hidden
input still exists.
+         const file = new File(['hello'], 'notes.txt', { type: 'text/plain' });
+         fireEvent.change(fileInput, { target: { files: [file] } });
+         await Promise.resolve();
+         expect(normalizeDocFileToIngestItemMock).not.toHaveBeenCalled();
+         expect(callLocalApiMock).not.toHaveBeenCalled();
+
+         rendered.rerender(
+             <Harness viewModel={NORMAL_PROMPT_STREAMING} />,
+         );
+         expect(textarea).not.toBeDisabled();
+         expect(attachButton).not.toBeDisabled();
+         expect(screen.queryByTestId('promptInputViewerStopButton')).toBeNull();
+
+         fireEvent.change(fileInput, { target: { files: [file] } });
+         await waitFor(() => {
+             expect(normalizeDocFileToIngestItemMock).toHaveBeenCalledTimes(1);
+         });
+     });
+ });
+
--------------------------------------------------------------------------------
- import { useCallback, useMemo, useState } from 'react';
-
+ import { useCallback, useMemo, useState } from 'react';
+ import type { JSX } from 'react';
+
--------------------------------------------------------------------------------
- const VIEW_SELECTION: PromptViewSelection = ['chatView'];
-
+ const VIEW_SELECTION: PromptViewSelection = ['chatView'];
+ const SESSION_LOCATOR = {
+     sessionId: SESSION_ID,
+     storageScopeRelPath: '',
+ } as const;
+
--------------------------------------------------------------------------------
-     viewModel: PromptLiveSessionViewModel;
- }) {
-
+     viewModel: PromptLiveSessionViewModel;
+ }): JSX.Element {
```

```
+
```

```
-         <PromptInput
-             sessionLocator={{
-                 sessionId: SESSION_ID,
-                 storageScopeRelPath: '',
-             }}
-
+         <PromptInput
+             sessionLocator={SESSION_LOCATOR}
+
```

```
-
-         rendered.rerender(
-             <Harness viewModel={NORMAL_PROMPT_STREAMING} />,
-         );
-         expect(textarea).not.toBeDisabled();
-         expect(attachButton).not.toBeDisabled();
-
+
+         rendered.rerender(<Harness viewModel={NORMAL_PROMPT_STREAMING} />);
+         const unlockedTextarea = document.querySelector('textarea');
+         if (!unlockedTextarea) {
+             throw new Error('Expected the unlocked Prompt textarea.');
+         }
+         expect(unlockedTextarea).not.toBeDisabled();
+         expect(screen.getByTestId('promptAttachButton')).not.toBeDisabled();
+
```

```
- import { beforeEach, describe, it, jest } from '@jest/globals';
- import { fireEvent, render, screen, waitFor } from '@testing-library/react';
-
+ import { beforeEach, describe, it, jest } from '@jest/globals';
+ import { act, fireEvent, render, screen, waitFor } from '@testing-library/react';
+
```

```
-
-         rendered.rerender(<Harness viewModel={NORMAL_PROMPT_STREAMING} />);
-
+
+         await act(async () => {
+             rendered.rerender(<Harness viewModel={NORMAL_PROMPT_STREAMING} />);
+         });
+
```

```
-         mime: 'text/plain',
-     },
- };
-
+         mime: 'text/plain',
+     },
+ };
+ const PROBLEMS_ATTACHMENT: PromptAttachment = {
+     kind: 'problems',
+     problems: {
+         id: 'b'.repeat(64),
+         count: 1,
+     },
+ };
+
```

```
-     const [attachments, setAttachments] = useState<readonly PromptAttachment[]>(
-         [],
-
+     const [attachments, setAttachments] = useState<readonly PromptAttachment[]>(
+         [PROBLEMS_ATTACHMENT],
```

```
+
```

```
-         expect(attachButton).toBeDisabled();
-
+         expect(attachButton).toBeDisabled();
+         expect(screen.queryByTestId('promptAttachmentRemove')).toBeNull();
+
```

```
-         expect(screen.getByTestId('promptAttachButton')).not.toBeDisabled();
-
+         expect(screen.getByTestId('promptAttachButton')).not.toBeDisabled();
+         expect(
+             screen.getByTestId('promptAttachmentRemove'),
+         ).toBeInTheDocument();
+
```

AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Copy
Review
Accept

patching file: *Prompt.liveSessionBootstrap.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.liveSessionBootstrap.test.tsx
Copy
Review
Accept
PatchFull file

```
- /** @jest-environment jsdom */
-
- import { describe, it } from '@jest/globals';
-
- import { dispatchPromptLiveSessionBootstrap } from
'../../liveSession/adapters/PromptLiveSessionBootstrap.adapter.js';
-
- describe('Prompt live-session outer bootstrap boundary', () => {
-     it('uses one raw snapshot, rejects duplicates, and marks every explicit live
query', () => {
-         const sid = 'a'.repeat(32);
-         const duplicate = dispatchPromptLiveSessionBootstrap(
-             `sessionId=${sid}
&storageScopeRelPath=&openMode=liveSession&openMode=liveSession`,
-         );
-         expect(duplicate).toEqual(
-             expect.objectContaining({
-                 kind: 'invalid',
-                 explicitLiveQuery: true,
-             }),
-         );
-         const classic = dispatchPromptLiveSessionBootstrap(
-             `sessionId=${sid}&storageScopeRelPath=&autoRunType=prompt`,
-         );
-         expect(classic).toEqual(
-             expect.objectContaining({
-                 kind: 'classic',
-                 explicitLiveQuery: false,
-             }),
-         );
-     });
- });
-
+ /** @jest-environment jsdom */
+
+ import type { JSX } from 'react';
```

```
+ import { afterEach, beforeEach, describe, it, jest } from '@jest/globals';
+ import { render, screen, waitFor } from '@testing-library/react';
+
+ const lifecycleOrder: string[] = [];
+ const useInitPageMock = jest.fn<(htmlPage: 'prompt') => void>();
+ const showVscAlertDialogMock = jest.fn<
+     (args: {
+         icon: 'error';
+         title: string;
+         labelMessage: string;
+     }) => Promise<void>
+ >();
+ const closeSelfMock = jest.fn<() => Promise<void>>();
+ const usePromptRunControllerMock = jest.fn(() => {
+     throw new Error('Invalid bootstrap must not mount Prompt runtime.');
+ });
+
+ jest.unstable_mockModule('../../../../lib/useInitPage.hook.js', () => ({
+     useInitPage: useInitPageMock,
+ }));
+ jest.unstable_mockModule(
+     '../../../../vsc-design-system/popups/VscAlertDialog.js',
+     () => ({ showVscAlertDialog: showVscAlertDialogMock }),
+ );
+ jest.unstable_mockModule('../../../../messageBus/closeSelf.util.js', () => ({
+     closeSelf: closeSelfMock,
+ }));
+ jest.unstable_mockModule('../usePromptRunController.hook.js', () => ({
+     usePromptRunController: usePromptRunControllerMock,
+ }));
+ jest.unstable_mockModule(
+     '../../PromptHydrationLine/PromptHydrationLine.js',
+     () => ({
+         PromptHydrationLine: (): JSX.Element => (
+             <div data-testid="invalidPromptBootstrap" />
+         ),
+     }),
+ );
+
+ const { Prompt } = await import('../Prompt.js');
+
+ /** Install one duplicate-sensitive explicit live query for the outer Prompt bootstrap.
*/
+ function installInvalidLiveQuery(): void {
+     const sessionId = 'a'.repeat(32);
+     Object.defineProperty(globalThis, '__AICODE_ROUTER_QUERY_STRING__', {
+         configurable: true,
+         value: `sessionId=${sessionId}
&storageScopeRelPath=&openMode=liveSession&openMode=liveSession`,
+     });
+ }
+
+ describe('Prompt live-session outer bootstrap boundary', () => {
+     beforeEach(() => {
+         lifecycleOrder.length = 0;
+         useInitPageMock.mockImplementation(() => {
+             lifecycleOrder.push('init');
+         });
+         showVscAlertDialogMock.mockImplementation(async () => {
+             lifecycleOrder.push('dialog');
+         });
+         closeSelfMock.mockImplementation(async () => {
+             lifecycleOrder.push('close');
```

```
+         });
+         usePromptRunControllerMock.mockClear();
+         installInvalidLiveQuery();
+     });
+
+     afterEach(() => {
+         Reflect.deleteProperty(globalThis, '__AICODE_ROUTER_QUERY_STRING__');
+         jest.restoreAllMocks();
+     });
+
+     it('initializes the bus-capable page boundary, skips runtime, explains the error,
and closes', async () => {
+         render(<Prompt />);
+
+         expect(screen.getByTestId('invalidPromptBootstrap')).toBeInTheDocument();
+         await waitFor(() => expect(closeSelfMock).toHaveBeenCalledTimes(1));
+         expect(usePromptRunControllerMock).not.toHaveBeenCalled();
+         expect(lifecycleOrder).toEqual(['init', 'dialog', 'close']);
+     });
+
+     it('still closes the invalid panel when the explanatory dialog itself fails', async
() => {
+         jest.spyOn(console, 'error').mockImplementation(() => {});
+         showVscAlertDialogMock.mockImplementationOnce(async () => {
+             lifecycleOrder.push('dialog');
+             throw new Error('dialog failed');
+         });
+
+         render(<Prompt />);
+
+         await waitFor(() => expect(closeSelfMock).toHaveBeenCalledTimes(1));
+         expect(lifecycleOrder).toEqual(['init', 'dialog', 'close']);
+         expect(usePromptRunControllerMock).not.toHaveBeenCalled();
+     });
+ });
+
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.liveSessionBootstrap.test.tsx

Copy

Review

Accept

patching file: *useAiStream.hook.liveSessionObserver.test.ts*

AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts

Copy

Review

Accept

PatchFull file

```
- import { describe, it } from '@jest/globals';
-
+ /** @jest-environment jsdom */
+
+ import { afterEach, beforeEach, describe, it, jest } from '@jest/globals';
+ import React from 'react';
+ import { act } from 'react';
+ import { createRoot } from 'react-dom/client';
+ import { waitFor } from '@testing-library/react';
+
+ import type { AiStreamedChunk } from
'../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+ import type {
+     AiStreamAttemptIdentity,
```

```
+     AiStreamStartResult,
+ } from '../AiStreamStartArgs.type.js';
+
+ type StreamListenerHandlers = {
+     onData: (chunk: AiStreamedChunk) => void;
+     onComplete: () => void;
+     onError: (error: unknown) => void;
+ };
+
+ type QueueRunMerged = (
+     chunk: AiStreamedChunk,
+     attemptIdentity: AiStreamAttemptIdentity,
+ ) => Promise<void> | void;
+
+ type StreamRegistration = {
+     args: unknown;
+     handlers: StreamListenerHandlers;
+ };
+
+ const queuedChunks: Array<{
+     chunk: AiStreamedChunk;
+     attemptIdentity: AiStreamAttemptIdentity;
+ }> = [];
+ const registrationsByChannel = new Map<string, StreamRegistration>();
+ const callLocalApiMock = jest.fn<
+     (url: string, args: unknown) => Promise<unknown>
+ >();
+ let queueRunMerged: QueueRunMerged | undefined;
+
+ jest.unstable_mockModule('../callLocalApi.util.js', () => ({
+     callLocalApi: callLocalApiMock,
+ }));
+ jest.unstable_mockModule('../useAiStreamQueue.hook.js', () => ({
+     useSerializedStreamQueue: (options: { runMerged: QueueRunMerged }) => {
+         queueRunMerged = options.runMerged;
+         return {
+             enqueueChunk: () => {
+                 throw new Error('Exact attempt identity is mandatory.');
+             },
+             enqueueChunkForAttempt: (
+                 chunk: AiStreamedChunk,
+                 attemptIdentity: AiStreamAttemptIdentity,
+             ): void => {
+                 queuedChunks.push({ chunk, attemptIdentity });
+             },
+             whenIdle: async (): Promise<void> => {},
+             waitForCommit: async (): Promise<void> => {},
+         };
+     },
+ }));
+ jest.unstable_mockModule(
+     '../usePromptStreamDispatchGateQueue.hook.js',
+     () => ({
+         usePromptStreamDispatchGateQueue: (options: {
+             dispatchMerged: QueueRunMerged;
+         }) => ({
+             enqueueMerged: async (): Promise<void> => {
+                 throw new Error('Exact attempt identity is mandatory.');
+             },
+             enqueueMergedForAttempt: async (
+                 chunk: AiStreamedChunk,
+                 attemptIdentity: AiStreamAttemptIdentity,
+             ): Promise<void> => {
```

```
+                 await options.dispatchMerged(chunk, attemptIdentity);
+             },
+             whenIdle: async (): Promise<void> => {},
+             resetInternalState: (): void => {},
+         }),
+     }),
+ );
+ jest.unstable_mockModule('../useStreamListener.hook.js', () => ({
+     useStreamListener: (
+         channel: string,
+         args: unknown,
+         handlers: StreamListenerHandlers,
+     ): void => {
+         if (args !== undefined) {
+             registrationsByChannel.set(channel, { args, handlers });
+         }
+     },
+ }));
+ jest.unstable_mockModule('../useAiStreamRawLiveLogs.hook.js', () => ({
+     useAiStreamRawLiveLogs: () => ({
+         rawLiveLogs: [],
+         onRawLiveReceived: (): void => {},
+         reset: (): void => {},
+     }),
+ }));
+ jest.unstable_mockModule(
+     '../useAiStreamSyntaxErrorTagOpenerSanitizer.hook.js',
+     () => ({
+         useAiStreamSyntaxErrorTagOpenerSanitizer: () => ({
+             sanitize: (value: string): string => value,
+             beginLogicalLine: (): void => {},
+             reset: (): void => {},
+         }),
+     }),
+ );
+ jest.unstable_mockModule(
+     '../useAiStreamSyntaxErrorTagNoAttributesSanitizer.hook.js',
+     () => ({
+         useAiStreamSyntaxErrorTagNoAttributesSanitizer: () => ({
+             sanitize: (value: string): string => value,
+             reset: (): void => {},
+         }),
+     }),
+ );
+
+ const { useAiStream } = await import('../useAiStream.hook.js');
+ type HarnessApi = ReturnType<typeof useAiStream>;
+ const harnessApiRef: { current: HarnessApi | null } = { current: null };
+
+ /** Mount the real transport hook while exposing its stable API to the test body. */
+ function Harness(): React.ReactElement {
+     const api = useAiStream('a'.repeat(32));
+     harnessApiRef.current = api;
+     return React.createElement('div');
+ }
+
--------------------------------------------------------------------------------
-
- import type { AiStreamStartArgs } from '../AiStreamStartArgs.type.js';
-
+
+ const RUN_REF = {
+     locator: {
+         sessionId: 'a'.repeat(32),
```

```
+         storageScopeRelPath: '',
+     },
+     turnId: 'turn',
+     sourceId: 'source',
+ } as const;
+
-------------------------------------------------------------------------------
- describe('useAiStream live-session observer contract', () => {
-     it('carries full runRef, replay mode, and exact Stop policy', () => {
-         const args: AiStreamStartArgs = {
-             kind: 'liveSessionObserver',
-             runRef: {
-                 locator: {
-                     sessionId: 'a'.repeat(32),
-                     storageScopeRelPath: '',
-
+ describe('useAiStream live-session observer contract', () => {
+     let container: HTMLDivElement;
+     let root: ReturnType<typeof createRoot>;
+
+     beforeEach(() => {
+         queuedChunks.length = 0;
+         registrationsByChannel.clear();
+         queueRunMerged = undefined;
+         harnessApiRef.current = null;
+         callLocalApiMock.mockReset();
+         callLocalApiMock.mockResolvedValue({ accepted: true });
+         container = document.createElement('div');
+         document.body.appendChild(container);
+         root = createRoot(container);
+     });
+
+     afterEach(() => {
+         act(() => root.unmount());
+         container.remove();
+         jest.restoreAllMocks();
+     });
+
+     it('routes the exact source observer and consumes Stop synchronously before the
first await', async () => {
+         await act(async () => {
+             root.render(React.createElement(Harness));
+         });
+         const api = harnessApiRef.current;
+         if (!api) throw new Error('Hook harness was not initialized.');
+
+         let startResult: AiStreamStartResult | undefined;
+         act(() => {
+             startResult = api.start({
+                 kind: 'liveSessionObserver',
+                 runRef: RUN_REF,
+                 viewerLeaseId: 'lease',
+                 replayMode: 'continueAssistant',
+                 stopPolicy: 'source',
+             });
+         });
+         expect(startResult).toEqual(
+             expect.objectContaining({
+                 accepted: true,
+                 runRef: RUN_REF,
+             }),
+         );
+         expect(
+             registrationsByChannel.get('proxylocalapi:liveSessionStream')
```

```
+                ?.args,
+        ).toEqual({
+            sessionId: RUN_REF.locator.sessionId,
+            storageScopeRelPath: RUN_REF.locator.storageScopeRelPath,
+            turnId: RUN_REF.turnId,
+            sourceId: RUN_REF.sourceId,
+            viewerLeaseId: 'lease',
+        });
+
+        let resolveAbort: ((value: unknown) => void) | undefined;
+        callLocalApiMock.mockImplementationOnce(
+            async () =>
+                await new Promise((resolve) => {
+                    resolveAbort = resolve;
+                }),
+        );
+        const activeApi = harnessApiRef.current;
+        if (!activeApi) throw new Error('Active hook API was not published.');
+        let firstStop: Promise<void> | undefined;
+        act(() => {
+            firstStop = activeApi.stop();
+            void activeApi.stop();
+        });
+
+        expect(callLocalApiMock).toHaveBeenCalledTimes(1);
+        expect(callLocalApiMock).toHaveBeenCalledWith(
+            '/api/ai/orch/liveSession/abortSource',
+            {
+                sessionId: RUN_REF.locator.sessionId,
+                storageScopeRelPath: RUN_REF.locator.storageScopeRelPath,
+                turnId: RUN_REF.turnId,
+                sourceId: RUN_REF.sourceId,
+            },
+        );
+        resolveAbort?.({ accepted: true });
+        await act(async () => {
+            await firstStop;
+        });
+        expect(harnessApiRef.current?.transportStopAvailable).toBe(false);
+    });
+
+    it('keeps one source terminal when a later transport error finalizes the attempt',
async () => {
+        jest.spyOn(console, 'error').mockImplementation(() => {});
+        await act(async () => {
+            root.render(React.createElement(Harness));
+        });
+        const api = harnessApiRef.current;
+        if (!api) throw new Error('Hook harness was not initialized.');
+        let startResult: AiStreamStartResult | undefined;
+        act(() => {
+            startResult = api.start({
+                kind: 'liveSessionObserver',
+                runRef: RUN_REF,
+                viewerLeaseId: 'lease',
+                replayMode: 'replaceTurnAssistants',
+                stopPolicy: 'viewerLease',
+            });
+        });
+        if (!startResult?.accepted) {
+            throw new Error('Expected the observer start to be accepted.');
+        }
+        const registration = registrationsByChannel.get(
```

```
+            'proxylocalapi:liveSessionStream',
+        );
+        const runner = queueRunMerged;
+        if (!registration || !runner) {
+            throw new Error('Observer listener/queue was not registered.');
+        }
+        const terminalChunk: AiStreamedChunk = {
+            turnId: RUN_REF.turnId,
+            type: 'meta',
+            value: {
+                metaType: 'error',
+                value: {
+                    reason: 'system_error',
+                    message: 'source error',
+                    details: 'source details',
+
--------------------------------------------------------------------------------
-                },
-                turnId: 'turn',
-                sourceId: 'source',
-
+                },
+
--------------------------------------------------------------------------------
-            },
-            viewerLeaseId: 'lease',
-            replayMode: 'continueAssistant',
-            stopPolicy: 'source',
-
+            },
+
--------------------------------------------------------------------------------
-        };
-        expect(args).toEqual(
-            expect.objectContaining({
-                stopPolicy: 'source',
-                runRef: expect.objectContaining({ sourceId: 'source' }),
-            }),
-
+        };
+
+        act(() => registration.handlers.onData(terminalChunk));
+        const queuedTerminal = queuedChunks[0];
+        if (!queuedTerminal) throw new Error('Terminal chunk was not queued.');
+        await act(async () => {
+            await runner(
+                queuedTerminal.chunk,
+                queuedTerminal.attemptIdentity,
+            );
+        });
+        act(() =>
+            registration.handlers.onError(
+                new Error('transport failed after terminal'),
+            ),
+
--------------------------------------------------------------------------------
-            ),
-        );
-
+            ),
+        );
+        await waitFor(() => {
+            expect(harnessApiRef.current?.outcome).toBe('transportError');
+        });
+        expect(queuedChunks).toHaveLength(1);
```

```
+         expect(harnessApiRef.current?.lastMetaChunk).toEqual(terminalChunk);
+         expect(harnessApiRef.current?.terminalDataSeen).toBe(true);
+
```

AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
Copy
Review
Accept

reading project: *usePromptRunController.hook.ts* [L250-L360]
patching file: *usePromptRunController.liveSessionObserver.test.tsx*

AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx
Copy
Review
Accept
PatchFull file

```
-
- import { describe, it } from '@jest/globals';
-
+
+ import { afterEach, beforeEach, describe, it, jest } from '@jest/globals';
+ import React from 'react';
+ import { act } from 'react';
+ import { createRoot } from 'react-dom/client';
+
+
```

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

```
- import type { LiveSessionRunRef } from
'../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
-
+ import type { LiveSessionRunRef } from
'../../../../sharedlib/types/liveSessionTypes/LiveSessionRun.type.js';
+
+ const RUN_REF: LiveSessionRunRef = {
+     locator: {
+         sessionId: 'a'.repeat(32),
+         storageScopeRelPath: '',
+     },
+     turnId: 'turn',
+     sourceId: 'source',
+ };
+ const UI_SETTINGS = {
+     autoScroll: false,
+     paginationEnabled: false,
+ } as const;
+ const startMock = jest.fn<() => AiStreamStartResult>();
+ const resetMock = jest.fn<() => void>();
+ const stopMock = jest.fn<() => Promise<void>>(async () => {});
+
+ const noticeStub = {
+     ingestToolCalls: (): void => {},
+     ensureGapAfterNotice: (): void => {},
+     finalizeActiveNoticeOnFirstMessageToken: (): void => {},
```

```
+     finalizeActiveNoticeOnFirstThinkingToken: (): void => {},
+     finalizeActiveNoticeAtTurnEnd: (): void => {},
+     reset: (): void => {},
+     summaryOpen: (): void => {},
+     summaryAppend: (): void => {},
+     summaryClose: (): void => {},
+     webSearchOpenIfNeeded: (): void => {},
+     webSearchAddItem: (): void => {},
+     webSearchSetResults: (): void => {},
+     webSearchMarkError: (): void => {},
+     markWriteOrPatchErrorByPath: (): void => {},
+     ciOpenItem: (): void => {},
+     ciAppendCode: (): void => {},
+     ciAppendOutputs: (): void => {},
+     ciAppendFile: (): void => {},
+     ciFinalize: (): void => {},
+     applyToolCallsUpdatePayload: (): void => {},
+ };
+
+ jest.unstable_mockModule('../../../../messageBus/useAiStream.hook.js', () => ({
+     useAiStream: () => ({
+         thinkingTokens: [],
+         tokens: [],
+         isStreaming: false,
+         rawLiveLogs: [],
+         start: startMock,
+         lastMetaChunk: undefined,
+         lastToolCalls: undefined,
+         lastUsageChunk: undefined,
+         fallbackPersistedAck: undefined,
+         activeRunRef: undefined,
+         streamInstanceId: undefined,
+         outcome: undefined,
+         terminalDataSeen: false,
+         transportStopAvailable: false,
+         stop: stopMock,
+         reset: resetMock,
+     }),
+ }));
+ jest.unstable_mockModule('../usePromptToolCallNotice.hook.js', () => ({
+     usePromptToolCallNotice: () => noticeStub,
+ }));
+ jest.unstable_mockModule('../usePromptAiQueryOptionsBase.hook.js', () => ({
+     usePromptAiQueryOptionsBase: () => ({
+         reasoning: 'medium',
+         verbosity: 'medium',
+     }),
+ }));
+
+ const { usePromptRunController } =
+     await import('../usePromptRunController.hook.js');
+ type ControllerApi = ReturnType<typeof usePromptRunController>;
+ const controllerApiRef: { current: ControllerApi | null } = { current: null };
+
+ /** Mount the real controller so accepted/refused observer starts exercise the real
scaffold mutation. */
+ function Harness(): React.ReactElement {
+     const controller = usePromptRunController({
+         sessionLocator: RUN_REF.locator,
+         dbSettingsExtUiPrompt: UI_SETTINGS,
+         threadIdOverride: RUN_REF.locator.sessionId,
+     });
+     controllerApiRef.current = controller;
```

```
+     return React.createElement('div');
+ }
+
```

```
- describe('Prompt live observer controller contract', () => {
-     it('requires accepted stream identity before placeholders and exposes finalization
identity', () => {
-         const result: AiStreamStartResult = {
-
+ describe('Prompt live observer controller contract', () => {
+     let container: HTMLDivElement;
+     let root: ReturnType<typeof createRoot>;
+
+     beforeEach(() => {
+         startMock.mockReset();
+         resetMock.mockClear();
+         stopMock.mockClear();
+         controllerApiRef.current = null;
+         container = document.createElement('div');
+         document.body.appendChild(container);
+         root = createRoot(container);
+     });
+
+     afterEach(() => {
+         act(() => root.unmount());
+         container.remove();
+     });
+
+     it('creates no placeholders when the observer transport refuses synchronous
acceptance', async () => {
+         startMock.mockReturnValue({ accepted: false });
+         await act(async () => {
+             root.render(React.createElement(Harness));
+         });
+         const controller = controllerApiRef.current;
+         if (!controller) throw new Error('Controller harness was not initialized.');
+
+         let result: AiStreamStartResult | undefined;
+         act(() => {
+             result = controller.startLiveObserver({
+                 runRef: RUN_REF,
+                 viewerLeaseId: 'lease',
+                 replayMode: 'replaceTurnAssistants',
+                 stopPolicy: 'source',
+             });
+         });
+
+         expect(result).toEqual({ accepted: false });
+         expect(controllerApiRef.current?.messages).toEqual([]);
+     });
+
+     it('commits exactly one canonical assistant scaffold after accepted observer
identity', async () => {
+         const accepted: AiStreamStartResult = {
+
```

```
-             accepted: true,
-             runRef: {
-                 locator: {
-                     sessionId: 'a'.repeat(32),
-                     storageScopeRelPath: '',
-                 },
-                 turnId: 'turn',
-                 sourceId: 'source',
```

```
-             },
-             streamInstanceId: 'stream',
-
+             accepted: true,
+             runRef: RUN_REF,
+             streamInstanceId: 'stream-observer',
+
```

```
-         };
-         expect(result.accepted && result.streamInstanceId).toBe('stream');
-
+         };
+         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.');
+
+         let result: AiStreamStartResult | undefined;
+         act(() => {
+             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',
+         });
+         expect(controllerApiRef.current?.messages).toEqual([
+             expect.objectContaining({
+                 turnId: RUN_REF.turnId,
+                 role: 'assistant',
+                 assistantBubbleKind: 'think',
+             }),
+             expect.objectContaining({
+                 turnId: RUN_REF.turnId,
+                 role: 'assistant',
+                 fileEditsUiFinalized: false,
+             }),
+         ]);
+
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx
Copy
Review
Accept

🔧 ✅ *writing file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Copy
Review
Accept

```
/** @jest-environment jsdom */
```

```
import type {
    ChangeEvent,
    Dispatch,
    JSX,
    MutableRefObject,
    SetStateAction,
} from 'react';
import {
    forwardRef,
    useCallback,
    useEffect,
    useImperativeHandle,
    useMemo,
    useRef,
    useState,
    useSyncExternalStore,
} from 'react';
import { jest } from '@jest/globals';
import { render, type RenderResult } from '@testing-library/react';

import type {
    AiStreamFallbackPersistedAck,
    AiStreamStartResult,
} from '../../../../messageBus/AiStreamStartArgs.type.js';
import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
import type { PromptMessage } from '../../PromptHistory/promptHistoryTypes.type.js';

export const PROMPT_BARRIER_SESSION_ID =
    '0123456789abcdef0123456789abcdef';
export const PROMPT_BARRIER_SEND_TURN_ID = 'send-turn';
export const PROMPT_BARRIER_SEND_STREAM_ID = 'stream-send';
export const PROMPT_BARRIER_EDIT_TURN_ID = 'edit-turn';
export const PROMPT_BARRIER_EDIT_STREAM_ID = 'stream-edit';

/** Initial persisted state injected by the hydration mock before a barrier scenario
begins. */
export interface PromptBarrierHydrationFixture {
    readonly promptDraft: string;
    readonly attachments: readonly PromptAttachment[];
    readonly nextTurnId?: string;
    readonly messages: readonly PromptMessage[];
}

const EMPTY_FIXTURE: PromptBarrierHydrationFixture = {
    promptDraft: '',
    attachments: [],
    messages: [],
};

let hydrationFixture: PromptBarrierHydrationFixture = EMPTY_FIXTURE;
let fallbackPersistedAck: AiStreamFallbackPersistedAck | undefined;
let finalizedStreamInstanceId: string | undefined;
let controllerSignalVersion = 0;
const controllerSignalListeners = new Set<() => void>();

/** Subscribe one mounted mock controller to module-scoped streamed signal changes. */
function subscribeControllerSignals(listener: () => void): () => void {
    controllerSignalListeners.add(listener);
    return () => controllerSignalListeners.delete(listener);
}

/** Return the monotone external-store version consumed by useSyncExternalStore. */
```

```
function getControllerSignalVersion(): number {
    return controllerSignalVersion;
}

/** Publish one ack/finalization change to every mounted Prompt harness. */
function publishControllerSignalChange(): void {
    controllerSignalVersion += 1;
    for (const listener of controllerSignalListeners) listener();
}

export const handleSendMock = jest.fn<() => AiStreamStartResult>();
export const handleSubmitEditMock = jest.fn<() => AiStreamStartResult>();
export const focusAtStartMock = jest.fn<() => void>();

const SEND_ACCEPTED_RESULT: AiStreamStartResult = {
    accepted: true,
    runRef: {
        locator: {
            sessionId: PROMPT_BARRIER_SESSION_ID,
            storageScopeRelPath: '',
        },
        turnId: PROMPT_BARRIER_SEND_TURN_ID,
        sourceId: 'source-send',
    },
    streamInstanceId: PROMPT_BARRIER_SEND_STREAM_ID,
};

const EDIT_ACCEPTED_RESULT: AiStreamStartResult = {
    accepted: true,
    runRef: {
        locator: {
            sessionId: PROMPT_BARRIER_SESSION_ID,
            storageScopeRelPath: '',
        },
        turnId: PROMPT_BARRIER_EDIT_TURN_ID,
        sourceId: 'source-edit',
    },
    streamInstanceId: PROMPT_BARRIER_EDIT_STREAM_ID,
};

const replaceAllMessagesSink = jest.fn<(messages: PromptMessage[]) => void>();
const busCallMock = jest.fn<() => Promise<unknown>>(async () => ({}));
const callLocalApiMock = jest.fn<
    (url: string, args?: unknown) => Promise<unknown>
>(async () => ({}));

/** Real-hook-shaped mock that keeps Prompt messages reactive while transport identity
remains test-controlled. */
function usePromptRunControllerMock() {
    const [messages, setMessages] = useState<PromptMessage[]>([]);
    const controllerSignalVersionSnapshot = useSyncExternalStore(
        subscribeControllerSignals,
        getControllerSignalVersion,
        getControllerSignalVersion,
    );
    const replaceAllMessages = useCallback(
        (nextMessages: PromptMessage[]): void => {
            replaceAllMessagesSink(nextMessages);
            setMessages([...nextMessages]);
        },
        [],
    );
```

```
    const noopStart = useCallback((): AiStreamStartResult => {
        return { accepted: false };
    }, []);
    const noop = useCallback((): void => {}, []);
    const noopAsync = useCallback(async (): Promise<void> => {}, []);

    return useMemo(
        () => {
            // Reading the external-store version makes ack/finalization changes an
explicit memo dependency.
            void controllerSignalVersionSnapshot;
            return {
                messages,
                isStreaming: false,
                uiPending: false,
                forceScrollSeq: 0,
                handleSend: handleSendMock,
                handleSendSpecifAction: noopStart,
                handleContinueResponse: noopStart,
                handleRetryQuery: noopStart,
                handleStopFromComposer: noopAsync,
                handleReplyButton: noop,
                handleThinkingOpenChange: noop,
                isPromptLocked: false,
                isRunningForInput: false,
                onBeforeSubmitUi: noop,
                lastReceivedStreamKind: undefined,
                abortAckPending: false,
                replaceAllMessages,
                startLiveObserver: noopStart,
                resetPromptRunState: noop,
                handleSubmitEdit: handleSubmitEditMock,
                rawLiveLogs: [],
                fallbackPersistedAck,
                activeRunRef: undefined,
                streamInstanceId: undefined,
                finalizedStreamInstanceId,
                streamOutcome: undefined,
                terminalDataSeen: false,
                transportStopAvailable: false,
                lastMetaChunk: undefined,
            };
        },
        [
            messages,
            noop,
            noopAsync,
            noopStart,
            replaceAllMessages,
            controllerSignalVersionSnapshot,
        ],
    );
}

type HydrationParams = {
    enabled: boolean;
    replaceAllMessages: (messages: PromptMessage[]) => void;
    setPromptValue: Dispatch<SetStateAction<string>>;
    setAttachments: Dispatch<SetStateAction<PromptAttachment[]>>;
    setNextTurnId: Dispatch<SetStateAction<string | undefined>>;
    setOpenThreadId: Dispatch<SetStateAction<string | undefined>>;
```

```
    setHydrateScrollSeq: Dispatch<SetStateAction<number>>;
    setIsHydrating: Dispatch<SetStateAction<boolean>>;
    hasLocalPromptOverrideRef: MutableRefObject<boolean>;
};

/** Apply the configured fixture once through the same setters consumed by production
hydration. */
function usePromptSessionHydrationMock(params: HydrationParams): void {
    const appliedRef = useRef(false);
    useEffect(() => {
        if (!params.enabled || appliedRef.current) return;
        appliedRef.current = true;

        // Install one coherent baseline before the test triggers Send/Edit and arms the
fallback barrier.
        params.setIsHydrating(true);
        params.replaceAllMessages([...hydrationFixture.messages]);
        params.setPromptValue(hydrationFixture.promptDraft);
        params.setAttachments([...hydrationFixture.attachments]);
        params.setNextTurnId(hydrationFixture.nextTurnId);
        params.setOpenThreadId(PROMPT_BARRIER_SESSION_ID);
        params.setHydrateScrollSeq((current) => current + 1);
        params.setIsHydrating(false);
    }, [params]);
}

const NO_LIVE_SESSION_RESULT = {
    markSourceStopRequested: (): boolean => false,
    retryHandoff: (): void => {},
    sourceStopAvailable: false,
    sourceStopPending: false,
    stopSource: async (): Promise<void> => {},
};

jest.unstable_mockModule('../../../../lib/useInitPage.hook.js', () => ({
    useInitPage: (): void => {},
}));
jest.unstable_mockModule('../../../../lib/useAiCatalog.hook.js', () => ({
    useAiCatalog: () => undefined,
}));
jest.unstable_mockModule('../../../../messageBus/frontMessageBus.util.js', () => ({
    bus: {
        isReady: (): boolean => true,
        on: (): void => {},
        off: (): void => {},
        call: busCallMock,
    },
}));
jest.unstable_mockModule('../../../../messageBus/callLocalApi.util.js', () => ({
    callLocalApi: callLocalApiMock,
}));
jest.unstable_mockModule('../../../../messageBus/closeSelf.util.js', () => ({
    closeSelf: async (): Promise<void> => {},
}));
jest.unstable_mockModule(
    '../../../../vsc-design-system/popups/VscAlertDialog.js',
    () => ({ showVscAlertDialog: async (): Promise<void> => {} }),
);
jest.unstable_mockModule(
    '../../../../vsc-design-system/layouts/VscVertStackLayout.js',
    () => ({
        VscVertStackLayout: (props: {
```

```
            children?: JSX.Element | JSX.Element[];
        }): JSX.Element => <div>{props.children}</div>,
        VscVertStackRow: (props: {
            children?: JSX.Element | JSX.Element[];
        }): JSX.Element => <div>{props.children}</div>,
    }),
);
jest.unstable_mockModule('../PromptMainViewport.js', () => ({
    PromptMainViewport: (props: {
        chatView: JSX.Element;
        logsView: JSX.Element;
    }): JSX.Element => (
        <div>
            {props.chatView}
            {props.logsView}
        </div>
    ),
}));
jest.unstable_mockModule(
    '../../PromptNetworkLogs/PromptNetworkLogsView.js',
    () => ({
        PromptNetworkLogsView: (): JSX.Element => <div data-testid="logs" />,
    }),
);
jest.unstable_mockModule(
    '../../PromptHydrationLine/PromptHydrationLine.js',
    () => ({
        PromptHydrationLine: (): JSX.Element => (
            <div data-testid="hydration" />
        ),
    }),
);
jest.unstable_mockModule('../../PromptHistory/PromptHistory.js', () => ({
    PromptHistory: (props: {
        messages: readonly PromptMessage[];
        onRequestEditLastPrompt: (args: {
            turnId: string;
            text: string;
        }) => void;
    }): JSX.Element => {
        const handleEdit = useCallback((): void => {
            props.onRequestEditLastPrompt({
                turnId: PROMPT_BARRIER_EDIT_TURN_ID,
                text: 'edited prompt',
            });
        }, [props.onRequestEditLastPrompt]);
        return (
            <div>
                <div data-testid="historyMessageCount">
                    {props.messages.length}
                </div>
                <button data-testid="editLastPrompt" onClick={handleEdit}>
                    Edit
                </button>
            </div>
        );
    },
}));
jest.unstable_mockModule('../../PromptInput/PromptInput.js', () => ({
    PromptInput: forwardRef(function PromptInputMock(
        props: {
```

```
        promptValue: string;
        submitTxnPending?: boolean;
        attachments: readonly PromptAttachment[];
        attachmentsTurnId?: string;
        isEditLastPromptActive?: boolean;
        composerMaximized: boolean;
        onChangePromptValue: (value: string) => void;
        onChangeComposerMaximized: (value: boolean) => void;
        onSend: (text: string) => void;
    },
    ref,
): JSX.Element {
    const textareaRef = useRef<HTMLTextAreaElement | null>(null);
    const focusAtStart = useCallback((): void => {
        focusAtStartMock();
        textareaRef.current?.focus();
        textareaRef.current?.setSelectionRange(0, 0);
    }, []);
    const isEventFromComposer = useCallback((): boolean => true, []);
    const submitFromGlobalHotkey = useCallback((): void => {}, []);
    const imperativeHandle = useMemo(
        () => ({
            isEventFromComposer,
            submitFromGlobalHotkey,
            focusAtStart,
        }),
        [focusAtStart, isEventFromComposer, submitFromGlobalHotkey],
    );
    useImperativeHandle(ref, () => imperativeHandle, [imperativeHandle]);
    const handleChange = useCallback(
        (event: ChangeEvent<HTMLTextAreaElement>): void => {
            props.onChangePromptValue(event.target.value);
        },
        [props.onChangePromptValue],
    );
    const handleSend = useCallback((): void => {
        props.onSend(props.promptValue);
    }, [props.onSend, props.promptValue]);
    const handleMaximize = useCallback((): void => {
        props.onChangeComposerMaximized(true);
    }, [props.onChangeComposerMaximized]);

    return (
        <div>
            <textarea
                ref={textareaRef}
                data-testid="promptInput"
                disabled={props.submitTxnPending === true}
                value={
                    props.submitTxnPending === true
                        ? ''
                        : props.promptValue
                }
                onChange={handleChange}
            />
            <div data-testid="attachmentCount">
                {props.attachments.length}
            </div>
            <div data-testid="attachmentsTurnId">
                {props.attachmentsTurnId ?? ''}
            </div>
```

```
                    <div data-testid="editMode">
                        {props.isEditLastPromptActive ? 'edit' : 'normal'}
                    </div>
                    <div data-testid="maximized">
                        {props.composerMaximized ? 'yes' : 'no'}
                    </div>
                    <button
                        data-testid="sendPrompt"
                        onClick={handleSend}
                    >
                        Send
                    </button>
                    <button
                        data-testid="maximizePrompt"
                        onClick={handleMaximize}
                    >
                        Maximize
                    </button>
                </div>
            );
        }),
}));
jest.unstable_mockModule('../usePromptDbSettingsExt.hook.js', () => ({
    usePromptDbSettingsExt: () => ({
        dbSettingsExtInit: {
            limits: {
                attachedFiles: {
                    maxFileSize: { type: 'noLimit' },
                },
            },
            featureFlags: [],
        },
        dbSettingsExtUiPrompt: {
            autoScroll: true,
            paginationEnabled: false,
            queryDoneSoundEnabled: false,
        },
        handleChangeDbSettingsExtUiPrompt: (): void => {},
    }),
}));
jest.unstable_mockModule('../usePromptIndexPending.hook.js', () => ({
    usePromptIndexPending: () => ({
        gateAction: (action: () => void | Promise<void>): void => {
            void action();
        },
        isIndexPending: false,
        forceNow: (): void => {},
        closeNotice: (): void => {},
    }),
}));
jest.unstable_mockModule('../usePromptCancelController.hook.js', () => ({
    usePromptCancelController: () => ({
        handleStopFromComposerRequested: async (): Promise<void> => {},
    }),
}));
jest.unstable_mockModule('../usePromptGlobalHotkeys.hook.js', () => ({
    usePromptGlobalHotkeys: (): void => {},
}));
jest.unstable_mockModule('../usePromptViewMode.hook.js', () => ({
    usePromptViewMode: () => ({
        viewSelection: ['chatView'],
```

```
        viewportSelection: ['chatView'],
        setViewSelection: (): void => {},
        isLogsOnly: false,
    }),
}));
jest.unstable_mockModule('../usePromptQueryDoneSoundSync.hook.js', () => ({
    usePromptQueryDoneSoundSync: (): void => {},
}));
jest.unstable_mockModule('../usePromptAiQueryOptionsBase.hook.js', () => ({
    usePromptAiQueryOptionsBase: () => ({
        reasoning: 'medium',
        verbosity: 'medium',
    }),
}));
jest.unstable_mockModule('../usePromptRunController.hook.js', () => ({
    usePromptRunController: usePromptRunControllerMock,
}));
jest.unstable_mockModule('../usePromptSessionHydration.hook.js', () => ({
    usePromptSessionHydration: usePromptSessionHydrationMock,
}));
jest.unstable_mockModule(
    '../../liveSession/core/usePromptLiveSession.hook.js',
    () => ({ usePromptLiveSession: () => NO_LIVE_SESSION_RESULT }),
);
jest.unstable_mockModule(
    '../../../../lib/requestAnimationFrameTimeout.util.js',
    () => ({
        requestAnimationFrameTimeout: (callback: () => void): void => {
            callback();
        },
    }),
);

const { Prompt } = await import('../Prompt.js');

/** Reset every mutable fixture and install one strict classic Prompt locator. */
export function resetPromptFallbackPersistBarrierHarness(
    fixture: PromptBarrierHydrationFixture = EMPTY_FIXTURE,
): void {
    hydrationFixture = fixture;
    fallbackPersistedAck = undefined;
    finalizedStreamInstanceId = undefined;
    controllerSignalVersion += 1;
    replaceAllMessagesSink.mockClear();
    handleSendMock.mockReset();
    handleSendMock.mockReturnValue(SEND_ACCEPTED_RESULT);
    handleSubmitEditMock.mockReset();
    handleSubmitEditMock.mockReturnValue(EDIT_ACCEPTED_RESULT);
    focusAtStartMock.mockClear();
    busCallMock.mockClear();
    callLocalApiMock.mockClear();
    Object.defineProperty(globalThis, '__AICODE_ROUTER_QUERY_STRING__', {
        configurable: true,
        value: `sessionId=${PROMPT_BARRIER_SESSION_ID}&storageScopeRelPath=`,
    });
}

/** Remove the injected router query after one test file finishes. */
export function cleanupPromptFallbackPersistBarrierHarness(): void {
    Reflect.deleteProperty(globalThis, '__AICODE_ROUTER_QUERY_STRING__');
}
```

```
/** Publish one fallback-persisted acknowledgement on the next Prompt rerender. */
export function setPromptFallbackPersistedAck(
    ack: AiStreamFallbackPersistedAck | undefined,
): void {
    fallbackPersistedAck = ack;
    publishControllerSignalChange();
}

/** Publish one exact frontend finalization identity on the next Prompt rerender. */
export function setPromptFinalizedStreamInstanceId(
    streamInstanceId: string | undefined,
): void {
    finalizedStreamInstanceId = streamInstanceId;
    publishControllerSignalChange();
}

/** Mount the real Prompt component against the shared barrier harness. */
export function renderPromptFallbackPersistBarrierHarness(): RenderResult {
    return render(<Prompt />);
}

/** Rerender the same Prompt root after a module-scoped ack/finalization signal changes.
*/
export function rerenderPromptFallbackPersistBarrierHarness(
    rendered: RenderResult,
): void {
    rendered.rerender(<Prompt />);
}
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Copy
Review
Accept

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx* [L1-L35], ❌ 🔎 e *"Cast import Prompt harness"*, *PromptFallbackPersistBarrier.testHarness.tsx* [L375-L420]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *promptHistoryTypes.type.ts* [L1-L90]
🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Copy
Review
Accept
PatchFull file

```
-
- import { describe, it } from '@jest/globals';
- import type { AiStreamFallbackPersistedAck } from
'../../../../messageBus/AiStreamStartArgs.type.js';
-
+
+ import { afterEach, beforeEach, describe, it } from '@jest/globals';
+ import { act, fireEvent, screen, waitFor } from '@testing-library/react';
+
+ import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
```

```
+ import {
+     cleanupPromptFallbackPersistBarrierHarness,
+     focusAtStartMock,
+     PROMPT_BARRIER_SEND_STREAM_ID,
+     PROMPT_BARRIER_SEND_TURN_ID,
+     renderPromptFallbackPersistBarrierHarness,
+     rerenderPromptFallbackPersistBarrierHarness,
+     resetPromptFallbackPersistBarrierHarness,
+     setPromptFallbackPersistedAck,
+ } from './PromptFallbackPersistBarrier.testHarness.js';
+
+ const PROBLEMS_ATTACHMENT: PromptAttachment = {
+     kind: 'problems',
+     problems: {
+         id: 'a'.repeat(64),
+         count: 2,
+     },
+ };
+
-------------------------------------------------------------------------------
- describe('Prompt fallback ack attachment barrier', () => {
-     it('matches both turnId and streamInstanceId before clearing composer state', () =>
{
-         const pending = { turnId: 'turn', streamInstanceId: 'stream-a' };
-         const stale: AiStreamFallbackPersistedAck = {
-             turnId: 'turn',
-             streamInstanceId: 'stream-b',
-         };
-         const exact: AiStreamFallbackPersistedAck = { ...pending };
-         expect(stale).not.toEqual(pending);
-         expect(exact).toEqual(pending);
-
+ describe('Prompt fallback ack attachment barrier', () => {
+     beforeEach(() => {
+         resetPromptFallbackPersistBarrierHarness({
+             promptDraft: 'keep this prompt until durable ack',
+             attachments: [PROBLEMS_ATTACHMENT],
+             nextTurnId: 'next-turn',
+             messages: [],
+         });
+     });
+
+     afterEach(() => {
+         cleanupPromptFallbackPersistBarrierHarness();
+     });
+
+     it('clears attachments and next-turn state only for the exact turn plus stream
acknowledgement', async () => {
+         const rendered = renderPromptFallbackPersistBarrierHarness();
+         await waitFor(() => {
+             expect(screen.getByTestId('attachmentCount')).toHaveTextContent(
+                 '1',
+             );
+         });
+
+         fireEvent.click(screen.getByTestId('maximizePrompt'));
+         fireEvent.click(screen.getByTestId('sendPrompt'));
+         expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+         expect(screen.getByTestId('promptInput')).toBeDisabled();
+
+         // A stale acknowledgement for the same logical turn but another accepted
stream must be inert.
+         await act(async () => {
+             setPromptFallbackPersistedAck({
```

```
+                 turnId: PROMPT_BARRIER_SEND_TURN_ID,
+                 streamInstanceId: 'stale-stream',
+             });
+             rerenderPromptFallbackPersistBarrierHarness(rendered);
+         });
+         expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+         expect(screen.getByTestId('attachmentsTurnId')).toHaveTextContent(
+             'next-turn',
+         );
+
+         // The stream identity alone is also insufficient when the persisted
acknowledgement belongs to another turn.
+         await act(async () => {
+             setPromptFallbackPersistedAck({
+                 turnId: 'stale-turn',
+                 streamInstanceId: PROMPT_BARRIER_SEND_STREAM_ID,
+             });
+             rerenderPromptFallbackPersistBarrierHarness(rendered);
+         });
+         expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+
+         await act(async () => {
+             setPromptFallbackPersistedAck({
+                 turnId: PROMPT_BARRIER_SEND_TURN_ID,
+                 streamInstanceId: PROMPT_BARRIER_SEND_STREAM_ID,
+             });
+             rerenderPromptFallbackPersistBarrierHarness(rendered);
+         });
+
+         await waitFor(() => {
+             expect(screen.getByTestId('attachmentCount')).toHaveTextContent(
+                 '0',
+             );
+         });
+         expect(screen.getByTestId('attachmentsTurnId').textContent).toBe('');
+         expect(screen.getByTestId('maximized')).toHaveTextContent('no');
+         expect(screen.getByTestId('promptInput')).not.toBeDisabled();
+         expect(focusAtStartMock).toHaveBeenCalledTimes(1);
+
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Copy
Review
Accept

🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearInput.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
Copy
Review
Accept
PatchFull file

```
-
- import { describe, it } from '@jest/globals';
-
+
+ import { afterEach, describe, it } from '@jest/globals';
+ import { act, fireEvent, screen, waitFor } from '@testing-library/react';
+
+ import type { PromptAttachment } from '../../lib/PromptAttachment.type.js';
+ import type { PromptMessage } from '../../PromptHistory/promptHistoryTypes.type.js';
+ import {
+     cleanupPromptFallbackPersistBarrierHarness,
```

```
+     focusAtStartMock,
+     PROMPT_BARRIER_EDIT_STREAM_ID,
+     PROMPT_BARRIER_EDIT_TURN_ID,
+     PROMPT_BARRIER_SEND_STREAM_ID,
+     renderPromptFallbackPersistBarrierHarness,
+     rerenderPromptFallbackPersistBarrierHarness,
+     resetPromptFallbackPersistBarrierHarness,
+     setPromptFinalizedStreamInstanceId,
+ } from './PromptFallbackPersistBarrier.testHarness.js';
+
+ const PROBLEMS_ATTACHMENT: PromptAttachment = {
+     kind: 'problems',
+     problems: {
+         id: 'b'.repeat(64),
+         count: 1,
+     },
+ };
+
-------------------------------------------------------------------------------
- describe('Prompt accepted-finalization fallback barrier', () => {
-     it('unmasks without clearing local composer fields when no matching ack arrived',
() => {
-         const composer = {
-             prompt: 'keep me',
-             attachments: ['doc'],
-             nextTurnId: 'next',
-             pendingEditTurnId: 'edit',
-             maximized: true,
-             masked: true,
-
+ describe('Prompt accepted-finalization fallback barrier', () => {
+     afterEach(() => {
+         cleanupPromptFallbackPersistBarrierHarness();
+     });
+
+     it('unmasks and restores focus without clearing prompt, attachments, nextTurnId, or
maximized state', async () => {
+         resetPromptFallbackPersistBarrierHarness({
+             promptDraft: 'keep me after finalization',
+             attachments: [PROBLEMS_ATTACHMENT],
+             nextTurnId: 'next-turn',
+             messages: [],
+         });
+         const rendered = renderPromptFallbackPersistBarrierHarness();
+         await waitFor(() => {
+             expect(screen.getByTestId('promptInput')).toHaveValue(
+                 'keep me after finalization',
+             );
+         });
+
+         fireEvent.click(screen.getByTestId('maximizePrompt'));
+         fireEvent.click(screen.getByTestId('sendPrompt'));
+         expect(screen.getByTestId('promptInput')).toBeDisabled();
+         expect(screen.getByTestId('promptInput')).toHaveValue('');
+         focusAtStartMock.mockClear();
+
+         await act(async () => {
+             setPromptFinalizedStreamInstanceId(
+                 PROMPT_BARRIER_SEND_STREAM_ID,
+             );
+             rerenderPromptFallbackPersistBarrierHarness(rendered);
+         });
+
+         await waitFor(() => {
```

```
+            expect(screen.getByTestId('promptInput')).not.toBeDisabled();
+        });
+        expect(screen.getByTestId('promptInput')).toHaveValue(
+            'keep me after finalization',
+        );
+        expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+        expect(screen.getByTestId('attachmentsTurnId')).toHaveTextContent(
+            'next-turn',
+        );
+        expect(screen.getByTestId('maximized')).toHaveTextContent('yes');
+        expect(focusAtStartMock).toHaveBeenCalledTimes(1);
+    });
+
+    it('preserves pending edit identity and restored message attachments without
fallback ack', async () => {
+        const editableUserMessage: PromptMessage = {
+            turnId: PROMPT_BARRIER_EDIT_TURN_ID,
+            uid: 'user-edit',
+            role: 'user',
+            text: 'original prompt',
+            attachments: [PROBLEMS_ATTACHMENT],
+
──────────────────────────────────────────────────────────────────────────────────
-        };
-        const finalizedStreamInstanceId = 'stream';
-        const pending = { turnId: 'turn', streamInstanceId: 'stream' };
-        if (finalizedStreamInstanceId === pending.streamInstanceId) {
-            composer.masked = false;
-        }
-        expect(composer).toEqual({
-            prompt: 'keep me',
-            attachments: ['doc'],
-            nextTurnId: 'next',
-            pendingEditTurnId: 'edit',
-            maximized: true,
-            masked: false,
-
+        };
+        resetPromptFallbackPersistBarrierHarness({
+            promptDraft: '',
+            attachments: [],
+            nextTurnId: 'next-turn',
+            messages: [editableUserMessage],
+        });
+        const rendered = renderPromptFallbackPersistBarrierHarness();
+        await waitFor(() => {
+            expect(screen.getByTestId('historyMessageCount')).toHaveTextContent(
+                '1',
+            );
+        });
+
+        fireEvent.click(screen.getByTestId('editLastPrompt'));
+        await waitFor(() => {
+            expect(screen.getByTestId('editMode')).toHaveTextContent('edit');
+        });
+        expect(screen.getByTestId('promptInput')).toHaveValue('edited prompt');
+        expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+        expect(screen.getByTestId('attachmentsTurnId')).toHaveTextContent(
+            PROMPT_BARRIER_EDIT_TURN_ID,
+        );
+
+        focusAtStartMock.mockClear();
+        fireEvent.click(screen.getByTestId('sendPrompt'));
+        await act(async () => {
```

```
+             setPromptFinalizedStreamInstanceId(
+                 PROMPT_BARRIER_EDIT_STREAM_ID,
+             );
+             rerenderPromptFallbackPersistBarrierHarness(rendered);
+         });
+
+         await waitFor(() => {
+             expect(screen.getByTestId('promptInput')).not.toBeDisabled();
+
```

```
-         });
-     });
-
+         });
+         expect(screen.getByTestId('promptInput')).toHaveValue('edited prompt');
+         expect(screen.getByTestId('attachmentCount')).toHaveTextContent('1');
+         expect(screen.getByTestId('editMode')).toHaveTextContent('edit');
+         expect(screen.getByTestId('attachmentsTurnId')).toHaveTextContent(
+             PROMPT_BARRIER_EDIT_TURN_ID,
+         );
+         expect(focusAtStartMock).toHaveBeenCalledTimes(1);
+     });
+
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
Copy
Review
Accept

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx* [L430-L455], *Prompt.fallbackPersistAck.clearAttachments.test.tsx* [L1-L150], *Prompt.fallbackPersistAck.clearInput.test.tsx* [L1-L220]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ❌ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Error
(what is it?)

🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearInput.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx* [L1-L100]
🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx* [L90-L155]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*

AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearInput.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading*
project: *preparePromptContinuationStreaming.util.ts*, *thinkingSections.hydration.test.ts*, 🔎 e *"Usage assistant hydratée"*, *FrontChatSession.type.ts* [L1-L220], *PromptFallbackPersistBarrier.testHarness.tsx* [L45-L150], *PromptFallbackPersistBarrier.testHarness.tsx* [L150-L185], *frontSessionHydration.meta.util.ts* [L1-L180], 🔎 e *"Snapshot usage hydraté"*, *frontSessionToPromptMessages.util.ts* [L860-L940]
🔧 ✅ *patching file*: *usePromptLiveSession.normalPromptIsomorphism.test.tsx*
AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Copy
Review
Accept
PatchFull file

```
- import { describe, it } from '@jest/globals';
-
-
+ import { describe, it } from '@jest/globals';
+
+ import { AiStopReason } from
'../../../../../sharedlib/types/aiTypes/AiStopReason.type.js';
+
```

```
- import { AiStopErrorReason } from
'../../../../../sharedlib/types/aiTypes/AiStopErrorReason.type.js';
-
+ import { AiStopErrorReason } from
'../../../../../sharedlib/types/aiTypes/AiStopErrorReason.type.js';
+ import type { AiStreamedChunk } from
'../../../../../sharedlib/types/aiTypes/AiStreamedChunk.type.js';
+
```

```
- import { ZLiveSessionOpenResult } from
'../../../../../sharedlib/types/liveSessionTypes/LiveSessionOpenResult.type.js';
-
+ import { ZLiveSessionOpenResult } from
'../../../../../sharedlib/types/liveSessionTypes/LiveSessionOpenResult.type.js';
+ import type { FrontChatSession } from '../../../hydration/FrontChatSession.type.js';
+
```

```
- import { applyHydratedPromptSessionState } from
'../../../Prompt/applyHydratedPromptSessionState.util.js';
-
+ import { applyHydratedPromptSessionState } from
'../../../Prompt/applyHydratedPromptSessionState.util.js';
+ import {
+     buildPromptStreamingAssistantMessages,
+     createPromptStreamingTurnPlan,
+ } from '../../../Prompt/preparePromptStreamingTurn.util.js';
+ import { preparePromptContinuationStreaming } from
'../../../Prompt/preparePromptContinuationStreaming.util.js';
+
```

```
- import { preparePromptContinuationStreaming } from
'../../../Prompt/preparePromptContinuationStreaming.util.js';
-
+ import { preparePromptContinuationStreaming } from
'../../../Prompt/preparePromptContinuationStreaming.util.js';
+
+ type ComparablePromptMessage = Pick<
+     PromptMessage,
+     | 'turnId'
+     | 'role'
+     | 'text'
+     | 'assistantBubbleKind'
+     | 'lastMetaChunk'
+     | 'usageSnapshot'
+     | 'fileEditsUiFinalized'
+ >;
+
+ const SESSION_ID = 'a'.repeat(32);
+ const TURN_ID = 'turn';
+ const END_TURN_META: AiStreamedChunk = {
+     turnId: TURN_ID,
+     type: 'meta',
+     value: {
+         metaType: 'stop_reason',
+         value: { reason: AiStopReason.END_TURN },
+     },
+ };
+ type UsageSnapshot = Extract<
+     AiStreamedChunk,
+     { type: 'meta'; value: { metaType: 'usage' } }
+ >['value']['value'];
+ const USAGE_SNAPSHOT: UsageSnapshot = {
+     input_tokens: 10,
+     output_tokens: 20,
+     total_tokens: 30,
+     query_time: {
+         start_ts_utc: 100,
+         end_ts_utc: 175,
+         duration_ms: 75,
+     },
+ };
+
+ /** Remove generated UIDs/callbacks so only user-visible and persisted streaming state
is compared. */
+ function toComparableMessages(
+     messages: readonly PromptMessage[],
+ ): ComparablePromptMessage[] {
+     return messages.map((message) => ({
+         turnId: message.turnId,
+         role: message.role,
+         text: message.text,
```

```
+         assistantBubbleKind:
+             message.role === 'assistant' &&
+             message.assistantBubbleKind !== 'think'
+                 ? 'msg'
+                 : message.assistantBubbleKind,
+         lastMetaChunk: message.lastMetaChunk,
+         usageSnapshot: message.usageSnapshot,
+         fileEditsUiFinalized: message.fileEditsUiFinalized,
+     }));
+ }
+
+ /** Run the real live-session Open schema and hydration pipeline for one replay mode.
*/
+ function hydrateLiveSession(
+     session: FrontChatSession,
+     replayMode: 'replaceTurnAssistants' | 'continueAssistant',
+ ): PromptMessage[] {
+     const result = ZLiveSessionOpenResult.parse({
+         kind: 'live',
+         session,
+         threadId: SESSION_ID,
+         turnId: TURN_ID,
+         sourceId: 'source',
+         replayMode,
+     });
+     if (result.kind !== 'live') {
+         throw new Error('Expected a live replay result.');
+     }
+
+     let hydratedMessages: PromptMessage[] = [];
+     applyHydratedPromptSessionState({
+         mode: 'hydrateFromSnapshot',
+         rawSession: result.session,
+         rawThreadId: result.threadId,
+         options: { allowOrphanToolResults: true },
+         scrollStrategy: 'classicHydration',
+         replaceAllMessages: (messages) => {
+             hydratedMessages = messages;
+         },
+         setPromptValue: () => {},
+         setAttachments: () => {},
+         setNextTurnId: () => {},
+         setOpenThreadId: () => {},
+         setHydrateScrollSeq: () => {},
+         hasLocalPromptOverrideRef: { current: false },
+         resetPromptRunState: () => {},
+     });
+     return hydratedMessages;
+ }
+
+ /** Build the shared persisted session envelope used by all isomorphism scenarios. */
+ function buildSession(messages: FrontChatSession['messages']): FrontChatSession {
+     return {
+         meta: {
+             formatVersionNumber: 1,
+             sessionId: SESSION_ID,
+             title: 'Chat',
+             createdAt: 1,
+             updatedAt: 2,
+             firstSendState: 'postSend',
+         },
+         messages,
+     };
```

```
+ }
+
```

```
- describe('normal Prompt live-session isomorphism contracts', () => {
-     it.each(['replaceTurnAssistants', 'continueAssistant'] as const)(
-         'hydrates the same canonical terminal assistant text produced by streaming for
%s replay',
-         (replayMode) => {
-             const streamedAssistantText = `Answer${buildStopSuffix(
-                 AiStopErrorReason.SYSTEM_ERROR,
-             )}`;
-             const result = ZLiveSessionOpenResult.parse({
-                 kind: 'live',
-                 session: {
-                     meta: {
-                         formatVersionNumber: 1,
-                         sessionId: 'a'.repeat(32),
-                         title: 'Chat',
-                         createdAt: 1,
-                         updatedAt: 2,
-                         firstSendState: 'postSend',
-
+ describe('normal Prompt live-session isomorphism contracts', () => {
+     it('rebuilds replace-style thinking, text, usage, and terminal state exactly like
the streaming scaffold', () => {
+         const streamedPlan = createPromptStreamingTurnPlan({
+             turnId: TURN_ID,
+             buildMessages: (targets) => [
+                 {
+                     turnId: TURN_ID,
+                     uid: 'user',
+                     role: 'user',
+                     text: 'Ask',
+                 },
+                 ...buildPromptStreamingAssistantMessages(targets),
+             ],
+         });
+         const streamedMessages = streamedPlan.messages.map(
+             (message): PromptMessage => {
+                 if (message.assistantBubbleKind === 'think') {
+                     return { ...message, text: 'Plan' };
+                 }
+                 if (message.role === 'assistant') {
+                     return {
+                         ...message,
+                         text: 'Answer',
+                         lastMetaChunk: END_TURN_META,
+                         usageSnapshot: USAGE_SNAPSHOT,
+                         fileEditsUiFinalized: true,
+                     };
+                 }
+                 return message;
+             },
+         );
+         const hydratedMessages = hydrateLiveSession(
+             buildSession([
+                 {
+                     role: 'user',
+                     turnId: TURN_ID,
+                     content: [{ text: 'Ask' }],
+                 },
+                 {
+                     role: 'assistant',
+                     turnId: TURN_ID,
```

```
+                     content: [
+                         { thinking: 'Plan', type: 'frontend_only' },
+                         { text: 'Answer' },
+                     ],
+                     stopReason: AiStopReason.END_TURN,
+                     usage: {
+                         inputTokens: 10,
+                         outputTokens: 20,
+                         totalTokens: 30,
+                         queryTime: {
+                             startTsUtc: 100,
+                             endTsUtc: 175,
+                             durationMs: 75,
+                         },
+
--------------------------------------------------------------------------------
-                     },
-                     messages: [
-                         {
-                             role: 'assistant',
-                             turnId: 'turn',
-                             content: [{ text: streamedAssistantText }],
-                         },
-                     ],
-
+                     },
+
--------------------------------------------------------------------------------
-                 },
-                 threadId: 'a'.repeat(32),
-                 turnId: 'turn',
-                 sourceId: 'source',
-                 replayMode,
-             });
-             if (result.kind !== 'live') {
-                 throw new Error('Expected a live replay result.');
-             }
-             const hydratedMessages: PromptMessage[][] = [];
-
+                 },
+             ]),
+             'replaceTurnAssistants',
+         );
+
+         expect(toComparableMessages(hydratedMessages)).toEqual(
+             toComparableMessages(streamedMessages),
+         );
+     });
+
--------------------------------------------------------------------------------
-
-             applyHydratedPromptSessionState({
-                 mode: 'hydrateFromSnapshot',
-                 rawSession: result.session,
-                 rawThreadId: result.threadId,
-                 options: { allowOrphanToolResults: true },
-                 scrollStrategy: 'classicHydration',
-                 replaceAllMessages: (messages) => {
-                     hydratedMessages.push(messages);
-
+
+     it('rebuilds continue-style base text and one fresh thinking attempt without
duplicate assistant bubbles', () => {
+         const initialMessages: PromptMessage[] = [
+             {
```

```
+                turnId: TURN_ID,
+                uid: 'user',
+                role: 'user',
+                text: 'Ask',
+            },
+            {
+                turnId: TURN_ID,
+                uid: 'assistant',
+                role: 'assistant',
+                text: 'Part one',
+                assistantBubbleKind: 'msg',
+                fileEditsUiFinalized: true,
+            },
+        ];
+        const continuation = preparePromptContinuationStreaming(initialMessages);
+        const streamedMessages = continuation.messages.map(
+            (message): PromptMessage => {
+                if (message.uid === continuation.thinkingUid) {
+                    return { ...message, text: 'Second plan' };
+                }
+                if (message.uid === continuation.assistantUid) {
+                    return {
+                        ...message,
+                        text: `${continuation.baseAssistantText} and part two`,
+                        lastMetaChunk: END_TURN_META,
+                        fileEditsUiFinalized: true,
+                    };
+                }
+                return message;
+            },
+        );
+        const hydratedMessages = hydrateLiveSession(
+            buildSession([
+                {
+                    role: 'user',
+                    turnId: TURN_ID,
+                    content: [{ text: 'Ask' }],
+                },
+                {
+                    role: 'assistant',
+                    turnId: TURN_ID,
+                    content: [
+                        { thinking: 'Second plan', type: 'frontend_only' },
+                        { text: 'Part one and part two' },
+                    ],
+                    stopReason: AiStopReason.END_TURN,
+
--------------------------------------------------------------------------------
-                },
-                setPromptValue: () => {},
-                setAttachments: () => {},
-                setNextTurnId: () => {},
-                setOpenThreadId: () => {},
-                setHydrateScrollSeq: () => {},
-                hasLocalPromptOverrideRef: { current: false },
-                resetPromptRunState: () => {},
-            });
-
+                },
+            ]),
+            'continueAssistant',
+        );
+
--------------------------------------------------------------------------------
```

```
-
-             const assistant = hydratedMessages[0]?.find(
-
+
+         expect(toComparableMessages(hydratedMessages)).toEqual(
+             toComparableMessages(streamedMessages),
+         );
+         expect(
+             hydratedMessages.filter(
+
--------------------------------------------------------------------------------
-                     message.role === 'assistant' &&
-                     message.turnId === 'turn' &&
-
+                     message.role === 'assistant' &&
+
--------------------------------------------------------------------------------
-                     message.assistantBubbleKind === 'msg',
-             );
-             expect(assistant?.text).toBe(streamedAssistantText);
-             expect(result).toEqual(
-                 expect.objectContaining({ sourceId: 'source', replayMode }),
-             );
-         },
-     );
-
+                     message.assistantBubbleKind === 'msg',
+             ),
+         ).toHaveLength(1);
+     });
+
+     it('keeps one canonical error terminal and suffix when transport fails after
terminal data', () => {
+         const expectedText = `Partial answer${buildStopSuffix(
+             AiStopErrorReason.SYSTEM_ERROR,
+         )}`;
+         const hydratedMessages = hydrateLiveSession(
+             buildSession([
+                 {
+                     role: 'user',
+                     turnId: TURN_ID,
+                     content: [{ text: 'Ask' }],
+                 },
+                 {
+                     role: 'assistant',
+                     turnId: TURN_ID,
+                     // The durable memory patch owns the canonical suffix before live
replay becomes handoffable.
+                     content: [{ text: expectedText }],
+                     errorReason: AiStopErrorReason.SYSTEM_ERROR,
+                     errorMessage: 'system error',
+                     errorDetails: 'source stack',
+                 },
+             ]),
+             'replaceTurnAssistants',
+         );
+         const assistantMessages = hydratedMessages.filter(
+             (message) =>
+                 message.role === 'assistant' &&
+                 message.assistantBubbleKind === 'msg',
+         );
+
+         expect(assistantMessages).toHaveLength(1);
+         expect(assistantMessages[0]).toEqual(
```

```
+             expect.objectContaining({
+                 text: expectedText,
+                 lastMetaChunk: {
+                     turnId: TURN_ID,
+                     type: 'meta',
+                     value: {
+                         metaType: 'error',
+                         value: {
+                             reason: AiStopErrorReason.SYSTEM_ERROR,
+                             message: 'system error',
+                             details: 'source stack',
+                         },
+                     },
+                 },
+             }),
+         );
+         expect(
+             assistantMessages[0]?.text.split(
+                 buildStopSuffix(AiStopErrorReason.SYSTEM_ERROR),
+             ),
+         ).toHaveLength(2);
+     });
+
```

AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Copy
Review
Accept

🔧 ✅ *scanning git*: ᛘ *"Contrôle whitespace"*, ᛘ *"Résumé corrections"*, ᛘ *"Fichiers corrigés"*
🔧 ✅ *reading project*: 🔎 e *"Refs drag PromptInput"*, 📄 *Prompt.fallbackPersistAck.clearAttachments.test.tsx* [L1-L100], 📄 *Prompt.fallbackPersistAck.clearInput.test.tsx* [L1-L140], 📄 *PromptFallbackPersistBarrier.testHarness.tsx* [L400-L494], 📄 *useAiStream.hook.liveSessionObserver.test.ts* [L170-L280]
🔧 ✅ *patching file*: 📄 *useAiStream.hook.liveSessionObserver.test.ts*
AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
Grouped
(what is it?)

🔧 ✅ *reading project*: 📄 *useAiStream.hook.ts* [L945-L975]
🔧 ✅ *scanning git*: ᛘ *"Mutations Stop restantes"*
🔧 ✅ *reading project*: 🔎 e *"Attentes stack erreur"*, 🔎 e *"API notices Prompt"*, 📄 *usePromptToolCallNotice.hook.ts* [L460-L600], 📄 *usePromptRunController.liveSessionObserver.test.tsx* [L20-L65]
🔧 ✅ *patching file*: 📄 *usePromptRunController.liveSessionObserver.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx
Grouped
(what is it?)

🔧 ✅ *scanning recent files*
🔧 ✅ *reading project*: 📄 *usePromptComposerAttachments.hook.ts* [L1-L120]
🔧 ✅ *patching file*: 📄 *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearInput.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *tsconfig.json* [L1-L160], *tsconfig.json* [L1-L160], *tsconfig.json* [L1-L160], *useAiStream.hook.ts* [L780-L850], *PromptFallbackPersistBarrier.testHarness.tsx* [L235-L350], *PromptFallbackPersistBarrier.testHarness.tsx* [L350-L410]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptInput.batchLiveViewerMode.test.tsx* [L1-L150]
🔧 ✅ *patching file*: *PromptInput.batchLiveViewerMode.test.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Annulation handler assertions"*, 🔎 e *"Construction adaptateur Batch"*, *EasyAgentsBatchRunManager.service.ts* [L190-L220], 🔎 e *"Construction service Retry"*, 🔎 h *"Tests détails erreur source"*, *PromptInput.tsx* [L2080-L2170], 🔎 e *"Pending removal signatures"*, *PromptInput.tsx* [L500-L740], *PromptInput.tsx* [L730-L780], 🔎 e *"Signature remove pending"*, 🔎 e *"Signature pending doc"*
🔧 ✅ *patching file*: *PromptInput.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *usePromptRunController.finalization.hook.ts* [L1-L220], 🔎 e *"Suffixe hydratation"*, *usePromptLiveSession.normalPromptIsomorphism.test.tsx* [L285-L390], *usePromptLiveSession.normalPromptIsomorphism.test.tsx* [L240-L310]
🔧 ✅ *patching file*: *usePromptLiveSession.normalPromptIsomorphism.test.tsx*
AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx* [L300-L345]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *SidebarStore.rehydrate.manager.test.ts* [L1-L95], 🔎 e *"Type view model Prompt"*, *Prompt.tsx* [L55-L160], *Prompt.tsx* [L250-L350], *Prompt.tsx* [L770-L815]
🔧 ✅ *patching file*: *Prompt.tsx*
AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Annulation réservation registre"*, *LiveSessionRegistry.service.ts* [L190-L235], *LiveSessionRegistry.service.ts* [L155-L200], *SidebarQueryRunningBridge.race.test.ts* [L90-L145]
🔧 ✅ *patching file*: *SidebarQueryRunningBridge.race.test.ts*
AICode/src/extension/src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Assertions sourceSettled"*, *jest.config.ts* [L1-L120], *tsconfig.jest.json* [L1-L100], 🔎 e *"Réponse draftAddDoc"*, *usePendingDocQueue.hook.ts* [L680-L735], *useAiStream.hook.liveSessionObserver.test.ts* [L1-L320]
🔧 ✅ *patching file*: *useAiStream.hook.liveSessionObserver.test.ts*
AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
Grouped
(what is it?)

🔧 ✅ *reading project*: *usePromptRunController.liveSessionObserver.test.tsx* [L1-L220]
🔧 ✅ *patching file*: *usePromptRunController.liveSessionObserver.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *frontSessionHydration.usageMapping.util.ts* [L1-L150], *SidebarQueryRunningBridge.ts* [L1-L25]
🔧 ✅ *patching file*: *SidebarQueryRunningBridge.ts*
AICode/src/extension/src/messageBus/handlers/ia/SidebarQueryRunningBridge.ts
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptInput.batchLiveViewerMode.test.tsx* [L200-L260]
🔧 ✅ *patching file*: *PromptInput.batchLiveViewerMode.test.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *PromptInput.batchLiveViewerMode.test.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx*
🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Type sélection Prompt"*, *EasyAgentsBatchRunRetryMessages.const.ts* [L1-L30], *EasyAgentsBatchRunRetry.guards.service.test.ts* [L1-L140], *EasyAgentsBatchRunRetry.guards.service.test.ts* [L140-L175]

🔧 ✅ *patching file*: *EasyAgentsBatchRunRetry.guards.service.test.ts*
AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.guards.service.test.ts
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Usage sync external store"*, *Prompt.tsx* [L55-L150], *Prompt.tsx* [L245-L320], *Prompt.tsx* [L455-L555], *Prompt.tsx* [L730-L790], *EasyAgentsBatchLiveSessionAdapter.service.ts* [L1-L210], *aiStopSuffix.util.ts* [L80-L255], *iaStreamHandlers.ts* [L195-L250]
🔧 ✅ *patching file*: *Prompt.fallbackPersistAck.clearAttachments.test.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *usePromptLiveSession.normalPromptIsomorphism.test.tsx*
AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Grouped
(what is it?)

🔧 ✅ *patching file*: *PromptInput.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: 🔎 e *"Viewer mode restant"*
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptFallbackPersistBarrier.testHarness.tsx* [L112-L175]
🔧 ✅ *patching file*: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *Prompt.liveSessionBootstrap.test.tsx* [L1-L180], *.prettierrc*, *eslint.config.ts* [L1-L200], *package.json* [L1-L120], 🔎 h *"Lint frontend prettier"*, *PackageScriptsLocalLint.service.ts* [L160-L215], *AttachList.tsx* [L1-L180]
🔧 ✅ *patching file*: *PromptInput.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.tsx
Grouped
(what is it?)

🔧 ❌ *patching file*: *PromptInput.batchLiveViewerMode.test.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Error
(what is it?)

🔧 ✅ *reading project*: *PromptInput.batchLiveViewerMode.test.tsx* [L45-L100], *PromptInput.batchLiveViewerMode.test.tsx* [L100-L135], *PromptInput.batchLiveViewerMode.test.tsx* [L215-L270]
🔧 ✅ *patching file*: *PromptInput.batchLiveViewerMode.test.tsx*
AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *PromptInput.tsx* [L2090-L2195], *usePromptLiveSession.normalPromptIsomorphism.test.tsx* [L1-L340]
🔧 ❌ *patching file*: *usePromptLiveSession.normalPromptIsomorphism.test.tsx*
AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Error
(what is it?)

🔧 ✅ *patching file*: *usePromptLiveSession.normalPromptIsomorphism.test.tsx*
AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
Grouped
(what is it?)

🔧 ✅ *reading project*: *AttachListItemProblems.tsx* [L1-L100]

Corrections appliquées ✅

Joel, toutes les anomalies **applicatives et testables dans le périmètre autorisé** ont été corrigées. Le mille-feuille asynchrone a maintenant nettement moins de miettes entre les couches 😄

Domaine	Corrections réalisées	F
Réservation de source Prompt	Annulation systématique de la réservation encore active après le transport. Le cas où l'endpoint consomme le handle puis échoue avant `registerRun()` produit maintenant une retraite `reservationCancelled`, sans locator bloqué en `not_ready`.	`iaStreamHandl` source
Bootstrap live invalide	Initialisation de la page déplacée avant la classification du bootstrap. La fermeture du panneau est tentée indépendamment du succès du dialogue. L'ancien chemin invalide mort à l'intérieur du runtime a été supprimé.	`Prompt.tsx` et te
Attachments avant replay	Verrouillage complet avant armement du replay : bouton Attach, picker, paste, drag-and-drop, ajouts, suppressions, clear et éléments pending. La liste devient réellement read-only, sans boutons de suppression, au lieu de seulement ignorer les clics.	`PromptInput.t` PromptInput
Retry destructif EasyAgents	Ajout d'une autorité directe du registre pour détecter une source ou réservation Prompt active, même si la projection Activity est absente ou périmée. Retry refuse désormais la destruction avant fermeture, éviction ou suppression disque.	adaptateur liveSes Retry et leurs tests
Protocole source V1	Le décodeur n'invente plus `status` ou `errorCode`. Les chaînes doivent contenir un envelope JSON réel ; les `Error` doivent posséder les propriétés structurées exactes. Une erreur ordinaire ou une simple chaîne préfixée est rejetée.	`aiStopSuffix.`
Diagnostics d'erreur	Les objets `Error` produisent désormais une ligne de stack concise et exploitable plutôt qu'une stack complète dépendante de l'environnement. Les détails textuels multilignes restent conservés tels quels.	tests des suffixes e
Stop transport transactionnel	Ajout d'une autorité synchrone par ref. La disponibilité Stop est consommée avant le premier `await`, empêchant deux aborts lors d'un double clic dans la même pile React.	`useAiStream.h`
Activity	Reconstruction selon la priorité `runtime → ancien store → panneau`	`SidebarStore.`

Domaine	Corrections réalisées	F
rehydrate	`→ persistance → défaut`, avec conservation du titre, de `openedAt`, de l’usage et des durées uniquement lorsque l’identité du run correspond.	
Durée terminale Activity	La durée finale est calculée exactement depuis `sourceStartedAtMs` avant suppression du runtime ; elle ne reste plus figée sur le précédent tick du timer d’une seconde.	runtime Activity e
Politiques adaptateur Prompt	`sendEnabled`, `historyInteractionsEnabled` et `cleanupOwned` sont maintenant réellement consommés par Prompt pour les hotkeys, l’historique et l’enregistrement du cleanup, au lieu d’être seulement déclarés dans le view model.	`Prompt.tsx`
Barrières fallback persist	Restauration de vrais tests React : ack exact `{turnId, streamInstanceId}`, ack stale, conservation sans ack du texte, des attachments, de `nextTurnId`, de l’état Edit, du mode maximisé et restauration du focus.	test attachments, t finalisation et harr
Tests observer frontend	Les anciennes assertions d’objets ont été remplacées par des montages des vrais hooks : acceptation/refus, scaffold assistant unique, identité exacte, route Stop source, double clic et terminal suivi d’une erreur transport sans second terminal.	test useAiStream observer
Isomorphisme normal Prompt	Comparaison renforcée entre les plans de streaming et l’hydratation réelle pour replace, continuation, thinking, texte, usage, terminal, erreur durable et suffixe unique.	test d’isomorphisn
Périmètre protégé	Aucun fichier généré n’a été modifié. L’arbre personnel interdit est resté entièrement hors de cette correction.	—

Résultat

- Les **cinq bugs runtime prioritaires** identifiés sont corrigés.
- Les politiques frontend auparavant déclaratives sont maintenant appliquées.
- Les protections Activity et Retry utilisent de nouveau les autorités exactes.
- Les principaux tests qui avaient été réduits à des assertions statiques exercent désormais les composants et hooks réels.
- Les changements restent strictement typés, sans `any`, sans contournement lint, sans fallback historique implicite et sans modification manuelle de fichiers générés.

Nom de commit proposé

text
Copy
```
fix(live-session): close replay races and restore behavioral coverage
```

🎵 Les réservations rentrent au bercail 🎷 Les replays restent sur les rails 🎶
</ASSISTANT>
<USER>
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *reading project*: *useAiStream.hook.liveSessionObserver.test.ts* [L1-L285], *PromptFallbackPersistBarrier.testHarness.tsx* [L260-L355], *useAiStream.hook.promptSourceIdentity.test.ts* [L1-L90], *Prompt.fallbackPersistAck.clearInput.test.tsx* [L1-L80], *prompt-stream-dispatch-lock.programming-doc-readme.md*, *fallback-persist-and-title.programming-doc-readme.md*, *PromptFallbackPersistBarrier.testHarness.tsx* [L355-L400]
🔧 ✅ *patching file*: *useAiStream.hook.liveSessionObserver.test.ts*
AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
Copy
Review

Accept
PatchFull file

```
- import { waitFor } from '@testing-library/react';
-
-
+ import { waitFor } from '@testing-library/react';
+
+ import { AiStopErrorReason } from
'../../sharedlib/types/aiTypes/AiStopErrorReason.type.js';
+
```

```
-                 value: {
-                     reason: 'system_error',
-
+                 value: {
+                     reason: AiStopErrorReason.SYSTEM_ERROR,
+
```

AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
Copy
Review
Accept

patching file: *PromptFallbackPersistBarrier.testHarness.tsx*
AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Copy
Review
Accept
PatchFull file

```
-     }): JSX.Element => {
-
+     }): JSX.Element => {
+         // Destructure callback inputs so React hook dependencies track only the values
each callback consumes.
+         const { messages, onRequestEditLastPrompt } = props;
+
```

```
-         const handleEdit = useCallback((): void => {
-             props.onRequestEditLastPrompt({
-
+         const handleEdit = useCallback((): void => {
+             onRequestEditLastPrompt({
+
```

```
-             });
-         }, [props.onRequestEditLastPrompt]);
-
+             });
+         }, [onRequestEditLastPrompt]);
+
```

```
-                 <div data-testid="historyMessageCount">
-                     {props.messages.length}
-
+                 <div data-testid="historyMessageCount">
+                     {messages.length}
+
```

```
-     ): JSX.Element {
-
+     ): JSX.Element {
+         // Keep callback dependencies precise instead of depending on the unstable
aggregate props object.
+         const {
```

```
+             promptValue,
+             submitTxnPending,
+             attachments,
+             attachmentsTurnId,
+             isEditLastPromptActive,
+             composerMaximized,
+             onChangePromptValue,
+             onChangeComposerMaximized,
+             onSend,
+         } = props;
+
-------------------------------------------------------------------------------
-             (event: ChangeEvent<HTMLTextAreaElement>): void => {
-                 props.onChangePromptValue(event.target.value);
-
+             (event: ChangeEvent<HTMLTextAreaElement>): void => {
+                 onChangePromptValue(event.target.value);
+
-------------------------------------------------------------------------------
-             },
-             [props.onChangePromptValue],
-
+             },
+             [onChangePromptValue],
+
-------------------------------------------------------------------------------
-         const handleSend = useCallback((): void => {
-             props.onSend(props.promptValue);
-         }, [props.onSend, props.promptValue]);
-
+         const handleSend = useCallback((): void => {
+             onSend(promptValue);
+         }, [onSend, promptValue]);
+
-------------------------------------------------------------------------------
-         const handleMaximize = useCallback((): void => {
-             props.onChangeComposerMaximized(true);
-         }, [props.onChangeComposerMaximized]);
-
+         const handleMaximize = useCallback((): void => {
+             onChangeComposerMaximized(true);
+         }, [onChangeComposerMaximized]);
+
-------------------------------------------------------------------------------
-                     data-testid="promptInput"
-                     disabled={props.submitTxnPending === true}
-
+                     data-testid="promptInput"
+                     disabled={submitTxnPending === true}
+
-------------------------------------------------------------------------------
-                     value={
-                         props.submitTxnPending === true ? '' : props.promptValue
-
+                     value={
+                         submitTxnPending === true ? '' : promptValue
+
-------------------------------------------------------------------------------
-                 <div data-testid="attachmentCount">
-                     {props.attachments.length}
-
+                 <div data-testid="attachmentCount">
+                     {attachments.length}
+
-------------------------------------------------------------------------------
-                 <div data-testid="attachmentsTurnId">
```

```
-                     {props.attachmentsTurnId ?? ''}
-
+                 <div data-testid="attachmentsTurnId">
+                     {attachmentsTurnId ?? ''}
+
```

```
-                 <div data-testid="editMode">
-                     {props.isEditLastPromptActive ? 'edit' : 'normal'}
-
+                 <div data-testid="editMode">
+                     {isEditLastPromptActive ? 'edit' : 'normal'}
+
```

```
-                 <div data-testid="maximized">
-                     {props.composerMaximized ? 'yes' : 'no'}
-
+                 <div data-testid="maximized">
+                     {composerMaximized ? 'yes' : 'no'}
+
```

AICode/src/frontend/src/app/prompt/Prompt/__test__/PromptFallbackPersistBarrier.testHarness.tsx
Copy
Review
Accept

reading project: *useAiStream.hook.liveSessionObserver.test.ts* [L1-L15], *useAiStream.hook.liveSessionObserver.test.ts* [L245-L265], *PromptFallbackPersistBarrier.testHarness.tsx* [L270-L375]

scanning git: *"Vérifier le diff ciblé"*

Diagnostics corrigés

J'ai appliqué **LLM-DEBUG-3** : les cinq diagnostics provenaient exclusivement du code de test ajouté, pas du code applicatif.

Diagnostic	Cause racine	
Type incompatible pour `system_error`	Le test utilisait une chaîne littérale alors que le contrat exige l'enum strict `AiStopErrorReason`.	Import de `AiStopErrorReason` et de `AiStopErrorReason.SYSTEM_`
Dépendance React manquante dans `handleEdit`	Le callback lisait une propriété depuis l'objet agrégé `props`, que la règle Hooks considère comme une dépendance supplémentaire.	Destructuration de `messages` et `onl` uniquement.
Dépendance React manquante dans `handleChange`	Même accès indirect via l'objet `props`.	Destructuration de `onChangePromp`
Dépendance React manquante dans `handleSend`	Le callback lisait `props.onSend` et `props.promptValue`.	Destructuration de `onSend` et `promp`
Dépendance React manquante dans `handleMaximize`	Le callback lisait `props.onChangeComposerMaximized`.	Destructuration de `onChangeCompo`

Les quatre corrections React sont regroupées dans `PromptFallbackPersistBarrier.testHarness.tsx`. Les rendus du mock utilisent également les valeurs destructurées afin de conserver une source de vérité unique et lisible.

Résumé

Fichier	Modification	Complétude
test observer liveSession	Remplacement de la chaîne permissive par l'enum strict.	✅
harness des barrières fallback	Destructuration ciblée des props et tableaux de dépendances React exhaustifs.	✅
Total	**1 erreur TypeScript et 4 warnings React Hooks corrigés.**	✅

Nom de commit proposé

text
Copy
```
fix(live-session): resolve test typing and hook dependency diagnostics
```

🎵 Les types retrouvent leur accord 🎻 Les hooks ne préviennent plus à tort 🎶
</ASSISTANT>
<USER>
Summary of all failing tests FAIL src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts ● SidebarStore activity rehydration › restores metadata in runtime, store, panel, persistence, then default priority order expect(received).toEqual(expected) // deep equality - Expected - 2 + Received + 18 - ObjectContaining { + Object { + "liveQueryElapsedMs": 1783941706892, "openedAt": 70, - "title": "Panel title", + "ownership": "source", + "panelState": "open", + "runRef": Object { + "locator": Object { + "sessionId": "11111111111111111111111111111111", + "storageScopeRelPath": "", + }, + "sourceId": "source-panel", + "turnId": "turn-panel", + }, + "sessionId": "11111111111111111111111111111111", + "status": "running", + "stopRequested": false, + "storageScopeRelPath": "", + "title": "�� Panel title", + "updatedAt": 1783941706912, } 420 | }), 421 |); > 422 | expect(panelItem).toEqual(| ^ 423 | expect.objectContaining({ 424 | title: 'Panel title', 425 | openedAt: 70, at Object.<anonymous> (src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts:422:27) FAIL src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts ● iaStreamHandlers source lifecycle › cancels a consumed reservation when transport fails before source promotion expect(received).toEqual(expected) // deep equality Expected: ArrayContaining ["transport", "unstage", "cancel", "finalize", "soundStop", "coordinator"] Received: ["reserve", "stage", "activityStart", "soundStart", "transport", "cancel", "finalize", "soundStop", "coordinator"] 255 |).rejects.toThrow('failed before promotion'); 256 | > 257 | expect(order).toEqual(| ^ 258 | expect.arrayContaining([259 | 'transport', 260 | 'unstage', at Object.<anonymous> (src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts:257:23) Test Suites: 2 failed, 822 passed, 824 total Tests: 2 failed, 2656 passed, 2658 total Snapshots: 1 passed, 1 total Time: 69.681 s Ran all test suites. Command failed: C:\Program Files\nodejs\node.exe --expose-gc --max-old-space-size=8192 --experimental-vm-modules C:\dev\AICode\src\extension\node_modules\jest\bin\jest.js --config C:\dev\AICode\src\extension\jest.config.ts --maxWorkers 10 (exit 1)
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *reading project*: liveSession.programming-doc-readme.md, SidebarStore.rehydrate.manager.ts [L1-L430], SidebarStore.rehydrate.manager.test.ts [L1-L450], iaStreamHandlers.ts [L1-L280], iaStreamHandlers.queryDoneSound.test.ts [L1-L285], SidebarStore.queryWindow.listeners.manager.ts [L1-L210], openPanel.util.ts [L250-L360], SidebarStore.chatTitles.manager.ts [L1-L200], 🔎 e "Préfixes onglet Prompt", ChatConfigConst.const.ts [L18-L40], 🔎 e "Préfixes obsolètes test", 🔎 e "Mock unstage test"
🔧 ✅ *patching file*: SidebarStore.rehydrate.manager.test.ts
AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts
Copy
Review
Accept
PatchFull file

```
- import type * as vscode from 'vscode';
-
+ import type * as vscode from 'vscode';
+ import { ChatConfigConst } from
'../../../../aiTools/orch/context/persistence/ChatConfigConst.const.js';
+
```

```
-             buildPromptPanelKey(openLocator),
-             panel('🤖 Open source'),
-
```

```
+             buildPromptPanelKey(openLocator),
+             panel(
+                 `${ChatConfigConst.CONFIG_PROMPT_TAB_TITLE_PREFIX_IDLE}Open source`,
+             ),
+
```

```
-             buildPromptPanelKey(runtimeLocator),
-             panel('🤖 Panel fallback'),
-
+             buildPromptPanelKey(runtimeLocator),
+             panel(
+                 `${ChatConfigConst.CONFIG_PROMPT_TAB_TITLE_PREFIX_IDLE}Panel fallback`,
+             ),
+
```

```
-             buildPromptPanelKey(panelLocator),
-             panel('🤖 Panel title'),
-
+             buildPromptPanelKey(panelLocator),
+             panel(
+                 `${ChatConfigConst.CONFIG_PROMPT_TAB_TITLE_PREFIX_IDLE}Panel title`,
+             ),
+
```

AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts
Copy
Review
Accept

🔧 ✅ *patching file*: *iaStreamHandlers.queryDoneSound.test.ts*
AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts
Copy
Review
Accept
PatchFull file

```
-         // post-forwarding finalization so the locator cannot remain permanently not-
ready.
-         unstageReservationMock.mockReturnValue(false);
-
+         // post-forwarding finalization so the locator cannot remain permanently not-
ready.
+         unstageReservationMock.mockImplementation(() => {
+             // Preserve the order instrumentation while returning the consumed-
capability outcome.
+             order.push('unstage');
+             return false;
+         });
+
```

AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts
Copy
Review
Accept

🔧 ✅ *reading project*: *SidebarStore.rehydrate.manager.test.ts* [L1-L18], *SidebarStore.rehydrate.manager.test.ts* [L88-L115], *SidebarStore.rehydrate.manager.test.ts* [L340-L365], *iaStreamHandlers.queryDoneSound.test.ts* [L215-L240]
🔧 ✅ *scanning git*: *"Contrôle des deux tests"*
Correctif appliqué avec LLM-DEBUG-3 ✅
Les deux échecs provenaient **des tests eux-mêmes**, pas du code applicatif.

1. Réhydratation Activity — préfixe d'onglet obsolète
Dans `SidebarStore.rehydrate.manager.test.ts`, le test simulait un titre de panneau avec le préfixe historique 🤖 .
Or le code de production retire volontairement **uniquement les préfixes Prompt canoniques** :

- 💭 pour une session idle ;
- ⏳ pour une session running.

Le helper de production était donc correct : il ne devait pas retirer un préfixe inconnu. Le test a été corrigé pour construire ses titres avec `ChatConfigConst.CONFIG_PROMPT_TAB_TITLE_PREFIX_IDLE`, ce qui :

- évite un emoji codé en dur ;
- suit automatiquement le contrat du projet ;
- vérifie réellement la suppression du préfixe canonique ;
- conserve l'attente correcte `title: 'Panel title'`.

La grande valeur de `liveQueryElapsedMs` visible dans le diff Jest n'était pas la cause de l'échec : `expect.objectContaining(...)` l'ignorait. Elle venait simplement du timestamp synthétique `sourceStartedAtMs: 20`.
2. Lifecycle source — instrumentation du mock supprimée
Dans `iaStreamHandlers.queryDoneSound.test.ts`, cette configuration :
`unstageReservationMock.mockReturnValue(false)`
remplaçait complètement l'implémentation précédente du mock, y compris son instrumentation :
`order.push('unstage')`.
Le code applicatif appelait bien `unstageNormalPromptLiveSessionReservation(...)`, mais le test supprimait lui-même la trace censée prouver cet appel.
Le mock utilise désormais une implémentation explicite qui :

1. enregistre `unstage` dans l'ordre d'exécution ;
2. retourne `false` pour simuler un handle déjà consommé par l'endpoint ;
3. permet ensuite de vérifier l'ordre exact `unstage → cancel → finalize`.

Relance ciblée conseillée
Depuis le projet extension :

bash
Copy
```
node --expose-gc --max-old-space-size=8192 --experimental-vm-modules
node_modules/jest/bin/jest.js --config jest.config.ts --runTestsByPath
src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts
src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts --runInBand
```

Après cette vérification ciblée, relancez la suite complète afin de confirmer les **824 suites**.
Résumé des modifications

Test	Origine de l'échec	Correction	Code applicatif modifié	État
réhydratation Activity	Fixture utilisant l'ancien préfixe 🤖 , non reconnu volontairement par le helper strict	Utilisation du préfixe canonique provenant de `ChatConfigConst`	Non	✅
lifecycle source Prompt	`mockReturnValue(false)` supprimait l'instrumentation `order.push('unstage')`	`mockImplementation(...)` conserve la trace tout en retournant `false`	Non	✅
Total	**Deux erreurs de test**	**Deux tests corrigés sans altérer le comportement produit**	**Non**	✅

Nom de commit proposé

text
Copy
```
test(live-session): fix rehydration prefix and reservation-order fixtures
```

🎵 Les mocks gardent enfin leur mémoire 🎹 Les préfixes racontent la bonne histoire 🎶
</ASSISTANT>
SHOW ALLPage 1 / 1