Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=1382ef772f8b4cf8ae82c3a7bccc03a6-RS-Affiner-finale-de
<USER>
[Refine]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_004205
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *scanning git*: ᛘ *"Vérifier les commits"*, ᛘ *"Comparer les tranches"*
🔧 ✅ *reading project*: *liveSession.programming-doc-readme.md*, *isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md*, *thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md*, *easyAgents.programming-doc-readme.md*, 🔎 h *"Architecture liveSession"*, 🔎 e *"État activité windows"*, 🔎 e *"Ouvertures panneau Prompt"*, 🔎 e *"Démarrage stream IA"*
🔧 ✅ *scanning git*: ᛘ *"Évolutions depuis réaudit"*, ᛘ *"Historique récent ciblé"*
🔧 ✅ *reading project*: 👉 *"Cœur et points d'entrée"*, 👉 *"Panneaux et activité"*, 👉 *"Frontend stream et Prompt"*,
🔎 r *"openCentralPanel"*,
🔎 r *"cleanupOnPromptDispose"*,
🔎 r *"closePromptPanel"*, 🔎 r *"useAiStream"*, 🔎 r *"PromptQueryDoneSoundManager"*, 🔎 e *"Ouverture bus Prompt"*, 🔎 e *"Fermeture Prompt"*, 🔎 h *"Continuation et mémoire"*,
iaStreamHandlers.ts,
useAiStream.hook.ts,
usePromptLiveSession.hook.ts, *PromptThreadCleanup.manager.ts*, *openPanel.util.ts* [L378-L720],
SidebarStore.handlers.manager.ts,
SidebarUiMessages.ts,
MessageRegistry.ts,
liveSessionStreamHandlers.ts,
PromptQueryDoneSound.manager.ts,
AiStreamStartArgs.type.ts,
LiveSessionRun.type.ts,
LiveSessionOpenResult.type.ts,
LiveSessionRecord.type.ts,
LiveSessionOpen.service.ts, *EasyAgentsBatchLiveSessionAdapter.service.ts*, 🔎 e *"Assertions stream payload"*, 🔎 e *"RunRef fixtures live"*, 🔎 h *"Stop et clear thread"*,
VscButton.tsx,
VscIcon.tsx,
closePromptPanel.util.ts,
AiPromptUiMessages.ts,
uiAiPromptHandlers.ts,
SidebarStore.queryWindow.listeners.manager.ts,
SidebarStore.queryWindow.mutations.manager.ts, *SidebarStore.rehydrate.manager.ts*, 🔎 e *"Icône arrêt carrée"*, 🔎 e *"Option disableAfterClick"*, *AiQuery.type.ts*, *usePromptRunCommands.hook.ts* [L1-L360],
preparePromptStreamingTurn.util.ts,
usePromptThinkingLifecycle.hook.ts, *usePromptRunController.finalization.hook.ts*, *usePromptToolUse.hook.ts*, 🔎 e *"Types requêtes IA"*, 🔎 e *"Types actions Prompt"*, *usePromptRunCommands.hook.ts* [L340-L570], 🔎 e *"Lecture messages mémoire"*, 🔎 e *"Queue par session"*, 🔎 h *"Accès mémoire Orch"*, *OrchMemory.ts* [L560-L700], *OrchMessagesMemory.ts* [L1-L240], *ChatPersistenceWriteQueue.service.ts* [L1-L240], *OrchAsk.ts*, 🔎 e *"Accès direct messagesMemory"*, 🔎 e *"Mémoire Orch publique"*, 🔎 e *"Accès orchAsk memory"*, 🔎 e *"Exposition messages"*, *OrchAskStream.ts* [L1-L220], *OrchContextBuilder.ts* [L1-L180], 🔎 e *"Context builder stream"*, 🔎 e *"Méthode snapshot mémoire"*, 🔎 e *"Snapshot messages thread"*,
PromptLiveSessionBanner.tsx,

PromptLiveSession.types.ts, EasyAgentsBatchLiveSession.adapter.ts, e "Modes runtime Prompt", e "Vue live lecture seule", e "Banner liveSession", PromptInput.tsx [L700-L920], e "View model PromptInput", e "Viewer mode PromptInput", PromptInput.tsx [L1450-L1710], e "Type activité historique"

scanning git: "Comparer tranche et HEAD"

reading project: e "Handoff normal atomique", e "Settlement source exact", LiveSessionRegistry.service.ts [L228-L620], LiveSessionService.ts [L100-L220]

specifying "SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_013420"

SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_013420

Copy

Refine

Code

Verify

Markdown formattedMarkdown source

Spécification — Finaliser `liveSession` pour les sessions Prompt normales

Contexte

La tranche 1 correspond au sous-système fonctionnel présent au commit `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`. La présente réinspection a été reprise depuis zéro sur le dépôt courant au commit `7a39caa251158f63a6a34d6ca3cf2a0e82993425`. Le diff entre ces deux commits ne touche que quatre fichiers de génération d'instructions SPECIF, sans modifier les contrats ou chemins d'exécution concernés par cette spécification ; la base fonctionnelle `liveSession` reste donc celle de la tranche 1.

Cette fondation fournit déjà :

- un registre process-local de runs exacts ;
- un buffer ordonné complet de `AiStreamedChunk` ;
- des leases de viewer exactes ;
- les barrières distinctes `sourceSettled` et `handoffReady` ;
- l'ouverture live-safe sans reset de la mémoire Orch ;
- le replay backlog + tail ;
- la protection anti-ABA par époque de locator ;
- le terminal exact et la distinction `TOOL_USE` / vraie fin de tour ;
- les transports Slimbk et message bus génériques ;
- un hook frontend générique réutilisant Queue A, Queue B, les locks et les sanitizers Prompt ;
- un adaptateur EasyAgents qui conserve le comportement produit historique.

La tranche finale doit activer ce mécanisme pour les sessions Prompt AICode normales en une seule implémentation cohérente. Le couplage entre capture de la source, fermeture du panneau, projection Activité, réouverture, replay, contrôle Stop, son de fin et nettoyage terminal doit être traité dans le même changement afin qu'aucun contrat intermédiaire incomplet ne soit livré.

Le problème utilisateur à résoudre est le suivant : un renderer Prompt peut devenir gris, notamment après une saturation mémoire Electron. Aujourd'hui, fermer ce panneau déclenche explicitement `Abort → AwaitStop → ClearMemory`, détruit la génération en cours et oblige à relancer le dernier message depuis zéro. Après cette tranche :

- fermer un panneau Prompt normal pendant une génération ne doit plus arrêter cette génération ;
- la génération continue dans l'extension ;
- la session reste visible dans Activité pendant son exécution ;
- cliquer la session dans Activité ou dans la liste des chats rouvre le Prompt sur la source exacte encore active ;
- le nouveau panneau reconstruit l'historique et rattrape le stream sans perte ni duplication ;
- si le panneau reste fermé jusqu'à la fin du processus, l'item disparaît d'Activité ;
- si le panneau est ouvert à la fin, l'item reste en état Idle comme aujourd'hui ;
- un bouton Stop apparaît sur une troisième ligne d'Activité uniquement pour une source Prompt normale, lorsque le panneau est fermé et que cette source est encore active.

La réinspection complète du code confirme l'architecture générale, mais impose les invariants supplémentaires suivants :

- l'Activité doit distinguer les runs `source` des runs `observer`, sinon une fermeture EasyAgents créerait à tort un item détaché avec bouton Stop ;
- le coordinateur de panneau ne doit pas importer le low-level opener qui l'instancie, sinon la conception crée un cycle ESM ;
- une query liveSession EasyAgents explicitement fournie par le caller doit être préservée et ne doit jamais être remplacée par une ouverture classique ;

- `sourceId` doit identifier la source backend exacte, mais il ne suffit pas à distinguer deux tentatives frontend de replay de la même source : un `streamInstanceId` frontend-only doit donc identifier chaque appel accepté de `useAiStream.start()` ;
- les callbacks frontend `onData`, `onComplete` et `onError`, y compris leurs continuations asynchrones après drainage des queues, doivent être gardés par le couple `sourceId + streamInstanceId` ;
- le handoff direct normal ne peut être déclenché que par un vrai terminal reçu dans les données du replay et par une complétion transport normale ; une erreur de transport synthétique ou une complétion sans vrai terminal déclenche la récupération, jamais un faux passage à `normal` ;
- `continueResponse` doit réutiliser la bulle thinking agrégée existante et son texte de base ; l'initialisation de ce texte doit être corrélée au `streamInstanceId`, y compris lors d'un retry transport de la même source ;
- la baseline de continuation doit figer les messages ciblés, mais ne doit pas figer le brouillon, les attachments, `nextTurnId`, le titre ou les autres champs d'enveloppe qui peuvent évoluer après le début du stream ; chaque Open doit superposer les messages immuables de continuation sur l'enveloppe persistée la plus récente ;
- les erreurs de source transformées en header Slimbk doivent utiliser un encodage réversible et versionné du `AiErrorMetaValue` ; le buffer et le frontend continu doivent reconstruire exactement la même erreur ;
- le client Slimbk généré n'attend pas les callbacks `onChunk` asynchrones : le handler normal doit donc maintenir et attendre une pipeline FIFO explicite avant ses cleanups, faute de quoi `usage`, terminal, Activity et forwarding peuvent être réordonnés ;
- pour une source Prompt normale, `sourceSettled` et `handoffReady` ne doivent pas être publiés par l'endpoint backend avant que le handler extension ait entièrement drainé sa pipeline FIFO de forwarding : sinon le pointeur actif disparaît alors que des chunks, le son et l'Activité sont encore en transit, et une fermeture de panneau peut déclencher un cleanup destructif prématuré ;
- la finalisation normale du registre doit donc être une opération atomique exécutée par le handler extension après le drainage de la pipeline : réservation restante annulée comme non démarrée, ou record exact marqué `sourceSettled` puis `handoffReady` uniquement s'il possède baseline et vrai terminal ;
- le son de fin doit lui aussi être corrélé au run exact `{ locator, turnId, sourceId }`, sinon le cleanup tardif d'une ancienne source partageant le même turn peut désarmer ou déclencher le son de la nouvelle source ;
- la libération de lease lors du dispose doit être dispatchée par adaptateur : EasyAgents doit continuer à passer par sa policy `last viewer → kill`, tandis que `normalPrompt` libère seulement la lease et laisse la source continuer ;
- le coordinateur et la réhydratation Activité doivent pouvoir observer tout run actif, source ou observer, tandis que deletion/Retry continuent d'utiliser l'autorité plus étroite « source normale active ou réservée » ;
- la métadonnée de retraite terminale doit conserver l'époque terminale du run ; une éviction tardive de l'ancien run après le démarrage d'une source plus récente ne doit jamais rendre l'ancien terminal de nouveau éligible au fallback persistant ;
- les suppressions de chat et le Retry EasyAgents doivent consulter le registre `liveSession` comme autorité, et non dépendre seulement du SidebarStore de présentation.

  La tranche doit préserver les invariants suivants :
- aucune modification des algorithmes Queue A, Queue B, `PromptStreamLockStore` ou des sanitizers ;
- aucune duplication du timer Activité ou du son de fin lors d'un replay ;
- aucun second bouton Stop dans le Prompt : le panneau reconnecté réutilise le bouton Stop normal du compositeur ;
- aucune notion de panneau « invisible » : le panneau est soit `open`, soit `closed` ;
- le compositeur normal doit être restauré dès que le snapshot de base est installé et que le replay est armé ; pendant la réponse, seul le verrou Send normal reste actif, la saisie du prochain brouillon et la préparation des attachments restent autorisées ;
- aucun `*.generated.*` ne doit être édité manuellement ; les fichiers générés seront régénérés par la compilation normale à partir des endpoints et contrats sources ;
- aucun fichier sous le répertoire documentaire personnel interdit ne doit être chargé ou modifié ;
- tous les commentaires, tests et identifiants de code ajoutés ou modifiés sont en anglais ;
- aucun `any`, aucun `eslint-disable`, aucun TODO, aucun cast inutile, aucune importation dynamique de production et aucune compatibilité implicite non demandée.

Objectifs décrits par l'utilisateur

Besoin initial exprimé par l'utilisateur

- Livrer toute la tranche finale en une seule implémentation, et non en plusieurs tranches fonctionnelles susceptibles de rater le couplage entre backend, panneau, frontend et Activité.
- Permettre à une génération Prompt normale de continuer lorsque son panneau est fermé.
- Conserver l'item dans Activité si le panneau est ouvert, ou si le panneau est fermé mais que le processus est encore en exécution.

- Supprimer l’item d’Activité si le panneau est fermé et que le processus est terminé.
- Ajouter une troisième ligne avec un bouton Stop uniquement lorsque le panneau est fermé et qu’une source Prompt normale est active.
- Utiliser pour ce bouton l’icône Stop existante, la taille `small`, la variante `danger`, le label `Stop`, et le désactiver immédiatement après clic.
- Faire disparaître la troisième ligne et le bouton dès que le panneau est rouvert ou que le processus se termine.
- Réutiliser dans le panneau reconnecté le bouton Stop normal déjà existant ; n’ajouter aucun second bouton Stop dans le Prompt.
- Restaurer l’état normal du Prompt au plus tôt : aucun read-only prolongé propre au mécanisme de reconnexion ; seule la courte phase nécessaire à l’installation du snapshot et à l’armement du replay peut masquer ou verrouiller le compositeur.
- Une fois le replay armé, afficher le compositeur normal, autoriser la saisie et les attachments du prochain message, mais conserver le verrou Send normal tant que le stream actuel est actif.
- Étendre SidebarStore pour représenter des activités de session et pas uniquement des fenêtres ouvertes.
- Conserver le buffer raw-live complet, sans compression ni limitation ajoutée.
- Tester tout code créé, déplacé ou modifié, y compris toutes les courses de fermeture, réouverture, arrêt, retry transport et succession de sources.
- Mettre à jour la programming doc générique `liveSession` et la documentation d’isomorphisme streaming/réhydratation.
- Respecter l’architecture modulaire existante : backend OOP composé, frontend en hooks/adaptateurs/composants unitaires, contrats stricts shared, commentaires techniques en anglais, aucun `any`, aucun cast inutile, aucun `eslint-disable`, aucun TODO et aucune importation dynamique.

Clarifications implicites ajoutées par AICode

- Le changement part fonctionnellement du commit `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`; le commit courant inspecté `7a39caa251158f63a6a34d6ca3cf2a0e82993425` ne contient aucune évolution pertinente de ce sous-système.
- `LiveSessionRunRef` devient obligatoirement `{ locator, turnId, sourceId }`. Le `turnId` reste la corrélation logique persistante des messages ; le `sourceId` identifie une invocation précise de la source backend.
- Un `streamInstanceId` frontend-only, distinct de `sourceId`, identifie chaque tentative de transport acceptée. Il change aussi lors d’un retry de replay de la même source et ne traverse jamais les contrats backend.
- Le `sourceId` est obligatoire dans toutes les routes/query strings liveSession et dans tous les guards de source. Aucune opération exacte ne retombe sur le run actif d’un locator lorsqu’une identité exacte échoue.
- Pour EasyAgents, l’adaptateur utilise explicitement `sourceId = turnId`, parce que son contrat garantit une source unique par tour.
- Pour les streams Prompt normaux, `useAiStream.start()` génère le `sourceId` seulement après l’acceptation du start et génère toujours un nouveau `streamInstanceId` pour cette tentative.
- Les callbacks d’une ancienne source ou d’une ancienne tentative de transport sont ignorés avant toute mutation, après tout `await`, et avant tout reset des queues/locks. Un ancien `onComplete` ne peut donc jamais arrêter ou réinitialiser une source/tentative plus récente.
- Le registre ajoute des réservations de source pour les streams normaux entre l’entrée du handler extension et le retour réussi de `askStream(...)`. La réservation est la première mutation synchrone du handler, avant tout `await`.
- Une réservation n’est pas un record replayable et ne viole pas l’invariant « seul `registerRun` crée un record ».
- Une réservation protège l’ouverture avant baseline, l’arrêt précoce, l’identité Activité et la fermeture immédiate du panneau. Elle est transformée atomiquement en record après le retour de `askStream(...)`, ou annulée et définitivement retirée si la source ne démarre pas.
- La transformation réservation → record conserve `sourceStartedAtMs` et `abortRequested`. Un abort réclamé avant l’enregistrement est appliqué immédiatement après le priming et l’enregistrement, avant l’interprétation du premier chunk.
- Le registre conserve pour les runs évincés une retraite structurée. Une retraite terminale mémorise l’époque terminale du run et n’est éligible que si aucune transition ultérieure du locator n’a eu lieu. Une éviction TTL ne peut actualiser cette époque que si le run est toujours le dernier terminal du locator.
- Une réservation annulée, un run pré-baseline évincé, une éviction destructive ou un ancien terminal évincé après le démarrage d’une source plus récente ne créent jamais un fallback terminal autorisé.
- Le cœur prend en charge deux readiness policies : `baselineAndObservableChunk` pour EasyAgents, inchangée, et `baselineOnly` pour les sessions normales afin qu’elles puissent être rouvertes dès la baseline durable, avant le premier token visible.
- Le cœur transporte un `replayMode` strict : `replaceTurnAssistants` pour prompt/retry/edit/specif/compact-context et `continueAssistant` pour `continueResponse`.

- `continueResponse` capture sous sérialisation une baseline immuable des messages OrchMemory avant `askStream(...)`, en validant l'assistant ciblé. Le registre ne fige pas le brouillon ou l'enveloppe complète.
- À chaque Open `continueAssistant`, le backend recharge l'enveloppe persistée la plus récente et remplace uniquement son tableau `messages` par la copie défensive de la baseline. Les brouillons, attachments, `nextTurnId`, titres et métadonnées non conversationnelles écrits après le démarrage restent donc récupérables.
- `markBaselineReady` accepte une baseline de messages seulement pour `continueAssistant`; elle est interdite pour `replaceTurnAssistants` et pour les incohérences de ownership/readiness. Le registre clone défensivement à l'entrée et à la sortie.
- Le lifecycle du `Readable` lazy est extrait du child executor EasyAgents dans un service générique réutilisé par EasyAgents et par l'endpoint de stream normal. Il conserve l'ordre : création du Readable → première lecture pendante → source-start synchrone → traitement du premier résultat → tail → fermeture/abort/settlement exact.
- Le stream normal est enregistré avec ownership `source`. Le stream de replay n'ouvre jamais une seconde activité et ne joue jamais un second son.
- Le handler source normal reste l'unique propriétaire du timer Activité et du son de fin. Son forwarding est sérialisé par une pipeline FIFO locale explicitement attendue avant la finalisation du registre et les cleanups.
- Chaque chunk normal exposé au frontend est appendu au registre avant l'exposition.
- Le premier `meta.error` transformé en header Slimbk utilise un encodage privé, versionné et réversible du `AiErrorMetaValue`. Le frontend décode cet encodage pour reconstruire le chunk exact ; les erreurs de transport ordinaires restent canonisées en `SYSTEM_ERROR`.
- Les throws de source après baseline et les fins de stream sans vrai terminal sont transformés en un `meta.error` canonique et terminal, bufferisé puis exposé, afin que le replay ne puisse jamais atteindre un handoff normal sans terminal réel.
- Le service source normal consomme et canonise la source, mais ne retire pas lui-même le run de l'index actif. Après le retour du transport Slimbk et le drainage de tous les callbacks `onChunk`, le handler extension exécute l'opération atomique de finalisation normale : une réservation restante est annulée ; un record sans baseline est évincé non terminalement ; un record avec baseline doit posséder un vrai terminal, puis devient simultanément `sourceSettled` et `handoffReady`.
- Tant que cette finalisation post-forwarding n'a pas eu lieu, le run ou sa réservation reste visible comme source active pour le coordinateur, l'Activité, Stop, deletion et les guards de cleanup. Une fermeture pendant le drainage ne peut donc jamais nettoyer la mémoire, les attachments ou le son avant le dernier chunk forwardé.
- L'ouverture de tout panneau Prompt passe par une façade coordonnée commune, y compris les ouvertures depuis la liste des chats, Activité, les deep links, la commande New Chat et le bus générique.
- Une query liveSession explicite et valide fournie par EasyAgents garde la priorité et est transmise telle quelle. Une query liveSession explicite mais invalide n'est jamais silencieusement dégradée en ouverture classique ; le frontend conserve son erreur fail-fast actuelle.
- En l'absence de descripteur live explicite, le coordinateur choisit une query `normalPrompt` exacte si une source ownership `source` est réservée/active ; sinon il conserve la query classique du caller. Lorsqu'il injecte une query live normale, il retire les options auto-run incompatibles.
- Le coordinateur est une classe à dépendances injectées et n'importe jamais le module low-level qui le compose. Ses ports de release distinguent explicitement `easyAgentsBatch` et `normalPrompt`.
- Les opérations « ouverture de panneau », « dispose du panneau » et « nettoyage après fin d'une source détachée » sont sérialisées par locator avec des queues FIFO dédiées, supprimées lorsqu'elles redeviennent vides.
- Le low-level panneau retire synchroniquement le panneau du registre puis délègue l'ensemble des effets Prompt au coordinateur. La release de lease, les transitions Activité, le détachement de mapping, le cleanup mémoire, le cleanup son et le cleanup trivial ne sont pas exécutés hors de la queue du coordinateur.
- Le coordinateur réévalue le panneau et tout run actif exact juste avant un cleanup destructif. Un settlement tardif d'une ancienne source ne peut jamais nettoyer la mémoire ou le son d'une nouvelle source du même locator.
- Si le nettoyage terminal gagne la course, l'ouverture suivante est classique et réhydrate la mémoire ; si l'ouverture gagne, le nettoyage voit le panneau ouvert et ne clear pas la mémoire.
- Le mapping de cleanup Prompt est indexé par locator structurel complet et non plus par `sessionId` seul.
- Le panneau reconnecté utilise une policy frontend `normalPrompt`, distincte de la présentation viewer EasyAgents. L'état `liveSessionStreaming` n'implique plus automatiquement une présentation read-only.
- Le hook frontend traite `not_ready` différemment selon l'adaptateur : EasyAgents conserve son échec actuel ; le Prompt normal répète un unique Open à la fois avec un délai fixe annulable jusqu'à `live`, `terminal`, `not_found` ou unmount.
- Le Prompt normal possède un état distinct de panne de replay transport. Un `transportError` ou une complétion sans vrai terminal y conduit ; cet état ne réutilise pas abusivement le handoff terminal EasyAgents.

- Pour le Prompt normal, le handoff terminal est direct uniquement après un vrai terminal reçu via `onData`, drainage de Queue A/Queue B, finalisation Prompt et complétion transport normale.
- Pour EasyAgents, le handoff terminal strict avec snapshot exact retourné par Open reste inchangé.
- Le Stop normal d'un `promptQuery` devient exact grâce au `sourceId` conservé dans l'état interne de `useAiStream`; il utilise une route source exacte. L'ancien endpoint thread-only reste réservé aux nettoyages destructifs de cycle de vie.
- Le Stop d'Activité ne possède pas de lease. Il valide synchroniquement : item exact Running/source/closed, panneau toujours fermé, source exacte active ou réservée, source non settled et aucun Stop déjà accepté ; puis il réclame l'abort exact et appelle Orch sans `await` intermédiaire.
- Le gestionnaire de son conserve l'option utilisateur par thread, mais son état de run actif et sa déduplication terminale utilisent le runRef exact. Les chunks/stops stale et les callbacks d'une ancienne source sont des no-op et ne créent jamais un nouvel état implicite.
- SidebarStore utilise une union stricte à trois branches : Idle/open ; Running/observer/open ; Running/source/open|closed. Un observer fermé et un item fermé/idle sont impossibles par typage et doivent être supprimés.
- Le bouton détaché est rendu uniquement pour `Running + runOwnership='source' + panelState='closed'`.
- Le champ `queryState.windows` et le type `SidebarActivityQueryWindowItem` sont renommés respectivement `queryState.items` et `SidebarActivityQueryItem`.
- La réhydratation Activité fusionne panneaux ouverts, records observer actifs et sources normales actives/réservées. Un observer actif n'est projeté que si son panneau est ouvert ; son elapsed peut repartir à zéro pendant au plus un tick avant que le runtime timer existant ne republie sa valeur exacte.
- Le registre `liveSession`, et non le SidebarStore, est l'autorité pour interdire la suppression d'un chat ou le Retry destructif EasyAgents pendant une source normale active/réservée. Le store reste une projection UX supplémentaire.
- `continueResponse` réutilise la bulle assistant et la bulle thinking agrégée du tour. Un `baseThinkingTextRef`, le `streamInstanceId` et le séparateur canonique inter-message garantissent que streaming continu, retry transport, replay et hydratation classique convergent vers une seule bulle thinking identique.
- Les règles « supprimer la bulle thinking vide au premier token/tool call » ne s'appliquent jamais à une bulle contenant du thinking historique de continuation.
- Aucun résultat généré n'est listé comme fichier à éditer. La génération normale produira les clients et contrats correspondants après modification des endpoints sources.

Vue d'ensemble de la méthode choisie

Identité, tentative de transport, réservations et machine d'état

`LiveSessionRunRef` devient `{ locator, turnId, sourceId }`. Le run key structurel inclut les trois valeurs. Le frontend maintient séparément un `streamInstanceId`, renouvelé à chaque `start()` accepté, qui sert de garde locale et de clé de tentative.

Le registre possède :

- des réservations exactes de sources normales ;
- des records enregistrés/replayables ;
- un index actif unique par locator ;
- des leases actives/retired ;
- une époque monotone par locator ;
- des métadonnées de retraite terminale et non terminale ;
- un état `abortRequested` atomique ;
- la date unique `sourceStartedAtMs` ;
- le `replayMode` ;
- uniquement pour la continuation, une baseline immuable de messages.

Une source normale suit :

1. acceptation du start frontend ;
2. génération de `sourceId` et `streamInstanceId` ;
3. message bus vers l'extension ;
4. réservation exacte ownership `source` comme première mutation synchrone, avec un unique `sourceStartedAtMs` ;
5. publication Activité Running/source avec l'état réel du panneau ;
6. armement du son exact pour ce run ;
7. persistance éventuelle de la préférence SPECIF Code ;
8. chargement du catalogue ;
9. capture sérialisée des messages de baseline `continueResponse` si nécessaire ;
10. retour réussi de `askStream(...)` ;

11. priming du `Readable` lazy ;
12. transformation réservation → record et application immédiate d'un abort précoce ;
13. append de chaque chunk canonique avant exposition ;
14. baseline durable ;
15. replay possible ;
16. vrai terminal ou synthèse canonique d'une interruption ;
17. fin de consommation backend, sans retirer encore le run de l'index actif ;
18. retour du transport Slimbk puis drainage complet de la pipeline FIFO de forwarding extension ;
19. finalisation atomique post-forwarding : annulation d'une réservation non enregistrée, ou `sourceSettled + handoffReady` si baseline et vrai terminal, sinon éviction non terminale ;
20. transition Activité Idle si panneau ouvert, ou suppression si panneau fermé ;
21. cleanup mémoire/attachments/son/trivial uniquement si aucun panneau ni nouvelle source n'existe.

Replay, erreurs et baselines

`LiveSessionReplayMode` expose :

- `replaceTurnAssistants` : le snapshot live retire les assistants du tour exact ; le frontend recrée think + assistant et rejoue tous les chunks de cette source ;
- `continueAssistant` : le registre conserve les messages exacts d'avant continuation. À chaque Open, le backend recharge l'enveloppe persistée actuelle puis substitue ces messages, de sorte que le brouillon et les attachments les plus récents restent présents.

Le premier `meta.error` converti en header Slimbk suit un protocole réversible :

- le service source append le chunk original au buffer ;
- l'endpoint encode son `AiErrorMetaValue` dans un header `500_SERVER_ERROR` versionné ;
- le handler extension préserve cet encodage ;
- le frontend le décode et injecte exactement le même chunk logique dans Queue A ;
- les erreurs sans cet encodage deviennent un `SYSTEM_ERROR` canonique ;
- un EOF sans terminal devient `INTERRUPTED`.

Le terminal bufferisé ne suffit pas à rendre le run handoff-ready : pour les sources normales, le handler extension doit d'abord drainer tous les callbacks `onChunk`, puis appeler la finalisation atomique du registre. Le replay peut donc continuer à attendre entre la fin backend et la fin du forwarding, sans fenêtre où le coordinateur croirait la source absente.

Cycle de vie du panneau sans cycle de modules

La classe coordinateur ne connaît qu'un port low-level injecté et des ports métier injectés. Le wiring du panneau instancie le coordinateur et expose une fonction publique coordonnée pour les ouvertures Prompt. Les autres pages continuent d'utiliser le low-level existant.

Les releases sont dispatchées ainsi :

- `easyAgentsBatch + stopOnLastViewerClose` → manager EasyAgents, qui conserve la policy dernier viewer = hard stop ;
- `normalPrompt + continueOnPanelClose` → release générique de la lease, sans abort de source ;
- descripteur invalide → aucun fallback silencieux et aucun cleanup destructif d'un run actif.

Machine d'état Activité :

État	Panneau	Ownership	Projection
Idle	open	aucun run	deux lignes, aucune action Stop
Running	open	observer	deux lignes, UX EasyAgents inchangée
Running	open	source	deux lignes, Stop normal dans le Prompt
Running	closed	source	trois lignes, Stop détaché dans Activité
terminal	closed	aucun run	aucun item

Frontend Prompt normal reconnecté

Pendant `liveSessionOpening`, le panneau peut afficher un bref état de reconnexion. Dès que le snapshot est installé et le replay armé :

- l'adaptateur normal retourne une surface non-viewer ;
- le compositeur et les attachments sont visibles ;
- la saisie et la sauvegarde du draft sont actives ;
- `isStreaming` conserve le verrou Send et transforme le bouton normal en Stop ;
- aucune action Stop supplémentaire n'est rendue par le banner liveSession ;
- les interactions d'historique compatibles avec le stream sont réactivées ;
- l'ownership de cleanup reste différé jusqu'au vrai handoff terminal.

Chaque tentative de replay possède un `streamInstanceId`. Sur erreur transport, le hook recharge la baseline exacte, conserve les overrides locaux compatibles, puis démarre une nouvelle tentative avec un nouvel identifiant. Les callbacks de la tentative précédente ne peuvent plus toucher le runtime courant.

À la fin normale du replay, la policy normale passe directement le runtime à `normal`, conserve le draft/attachments locaux et réenregistre le thread pour les futures fermetures. EasyAgents conserve son Open terminal strict.

Activité, son et contrôles Stop

Le bouton détaché utilise `VscButton` avec :

- `icon="SquareStop"` ;
- `size="small"` ;
- `variant="danger"` ;
- label `Stop` ;
- `disableAfterClick` ;
- `enabled={!stopRequested}` ;
- un `testId` stable basé sur `sourceId`.

Le lien de titre reçoit également un `testId` stable basé sur le locator. Les handlers React par ligne sont portés par un sous-composant ou des callbacks mémorisés, sans hook appelé dans une boucle.

Le gestionnaire de son déduplique et garde les callbacks par runRef exact. Le handler source normal attend sa pipeline FIFO de chunks, finalise atomiquement le registre, puis seulement exécute `onAiStreamStop`, la transition Activité et le cleanup coordonné.

Liste exacte des fichiers avec détails d'implémentation

Contrats shared

1) `AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionReplay.type.ts` — créé

- Définir `ZLiveSessionReplayMode` et `LiveSessionReplayMode` avec `replaceTurnAssistants | continueAssistant`.
- Documenter précisément la reconstruction frontend de chaque branche.

2) `AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionRun.type.ts` — modifié

- Ajouter le schéma/type canonique `LiveSessionSourceId`, trim/non vide.
- Ajouter `sourceId` obligatoire à `ZLiveSessionRunRef`.
- Inclure `sourceId` dans `buildLiveSessionRunKey` sans modifier la clé locator.
- Documenter `turnId` logique/persistant et `sourceId` exécutionnel.

3) `AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionPanel.type.ts` — modifié

- Ajouter `LIVE_SESSION_SOURCE_ID_PARAM` et `LIVE_SESSION_ADAPTER_PARAM`.
- Définir l'adaptateur strict `easyAgentsBatch | normalPrompt`.
- Réutiliser le schéma canonique de sourceId.
- Conserver les close policies et rendre `continueOnPanelClose` effectif pour `normalPrompt`.

4) `AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionOpenResult.type.ts` — modifié

- Ajouter `sourceId` aux branches `live` et `terminal`.
- Ajouter `replayMode` uniquement à `live`.
- Étendre le transport Slimbk et `superRefine` pour exiger/interdire chaque champ selon la branche, y compris les propriétés explicitement présentes à `undefined`.

5) `AICode/src/sharedlib/src/types/liveSessionTypes/__test__/LiveSessionContracts.test.ts` — modifié

- Couvrir clé locator+turn+source, collisions, query params, adaptateurs, replay modes et transports.
- Vérifier que deux sources du même turn ont des clés différentes et qu'une absence de sourceId est refusée.

6) `AICode/src/sharedlib/src/types/aiTypes/aiStopSuffix.util.ts` — modifié

- Ajouter une fabrique canonique de `meta.error` pour erreurs transport/source.
- Ajouter un encodeur/décodeur privé versionné du `AiErrorMetaValue` utilisé dans le header Slimbk du premier `meta.error`.
- Le décodeur doit rechercher et valider strictement le payload même lorsqu'il est enveloppé dans une stack `SlimbkCallError` transportée comme chaîne.
- Produire `SYSTEM_ERROR` ou `INTERRUPTED` avec les messages canoniques existants pour les erreurs non encodées.
- Garantir que backend et frontend construisent le même chunk à partir du même payload encodé.

7) `AICode/src/sharedlib/src/types/aiTypes/__test__/aiStopSuffix.util.test.ts` — modifié

- Tester Error/string/status Slimbk, encodage/décodage réversible, préfixes transport, détails multilignes, payload invalide, SYSTEM_ERROR, INTERRUPTED et stabilité du chunk produit.

8) `AICode/src/sharedlib/src/types/busTypes/ui/SidebarStore.ts` — modifié

- Remplacer `SidebarActivityQueryWindowItem` par l'union stricte `SidebarActivityQueryItem`.
- Branches exactes :
- Idle : `status='idle'`, `panelState='open'`, aucune identité active ;
- Running observer : `status='running'`, `runOwnership='observer'`, `panelState='open'`, `activeTurnId`, `activeSourceId` ;
- Running source : `status='running'`, `runOwnership='source'`, `panelState='open'|'closed'`, `activeTurnId`, `activeSourceId`, `stopRequested`.
- Préserver titre, elapsed courant/dernier, `updatedAt` et `openedAt`.
- Renommer `queryState.windows` en `queryState.items`.

9) `AICode/src/sharedlib/src/types/busTypes/ui/SidebarUiMessages.ts` — modifié

- Ajouter `sidebar->ext:openQueryActivity` avec locator complet.
- Ajouter `sidebar->ext:stopDetachedQuery` avec `LiveSessionRunRef`, réponse stricte `{ accepted: boolean }`.

10) `AICode/src/sharedlib/src/types/busTypes/global/MessageRegistry.ts` — modifié

- Ajouter `sourceId` à `proxylocalapi:aiStream`.
- Ajouter `sourceId` à `proxylocalapi:liveSessionStream`.
- Le `streamInstanceId` reste frontend-only et ne doit pas être ajouté au transport backend.

Backend Orch `liveSession`

11) `AICode/src/extension/src/aiTools/orch/liveSession/liveSession.programming-doc-readme.md` — modifié

- Documenter sourceId, streamInstanceId frontend, réservations, readiness, replay modes, baseline de messages continuation, merge d'enveloppe récente, encodage erreur, ownership source/observer, lifecycle partagé, panel open/closed, Activity, son exact, direct handoff, retraites et claims Stop.
- Documenter explicitement la barrière post-forwarding : une source Prompt normale reste active/réservée jusqu'au drainage de la pipeline extension, puis seulement devient settled/handoff-ready.
- Ajouter les machines d'état et les races de sources successives, retry transport, panel, forwarding et cleanup.

12) `AICode/src/extension/src/aiTools/orch/liveSession/registry/LiveSessionRecord.type.ts` — modifié

- Ajouter `baselineOnly` à `LiveSessionReadiness`.
- Ajouter `replayMode`, `continueBaselineMessages`, `sourceStartedAtMs`, `abortRequested` et `viewerGroupId?`.
- Définir la réservation source, le snapshot public actif/réservé, le snapshot actif générique avec ownership, et la retraite terminale/non terminale avec époque.
- Stocker sur un record handoff-ready l'époque terminale capturée au handoff.
- Étendre permits/subscriptions à l'identité sourceId.
- Interdire par types/validations une baseline continuation hors `continueAssistant`.

13) `AICode/src/extension/src/aiTools/orch/liveSession/registry/LiveSessionRegistry.service.ts` — modifié

- Ajouter réservation/cancel exacts pour ownership source ; la réservation est synchrone et unique par locator.
- Exiger une réservation correspondante pour enregistrer une source normale ; conserver l'enregistrement direct EasyAgents.
- Transformer atomiquement réservation → record et retourner `abortRequested`.
- Étendre readiness et baseline de messages avec validations de replayMode.
- Ajouter des snapshots immuables `getActiveRunForLocator`, `listActiveRuns`, `getActiveSourceRun`, `listActiveSourceRuns`, couvrant les réservations pour les variantes source.
- Ajouter `hasActiveOrReservedSource` exact et `hasActiveOrReservedSourceForLocator`.
- Ajouter les claims atomiques/idempotents viewer et source ; tous partagent `abortRequested`.
- Remplacer `canAbortSource` par ces claims et supprimer l'ancienne API morte.
- Ajouter une finalisation normale atomique post-forwarding : si l'identité exacte est encore une réservation, l'annuler sans fallback ; si elle est un record, marquer `sourceSettled`, évincer sans terminal si aucune baseline n'existe, sinon exiger `hasTerminalMeta` puis publier `handoffReady` et l'époque terminale dans la même mutation logique.
- Cette finalisation doit être idempotente pour le même run et refuser toute mutation d'une source remplacée ou d'un ownership incompatible.

- Remplacer le set de retired keys par une retraite structurée.
- Une éviction TTL terminale n’est éligible que si l’époque courante correspond encore à l’époque terminale du record ; sinon elle est stale/non autorisée.
- Faire évoluer l’époque sur reserve/register/cancel/handoff et sur les évictions qui invalident réellement le locator, sans permettre à une éviction tardive d’ancien record de rafraîchir son autorité terminale.
- Préserver toutes les garanties de leases, retention et replay de tranche 1.

14) `AICode/src/extension/src/aiTools/orch/liveSession/registry/__test__/LiveSessionRegistry.service.test.ts` — modifié

- Couvrir sources successives même turn/sourceId distinct, reservation→record, cancel, conflict, baselineOnly, baseline messages immuable, abort pré/post-registration, listing générique/source, claims concurrents, retraites et époques.
- Couvrir la finalisation post-forwarding : run toujours actif avant l’appel, réservation annulée, record sans baseline évincé, record avec baseline sans terminal refusé, record terminal handoff-ready, idempotence et source remplacée.
- Ajouter le cas critique : A terminal retenu, B démarre, puis A est évincé tardivement ; A ne redevient jamais fallback-éligible.
- Rejouer toutes les garanties tranche 1 avec sourceId explicite.

15) `AICode/src/extension/src/aiTools/orch/liveSession/core/LiveSessionService.ts` — modifié

- Exposer reserve/cancel, résolution/listing générique et source, tests d’existence exact/locator, baseline messages, claims abort et finalisation normale atomique post-forwarding.
- Supprimer `canAbortSource`.
- Conserver Open/Replay/release/sweep/evict.

16) `AICode/src/extension/src/aiTools/orch/liveSession/core/__test__/LiveSessionService.test.ts` — modifié

- Couvrir parcours normal complet, réservations, réouverture avant/après baseline, sources successives, listings, claims, finalisation post-forwarding sans baseline, avec terminal, sans terminal et retrait terminal stale.

17) `AICode/src/extension/src/aiTools/orch/liveSession/open/LiveSessionOpen.service.ts` — modifié

- Retourner `not_ready` immédiatement pour une réservation exacte.
- Pour `replaceTurnAssistants`, conserver le chargement de l’enveloppe actuelle et le filtrage exact des assistants du tour.
- Pour `continueAssistant`, charger l’enveloppe persistée la plus récente puis remplacer seulement `messages` par la copie de `continueBaselineMessages` ; ne jamais réutiliser un brouillon figé au démarrage.
- Retourner sourceId et replayMode.
- Ne lire un fallback terminal absent-record que si une retraite terminale exacte et son époque l’autorisent.

18) `AICode/src/extension/src/aiTools/orch/liveSession/open/__test__/LiveSessionOpen.service.test.ts` — modifié

- Couvrir réservation, baselineOnly, baseline continuation immuable, overlay du brouillon/attachments/nextTurnId/titre les plus récents, source mismatch, deux sources du même turn, fallbacks avant/après nouvelle source, éviction tardive d’ancien terminal et retrait non terminal.
- Vérifier qu’après terminal bufferisé mais avant finalisation post-forwarding, Open reste `live`/attachable et ne bascule pas prématurément vers le fallback terminal.

19) `AICode/src/extension/src/aiTools/orch/liveSession/replay/__test__/LiveSessionReplay.service.test.ts` — modifié

- Ajouter sourceId à toutes les fixtures.
- Vérifier qu’une subscription ancienne ne reçoit jamais les chunks d’une source ultérieure du même turn.
- Vérifier qu’un replay ayant drainé le terminal attend encore la finalisation post-forwarding avant de se fermer proprement.

20) `AICode/src/extension/src/aiTools/orch/liveSession/source/LiveSessionSourceLifecycle.service.ts` — créé

- Extraire le lifecycle générique du Readable lazy en service OOP.
- Garantir : createReadable throw sans settlement, premier `next()` armé, source-start sync, consommation du premier résultat puis tail, abort/destroy/return best-effort, observation de la promesse pendante, settlement exact et agrégation déterministe.
- Le « settlement exact » du lifecycle signifie ici la fin de consommation de la source et l’appel du callback produit ; pour `normalPrompt`, le callback ne publie pas encore `sourceSettled/handoffReady` dans le registre, cette publication restant la responsabilité post-forwarding du handler extension.
- Ne contenir aucune logique EasyAgents ni Prompt.

21) `AICode/src/extension/src/aiTools/orch/liveSession/source/__test__/LiveSessionSourceLifecycle.service.test.ts` **— créé**

- Migrer/généraliser les tests de priming et cleanup : source-start throw, première lecture throw, tail throw, consumer throw, abort, return/destroy, settlement throw et ordre des erreurs.
- Vérifier que le service n'impose aucune sémantique de registre : le callback de settlement fourni par l'adaptateur reste l'unique frontière produit.

22) `AICode/src/extension/src/aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.ts` **— créé**

- Mapper chaque AiQuery vers replayMode ; readiness reste `baselineOnly`.
- Pour `continueResponse`, capturer sous `ChatPersistence.queue` l'enveloppe actuelle et les messages défensifs `OrchMemory`, valider l'assistant ciblé, puis stocker uniquement la baseline de messages.
- Enregistrer ownership source après priming, transmettre `sourceStartedAtMs`, marquer immédiatement la baseline continuation et appliquer un abort déjà réclamé.
- Parser les chunks, append avant exposition, marquer fallback baseline et suivre le vrai terminal.
- Le premier `meta.error` doit être appendu sans transformation, puis exposé à l'endpoint avec son encodage header réversible.
- Transformer les throws/EOF sans terminal en erreur canonique bufferisée/exposée.
- Ne pas appeler la finalisation `sourceSettled/handoffReady` du registre depuis ce service. Retourner/terminer seulement après que la source est consommée et que tout terminal canonique requis a été appendu ; le handler extension publie la finalisation atomique après drainage du forwarding.
- Utiliser le lifecycle partagé et un flag local garantissant qu'un échec de registration ne finalise jamais un record tiers.

23) `AICode/src/extension/src/aiTools/orch/liveSession/source/__test__/NormalPromptLiveSessionSource.service.test.ts` **— créé**

- Couvrir prompt/retry/edit/specif/compact/continue, capture messages, overlay ultérieur via Open, ordre append-before-yield, même turn/source différente, abort réservé, premier meta.error/header encodé, throw après baseline et EOF sans terminal.
- Vérifier que la fin de consommation ne publie ni `sourceSettled` ni `handoffReady`, que le record reste actif jusqu'à la finalisation post-forwarding, et qu'un échec de registration ne touche aucun record tiers.

Adaptation EasyAgents

24) `AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/EasyAgentsBatchLiveSessionAdapter.service.ts` **— modifié**

- Enregistrer `replayMode='replaceTurnAssistants'`, `sourceId=turnId` et `sourceStartedAtMs` au démarrage exact.
- Injecter le lifecycle générique dans le wiring Batch sans modifier la policy produit.
- Supprimer l'adaptateur mort `canAbortSource`; les endpoints utilisent désormais les claims du cœur.
- EasyAgents conserve son ordre historique propre : son callback produit peut continuer à marquer `sourceSettled`, puis le runtime Batch publie `handoffReady` après convergence métier. La barrière post-forwarding ajoutée ici concerne uniquement ownership `source` normalPrompt.

25) `AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/__test__/EasyAgentsBatchLiveSessionAdapter.service.test.ts` **— modifié**

- Vérifier sourceId=turnId, replayMode, timestamp source et suppression de l'ancien seam `canAbortSource`.
- Vérifier que l'extraction du lifecycle ne change pas l'ordre settlement/handoff EasyAgents.

26) `AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunExecution.service.ts` **— modifié**

- Construire tous les run refs avec sourceId=liveTurnId.
- Conserver identité ligne/turn et transitions Batch inchangées.

27) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunExecution.live-activity-and-gating.service.test.ts` **— modifié**

- Étendre les assertions à sourceId/replayMode et préserver l'ordre Watchable/Handoff.

28) `AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunChildExecution.service.ts` **— modifié**

- Remplacer le lifecycle local par `LiveSessionSourceLifecycleService`.
- Conserver strictement parsing Batch, classification et callbacks métier.

29) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunChildExecution.service.test.ts` **— modifié**

- Conserver les tests d'intégration Batch ; déplacer les tests purement lifecycle vers le nouveau service.

30) `AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunManager.service.ts` **— modifié**

- Injecter le lifecycle partagé dans le child executor.
- Remplacer `queryState.windows` par `items`.
- Pour le guard de Retry manuel, consulter aussi `liveSession.hasActiveOrReservedSourceForLocator(locator)`.

31) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunExecution.testHarness.ts` **— modifié**

- Injecter le lifecycle partagé et fournir des run refs avec sourceId.

32) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunLifecycle.service.test.ts` **— modifié**

- Adapter la construction du child executor au lifecycle partagé.

33) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchLastViewerStop.service.test.ts` **— modifié**

- Ajouter sourceId aux fixtures et préserver les courses last-viewer/ancien-nouveau run.

34) `AICode/src/extension/src/aiTools/easyAgents/easyAgents.programming-doc-readme.md` **— modifié**

- Documenter sourceId=turnId, lifecycle partagé et branche Activity observer/open uniquement.
- Conserver zéro viewer = hard stop.

Endpoints et message bus

35) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/open.endpoint.ts` **— modifié**

- Exiger sourceId et transporter sourceId/replayMode.

36) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/open.endpoint.test.ts` **— modifié**

- Couvrir nouveaux champs, branches et source mismatch.

37) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/stream.endpoint.ts` **— modifié**

- Exiger sourceId et relayer le run exact complet.

38) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/stream.endpoint.test.ts` **— modifié**

- Couvrir sourceId exact, mismatch et replay terminal.
- Couvrir le cas où le terminal est déjà dans le buffer mais où le replay attend encore la finalisation post-forwarding.

39) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/abort.endpoint.ts` **— modifié**

- Exiger sourceId.
- Utiliser un claim viewer atomique accepté pour ownership observer ou source avec lease active.
- Claim et `orchAsk.stream.abortStream(...)` restent dans la même pile synchrone.

40) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/abort.endpoint.test.ts` **— modifié**

- Couvrir viewer EasyAgents, viewer Prompt normal, stale source, double Stop, source remplacée et claim idempotent.

41) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/abortSource.endpoint.ts` **— créé**

- Route exacte du Stop normal du panneau source.
- Recevoir locator+turnId+sourceId, réclamer l'abort source exact, puis appeler Orch sans await intermédiaire.

42) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/abortSource.endpoint.test.ts` **— créé**

- Couvrir réservation/enregistrement, stale source, double clic, source absente et ordre claim → abort.

43) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/stream.endpoint.ts` **— modifié**

- Ajouter sourceId au schéma.
- Conserver persistance SPECIF Code avant catalogue.
- Déléguer création/consommation/instrumentation à `NormalPromptLiveSessionSourceService`.
- Pour le premier `meta.error`, envoyer un header 500 dont `errorMessage` contient l'encodage réversible versionné ; ne jamais bufferiser une représentation différente.
- Pour le premier chunk normal, conserver le header 200 puis forwarding unique.
- Ne pas publier `sourceSettled` ou `handoffReady` à la fin de l'endpoint : l'endpoint termine après consommation/canonisation, la finalisation du registre est volontairement différée au handler message bus après drainage FIFO.

44) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/__test__/stream.endpoint.test.ts` **— modifié**

- Tester délégation, sourceId, ordre SPECIF, premier chunk normal, premier meta.error/header encodé, absence de double forwarding et erreurs pré-source.
- Vérifier explicitement que l'endpoint ne finalise pas prématurément le registre.

45) `AICode/src/extension/src/messageBus/handlers/ia/liveSessionStreamHandlers.ts` **— modifié**

- Relayer sourceId.
- Ownership source : aucune activité/son supplémentaire.
- Ownership observer : activité/son uniquement si le panneau exact est encore ouvert ; jamais observer/closed.
- Passer le runRef exact au gestionnaire de son.
- Préserver pipeline async, ordre des chunks et cleanups partiels.

46) `AICode/src/extension/src/messageBus/handlers/ia/__test__/liveSessionStreamHandlers.test.ts` **— modifié**

- Couvrir sourceId, ownership, panneau observer déjà fermé, son exact, absence de double activité/son et ordre async.

47) `AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts` **— modifié**

- Recevoir sourceId frontend et construire le runRef exact.
- Réserver synchroniquement avant tout await et partager le même `sourceStartedAtMs` avec Activité.
- Démarrer l'activité source avec panelState lu synchroniquement juste avant mutation, puis armer le son exact.
- Transmettre sourceId à l'endpoint.
- Maintenir une pipeline FIFO locale pour tous les callbacks `onChunk`, car le client généré ne les attend pas ; attendre cette pipeline avant toute finalisation ou cleanup.
- Préserver sans altération l'erreur encodée du premier `meta.error` jusqu'au frontend.
- Après le retour/échec du transport et le drainage FIFO, appeler la finalisation normale atomique du registre avant `onAiStreamStop` et avant la transition Activité. Cette opération doit annuler une réservation non transformée ou publier `sourceSettled + handoffReady` pour le record terminal exact.
- Si la finalisation du registre échoue, continuer tout de même les cleanups suivants, agréger l'erreur dans l'ordre et ne jamais exécuter un cleanup destructif hors du coordinateur.
- En cleanup complet : son stop exact, activité stop exacte, cancel réservation idempotent de sécurité, puis coordinateur `onSourceSettled(runRef)`.
- Exécuter tous les cleanups même si transport, pipeline, finalisation du registre ou cleanup précédent échoue ; agréger les erreurs dans l'ordre.

48) `AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts` **— modifié**

- Vérifier runRef/sourceId, pipeline drainée avant finalisation du registre et avant Stop son, fermeture panneau sans cleanup son prématuré, source suivante même turn et absence de son replay source-owned.
- Ajouter la course terminal backend déjà produit + callback `onChunk` encore pendant : le run reste actif, la fermeture détache sans cleanup son, puis la finalisation et le son terminal se produisent une seule fois après drainage.

49) `AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.runIdPropagation.test.ts` **— modifié**

- Étendre runRef à locator+turn+source, vérifier réservation avant await, Activity exacte, pipeline FIFO, finalisation registre post-pipeline, cancel idempotent et settlement coordonné.
- Vérifier l'ordre strict `last forwarded chunk → registry finalization → sound stop → Activity stop → coordinator settlement`.

50) `AICode/src/extension/src/extension/panels/prompt/PromptQueryDoneSound.manager.ts` **— modifié**

- Conserver l'option `enabled` par thread, mais remplacer `observedStreaming`/`lastPlayedTurnId` par un état corrélé au `LiveSessionRunRef` exact.
- `onAiStreamStart`, `onAiStreamChunk` et `onAiStreamStop` reçoivent le runRef exact.
- Un chunk ou stop stale est un no-op ; il ne doit jamais créer implicitement un nouvel état ni modifier la source courante.
- Dédupliquer le son par run key, pas seulement par turnId, afin qu'une continuation ou un Retry réutilisant le turn puisse jouer son propre son une seule fois.
- Préserver : aucun son sur `TOOL_USE`, aucun son sur `ABORTED`, fréquences succès/erreur actuelles.

51) `AICode/src/extension/src/extension/panels/prompt/__test__/PromptQueryDoneSound.manager.test.ts` **— modifié**

- Adapter toutes les fixtures au runRef exact.
- Couvrir mêmes turnId/sourceId différents, stale chunk, stale stop, cleanup pendant ancienne source et absence de création d'état sur callbacks tardifs.

Activité runtime extension

52) `AICode/src/extension/src/messageBus/handlers/ia/SidebarQueryRunningBridge.ts` — modifié

- Renommer la classe interne en `SidebarQueryActivityRuntimeService` sans déplacer le fichier.
- Stocker runRef exact, ownership, timer, startMs et runId par locator structurel.
- Guards onChunk/onStop par sourceId et runId.
- Source ownership accepte panel open/closed ; observer ownership exige panel open.
- Piloter sourceStarted/elapsed/sourceSettled de la nouvelle union.

53) `AICode/src/extension/src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.race.test.ts` — modifié

- Couvrir sources successives même turn, stale chunks/stops, observer fermé, panneau source fermé/rouvert.

54) `AICode/src/extension/src/messageBus/handlers/ia/__test__/SidebarQueryRunningBridge.usage.test.ts` — modifié

- Couvrir usage exact par source et transition Idle/suppression selon ownership/panelState.

Coordination panneau Prompt

55) `AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelQuery.util.ts` — créé

- Parser locator canonique et descripteur live complet : runRef, lease, adapter, close policy.
- Construire query normalPrompt exacte avec lease fraîche.
- Distinguer query classique, live explicite valide, live explicite invalide.
- Préserver ordre déterministe, retirer tous les paramètres `autoRun*` lors d'une injection normalPrompt et interdire toute dégradation silencieuse.

56) `AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionPanelQuery.util.test.ts` — créé

- Couvrir locator, sourceId, adaptateur, ordre, policies, paramètres partiels/invalides, suppression autoRun et query normalPrompt.

57) `AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelCoordinator.service.ts` — créé

- Classe OOP à dépendances injectées ; aucun import du low-level opener.
- Injecter explicitement : opener low-level, lecture panneau, liveSession active-run/source resolver, release normal, release EasyAgents, transitions Activity, cleanup thread, cleanup son, cleanup trivial et UI chat.
- Map locatorKey → `SerialTaskQueue`, supprimée à idle.
- `openPromptPanel` : révéler l'existant et corriger panelState ; sinon préserver live explicite ; sinon résoudre une source normale active/réservée et injecter query ; sinon ouvrir classique.
- `onPanelDisposed` : libérer la lease selon adapter, réévaluer tout run actif, appliquer panelClosed, détacher mapping exact et choisir cleanup différé ou historique.
- Une source normale active/réservée entraîne detach sans abort et conservation du son jusqu'au terminal.
- Un observer EasyAgents conserve sa policy release/kill ; le cleanup du panneau ne doit jamais inventer un item observer/closed ni aborter via le cleanup Prompt une source encore active.
- Associer le descriptor cleanup détaché au runRef exact actif à la fermeture.
- `onSourceSettled` : cet événement n'est appelé par le handler normal qu'après le drainage FIFO et la finalisation atomique du registre ; ignorer settlement stale si nouvelle source active ; si panneau ouvert, supprimer l'état détaché ; sinon cleanup terminal sans abort/await.
- Réévaluer panneau et active run immédiatement avant ClearMemory et avant cleanup son/trivial.

58) `AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionPanelCoordinator.service.test.ts` — créé

- Couvrir priorité live EasyAgents, release adapter-specific, normal auto-injecté, query invalide non dégradée, open/cleanup dans les deux ordres, close pré-baseline, observer actif, settlement stale après nouvelle source, cleanup gagné/perdu et aucune mémoire/son effacée sous un panneau ou une nouvelle source.
- Ajouter la course de drainage : terminal déjà bufferisé mais finalisation registre non encore publiée ; le coordinateur doit encore voir la source active, détacher le panneau et interdire tout cleanup destructif jusqu'à `onSourceSettled` post-forwarding.

59) `AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionDetachedStop.service.ts` — créé

- Valider synchroniquement panneau fermé, item Running/source exact, source active/réservée et non déjà arrêtée.
- Claim exact, mutation `stopRequested` sans await intermédiaire, puis `abortStream` dans la même pile.
- Retourner `{accepted}`.

60) `AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionDetachedStop.service.test.ts` **— créé**

- Couvrir accepted/refused, panneau rouvert, source remplacée, double clic, réservation et store stale.

61) `AICode/src/extension/src/extension/panels/prompt/PromptThreadCleanup.manager.ts` **— modifié**

- Remplacer map sessionId par map locator structurelle.
- Exposer opérations distinctes : register, detach source active sans abort, cleanup destructif historique, cleanup terminal settled sans abort/await, cleanup orphan partagé.
- Le cleanup terminal peut utiliser le sessionId du runRef comme threadId même si l'enregistrement frontend n'était pas encore arrivé.
- Ne jamais Abort/ClearMemory une source normale active lors du dispose.

62) `AICode/src/extension/src/extension/panels/prompt/__test__/PromptThreadCleanup.manager.attachmentsCleanup.test.ts` **— modifié**

- Couvrir collisions sessionId entre scopes, detach actif, cleanup idle, cleanup terminal sans abort, mapping absent terminal et sérialisation attachments.

63) `AICode/src/extension/src/extension/panels/core/openPanel.util.ts` **— modifié**

- Composer l'instance coordinateur avec ports injectés ; exposer fonction publique async coordonnée Prompt.
- Conserver low-level create/reveal pour autres pages.
- Le low-level Prompt capture descriptor initial pour dispose.
- Sur dispose : supprimer synchroniquement panneau registry puis déléguer tous effets Prompt au coordinateur ; garder seulement bookkeeping générique sélection/readyViews.
- Adapter préfixe tab Running à `queryState.items/status`.

64) `AICode/src/extension/src/extension/panels/core/__test__/openPanel.util.test.ts` **— modifié**

- Couvrir wiring sans cycle, sourceId/adaptateur, reveal, délégation dispose unique, unregister avant coordinator, release ports et pages non-Prompt inchangées.

65) `AICode/src/extension/src/messageBus/handlers/ui/uiGenericHandlers.ts` **— modifié**

- Pour `htmlPage='prompt'`, await façade coordonnée ; conserver low-level autres pages.

66) `AICode/src/extension/src/messageBus/handlers/ui/__test__/uiGenericHandlers.openExternalUrl.test.ts` **— modifié**

- Mocker façade Prompt et préserver tests URL.

67) `AICode/src/extension/src/extension/commands/NewChat.command.ts` **— modifié**

- Await ouverture coordonnée de la nouvelle session.

68) `AICode/src/extension/src/extension/commands/__test__/NewChat.command.test.ts` **— modifié**

- Vérifier appel coordonné async et propagation d'erreur.

69) `AICode/src/extension/src/extension/uriHandlers/registerDeepLinkUriHandler.ts` **— modifié**

- Passer deep links chat par façade coordonnée ; conserver non-bloquant avec catch explicite.

70) `AICode/src/extension/src/extension/uriHandlers/__test__/registerDeepLinkUriHandler.scope.test.ts` **— modifié**

- Vérifier ouverture coordonnée root/nested et guards scope.

SidebarStore et Activité

71) `AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.listeners.manager.ts` **— modifié**

- Remplacer add/running génériques par transitions atomiques `panelOpened`, `sourceStarted` et `sourceSettled`.
- La lecture du panelState et la mutation sourceStarted doivent rester sans await entre elles.
- Respecter trois branches strictes et conserver listeners title/running tabs Prompt.

72) `AICode/src/extension/src/extension/panels/sidebar/SidebarStore.queryWindow.mutations.manager.ts` **— modifié**

- Renommer windows → items.
- Ajouter panelClosed, forceRemove, live elapsed exact et stopRequested exact.
- `panelClosed` retire Idle et Running/observer ; conserve uniquement Running/source en `closed`.
- Conserver rename ephemeral.

73) `AICode/src/extension/src/extension/panels/sidebar/SidebarStore.core.manager.ts` **— modifié**

- Initialiser `queryState.items`.
- Appeler `rehydrateQueryActivities`.

74) `AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts` **— modifié**

- Renommer en `rehydrateQueryActivities`.
- Fusionner panneaux ouverts, records observer actifs et sources normales actives/réservées sans doublons.

- Ouvert+source : Running/source/open ; source seule : Running/source/closed ; ouvert+observer actif : Running/observer/open ; panneau seul : Idle/open.
- Ignorer observer sans panneau ; ne jamais projeter observer/closed ni closed/idle.
- Calculer elapsed source depuis `sourceStartedAtMs`; initialiser observer actif à zéro si nécessaire, son runtime timer exact le republiera au tick suivant.
- Un terminal déjà bufferisé mais non encore finalisé post-forwarding reste une source active et doit être réhydraté comme Running, jamais comme Idle/absent.

75) **`AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.rehydrate.manager.test.ts`** **— modifié**

- Couvrir idle ouvert, source ouverte, source fermée, réservation, observer actif ouvert, observer sans panneau ignoré et déduplication.
- Couvrir source backend consommée mais forwarding non drainé : elle reste Running jusqu'à la finalisation registre.

76) **`AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.manager.queryRunningUpsert.test.ts`** **— modifié**

- Tester toute machine d'état et identités source exactes.

77) **`AICode/src/extension/src/extension/panels/sidebar/SidebarStore.chatTitles.manager.ts`** **— modifié**

- Lire `items` et conserver fallback title source fermée/running.

78) **`AICode/src/extension/src/extension/panels/sidebar/SidebarStore.handlers.manager.ts`** **— modifié**

- Enregistrer openQueryActivity et stopDetachedQuery.
- Déléguer au coordinateur et service Stop.

79) **`AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.handlers.manager.test.ts`** **— créé**

- Vérifier payload locator/runRef, réponses accepted et délégation.

80) **`AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts`** **— modifié**

- Lire `queryState.items`.
- Bloquer si store projette Running ou si liveSession signale source normale active/réservée.
- Ne fermer/force-remove qu'après validation autoritative.

81) **`AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts`** **— créé**

- Couvrir source ouverte, fermée, réservation, store stale, source absente et absence side effects si bloqué.
- Couvrir terminal bufferisé mais forwarding non drainé : le registre reste autoritaire et bloque encore la suppression.

82) **`AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/list.endpoint.windowsLock.eperm.test.ts`** **— modifié**

- Adapter fixtures `items`.

83) **`AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/diskAliases.lastWins.endpoints.test.ts`** **— modifié**

- Adapter fixtures `items`, conserver scénarios delete disque.

84) **`AICode/src/extension/src/aiTools/orch/context/persistence/__test__/ChatPersistenceWriter.windowsFallback.test.ts`** **— modifié**

- Adapter fixture SidebarStore à `items`.

85) **`AICode/src/extension/src/aiTools/orch/context/memory/core/mem/__test__/OrchMemory.naive_title.user_prompt_specif_multiline.e2e.repro.test.ts`** **— modifié**

- Adapter fixture SidebarStore à `items`.

86) **`AICode/src/frontend/src/app/sidebar/sidebar-activity/SidebarActivityQuery.tsx`** **— modifié**

- Utiliser `items` et union stricte.
- Ouvrir via `sidebar->ext:openQueryActivity` avec locator.
- Ajouter Stop par runRef exact.
- Rendre troisième ligne seulement Running/source/closed.
- Utiliser SquareStop/small/danger/disableAfterClick, `enabled={!stopRequested}`, testId sourceId.
- Ajouter testId stable lien ouverture.
- Extraire si nécessaire un sous-composant de ligne afin que callbacks soient `useCallback` et objets `useMemo`, sans hooks dans `.map()`.

87) `AICode/src/frontend/src/app/sidebar/sidebar-activity/__test__/SidebarActivityQuery.test.tsx` **— modifié**

- Tester root/nested via testIds, observer/source, ligne Stop, disparition open/terminal, payload exact, disable immédiat et stopRequested.
- Ne pas sélectionner par labels visibles et ne pas importer `expect` depuis `@jest/globals`.

Frontend `liveSession` et Prompt

88) `AICode/src/frontend/src/app/prompt/liveSession/adapters/NormalPromptLiveSession.adapter.ts` **— créé**

- Parser adapter normal, sourceId, lease, continueOnPanelClose.
- Policies : retry not_ready, direct handoff, replay recovery, surface interactive streaming.
- Streaming : non-viewer, aucune action Stop banner.
- Failure replay transport : surface récupération explicite, sans faux normal.

89) `AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/NormalPromptLiveSession.adapter.test.ts` **— créé**

- Couvrir parsing, sourceId, policies et view models opening/streaming/replayFailure/normal.

90) `AICode/src/frontend/src/app/prompt/liveSession/adapters/PromptLiveSessionBootstrap.adapter.ts` **— créé**

- Dispatcher strict par `liveSessionAdapter` vers EasyAgents ou normalPrompt.
- Rejeter absent/inconnu sans fallback.

91) `AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/PromptLiveSessionBootstrap.adapter.test.ts` **— créé**

- Couvrir dispatch, configurations partielles et zéro fallback.

92) `AICode/src/frontend/src/app/prompt/liveSession/core/PromptLiveSession.types.ts` **— modifié**

- Ajouter policies not-ready, handoff, transport failure et surface interactive.
- Ajouter un mode distinct `liveSessionReplayFailed` pour normalPrompt ; conserver `liveSessionHandoffFailed` pour le handoff snapshot EasyAgents.
- Supprimer inférence mode!=normal⇒viewer et helper mort.
- Étendre bootstrap/runRef à sourceId.

93) `AICode/src/frontend/src/app/prompt/liveSession/core/usePromptLiveSession.hook.ts` **— modifié**

- Transporter/valider sourceId et replayMode.
- Retry not_ready normalPrompt : un Open à la fois, délai fixe non busy-loop, timer annulé.
- Démarrer replay selon replayMode.
- Normal direct handoff uniquement après terminal data + queues drainées + finalisation Prompt + outcome transport `completed` de la tentative courante.
- `transportError` ou `completed` sans vrai terminal → `liveSessionReplayFailed` et récupération exacte depuis Open/baseline.
- Chaque retry réhydrate la baseline puis appelle `startLiveObserver`, qui crée un nouveau streamInstanceId même si sourceId/lease restent identiques.
- EasyAgents conserve Open terminal/hydratation/Retry.

94) `AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.hydration.test.tsx` **— modifié**

- Ajouter sourceId/replayMode et conserver scénarios EasyAgents.

95) `AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.handoff.test.tsx` **— modifié**

- Vérifier EasyAgents terminal Open, normal direct, refus sur transportError et refus sur completed sans terminal.

96) `AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPrompt.test.tsx` **— créé**

- Couvrir not-ready retry, surface interactive, draft/attachments, Stop normal, replay failure/retry même source avec nouvel streamInstanceId, direct handoff, bootstrap terminal et close/reopen.

97) `AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx` **— créé**

- Comparer stream continu, baseline+replay et hydratation finale.
- Couvrir prompt, retry, edit, specif, compact, continue, thinking continuation, tool calls, gates, usage, file edits, erreurs canoniques et EOF interrompu.

98) `AICode/src/frontend/src/app/prompt/liveSession/adapters/EasyAgentsBatchLiveSession.adapter.ts` **— modifié**

- Émettre sourceId=turnId et adapter easyAgentsBatch.

- Parser identité sans changer labels/UX.
  99) `AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/EasyAgentsBatchLiveSession.adapter.test.ts` **— modifié**
- Adapter ordre query, sourceId et dispatcher.
  100) `AICode/src/frontend/src/messageBus/AiStreamStartArgs.type.ts` **— modifié**
- Garder argument public promptQuery sans sourceId.
- Définir état actif interne strict transportant runRef complet et `streamInstanceId`.
- La branche observer reçoit runRef exact.
- Définir outcome transport corrélé à la tentative courante : `pending | completed | transportError`.
  101) `AICode/src/frontend/src/messageBus/useAiStream.hook.ts` **— modifié**
- Après guard d'acceptation, générer sourceId pour promptQuery et un streamInstanceId pour chaque start, y compris observer retry.
- Conserver refs synchroniques currentSourceId/currentStreamInstanceId.
- Construire callbacks par tentative capturant ces identités.
- Rejeter stale onData/onComplete/onError avant mutation, après tout await et avant reset locks/queues.
- Exposer active runRef, streamInstanceId et outcome transport courant au controller.
- Stop promptQuery via abortSource exact ; Stop observer via abort viewer exact.
- Utiliser fabrique canonique d'erreur transport et décodeur header.
- Ne modifier Queue A, Queue B, locks, sanitizers ni raw-live routing.
  102) `AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts` **— modifié**
- Ajouter sourceId, routes Stop exactes, outcome, streamInstanceId et rejet callbacks stale lors d'un retry de la même source.
  103) `AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.promptSourceIdentity.test.ts` **— créé**
- Vérifier sourceId unique par start prompt accepté, streamInstanceId unique par toute tentative, stabilité transport, abort exact, stale data/completion/error, même turn réutilisé et observer retry même source.
  104) `AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts` **— modifié**
- `startLiveObserver` reçoit runRef+lease+replayMode.
- Ajouter `baseThinkingTextRef` et transmettre streamInstanceId au thinking lifecycle.
- Exposer outcome transport et tentative active à Prompt/liveSession.
- Utiliser bootstrap replace ou continuation approprié.
- Reset run state réinitialise base assistant/thinking, separator et identité tentative.
  105) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx` **— modifié**
- Couvrir sourceId, replay modes, base thinking, streamInstanceId et outcome.

Continuation et thinking isomorphes

106) `AICode/src/frontend/src/app/prompt/Prompt/preparePromptContinuationStreaming.util.ts` **— créé**

- Trouver assistant non-thinking exact du turn ; fail-fast s'il manque.
- Capturer base assistant et base thinking.
- Réutiliser bulle thinking agrégée ou créer placeholder unique avant assistant.
- Réutiliser UID assistant hydraté.
- Armer séparateur inter-message seulement si thinking historique existe.
- Réinitialiser buffers, refs, notices, guards et `fileEditsUiFinalized`.
  107) `AICode/src/frontend/src/app/prompt/Prompt/__test__/preparePromptContinuationStreaming.util.test.ts` **— créé**
- Couvrir assistant exact, base text, thinking existant/absent, separator, assistant manquant, sources successives et aucune duplication.
  108) `AICode/src/frontend/src/app/prompt/Prompt/preparePromptStreamingTurn.util.ts` **— modifié**
- Réinitialiser explicitement base thinking et pending separator pour replaceTurnAssistants.
- Conserver contrat nouveaux turns.
  109) `AICode/src/frontend/src/app/prompt/Prompt/__test__/preparePromptStreamingTurn.util.test.ts` **— modifié**
- Vérifier reset bases assistant/thinking et separator.
  110) `AICode/src/frontend/src/app/prompt/Prompt/usePromptThinkingLifecycle.hook.ts` **— modifié**
- Recevoir `baseThinkingTextRef` et `streamInstanceId`.
- Utiliser une initialisation avant effets de transport, par exemple `useLayoutEffect`, pour remettre transcript/indices à la base exacte à chaque tentative, même si sourceId et UID thinking restent identiques.

- Ajouter separator inter-message une seule fois avant premier nouveau thinking.
- Ne pas supprimer bulle contenant thinking historique lorsque segment commence texte/tool.
- Réinitialiser proprement pour replace.
  111) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptThinkingLifecycle.continuation.test.tsx` **— créé**
- Couvrir base thinking sans nouveaux tokens, append separator, tool/text avant thinking, réinsertion, reset nouveau turn et retry transport même source avec nouvel streamInstanceId.
  112) `AICode/src/frontend/src/app/prompt/Prompt/usePromptToolUse.hook.ts` **— modifié**
- Supprimer bulle vide uniquement lorsque `baseThinkingTextRef` est vide.
- Préserver thinking historique pendant tool calls continuation.
  113) `AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.finalization.hook.ts` **— modifié**
- Réinitialiser base thinking et pending separator après finalisation terminale, y compris meta-only.
- Conserver ordre finalisation/métadonnées.
  114) `AICode/src/frontend/src/app/prompt/Prompt/usePromptRunCommands.hook.ts` **— modifié**
- Remplacer bootstrap inline continueResponse par helper partagé.
- Conserver Retry/Edit/Send/Specif hors nouveaux refs/guards.
  115) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.continueResponse.test.tsx` **— créé**
- Vérifier helper, même turn logique, nouvelle sourceId, nouveau streamInstanceId, thinking unique et assistant cible exact.

Intégration Prompt et documentation

116) `AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx` **— modifié**

- Capturer sourceId/adaptateur dans query figée et utiliser dispatcher.
- Passer outcome transport, runRef et streamInstanceId au hook liveSession.
- Déduire interactionsEnabled de surface/policy, pas du mode seul.
- Autoriser draft persistence et interactions normales pendant replay normalPrompt.
- Garder ownsThreadForCleanup=false jusqu'au direct handoff ; ensuite réenregistrer thread.
- Préserver classic hydration off pour tout live bootstrap.
  117) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.liveSessionBootstrap.test.tsx` **— modifié**
- Couvrir dispatch, sourceId, surface interactive, draft, query figée, hydration off, replay failure et ownership cleanup.
  118) `AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx` **— modifié**
- Conserver viewer EasyAgents et ajouter normalPrompt streaming : composer visible, attachments autorisés, Stop normal, aucun banner Stop.
- Utiliser testIds/roles stables.
  119) `AICode/src/frontend/src/app/easy-agents/easy-agents-batch/core/__test__/EasyAgentsBatch.liveWatch.test.tsx` **— modifié**
- Adapter query sourceId/adaptateur sans changement UX.
  120) `AICode/src/frontend/src/app/easy-agents/easy-agents-item/core/batch/__test__/EasyAgentBatchSurface.test.tsx` **— modifié**
- Adapter query live exacte sans changement UX.
  121) `AICode/src/frontend/src/app/prompt/hydration/isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md` **— modifié**
- Ajouter fermeture/réouverture comme troisième chemin isomorphe.
- Documenter continuation messages immuables + enveloppe récente, assistant+thinking, retry transport, erreurs canoniques et vrai terminal requis.
- Documenter que la fin backend n'est pas encore la fin observable : la source normale reste active jusqu'au drainage FIFO et à la finalisation registre post-forwarding.
  122) `AICode/src/frontend/src/app/prompt/thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md` **— modifié**
- Documenter suppression limitée au placeholder vide du segment courant.
- Documenter bulle agrégée unique, separator inter-message et reset par streamInstanceId.

Récapitulatif exact des chemins de fichiers

Créés

- `AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionReplay.type.ts`
- `AICode/src/extension/src/aiTools/orch/liveSession/source/LiveSessionSourceLifecycle.service.ts`

- AICode/src/extension/src/aiTools/orch/liveSession/source/__test__/LiveSessionSourceLifecycle.service.test.ts
- AICode/src/extension/src/aiTools/orch/liveSession/source/NormalPromptLiveSessionSource.service.ts
- AICode/src/extension/src/aiTools/orch/liveSession/source/__test__/NormalPromptLiveSessionSource.service.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/abortSource.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/abortSource.endpoint.test.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelQuery.util.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionPanelQuery.util.test.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionPanelCoordinator.service.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionPanelCoordinator.service.test.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/PromptLiveSessionDetachedStop.service.ts
- AICode/src/extension/src/extension/panels/prompt/liveSession/__test__/PromptLiveSessionDetachedStop.service.test.ts
- AICode/src/extension/src/extension/panels/sidebar/__test__/SidebarStore.handlers.manager.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts
- AICode/src/frontend/src/app/prompt/liveSession/adapters/NormalPromptLiveSession.adapter.ts
- AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/NormalPromptLiveSession.adapter.test.ts
- AICode/src/frontend/src/app/prompt/liveSession/adapters/PromptLiveSessionBootstrap.adapter.ts
- AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/PromptLiveSessionBootstrap.adapter.test.ts
- AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPrompt.test.tsx
- AICode/src/frontend/src/app/prompt/liveSession/core/__test__/usePromptLiveSession.normalPromptIsomorphism.test.tsx
- AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.promptSourceIdentity.test.ts
- AICode/src/frontend/src/app/prompt/Prompt/preparePromptContinuationStreaming.util.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/preparePromptContinuationStreaming.util.test.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptThinkingLifecycle.continuation.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.continueResponse.test.tsx

Modifiés

- AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionRun.type.ts
- AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionPanel.type.ts
- AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionOpenResult.type.ts
- AICode/src/sharedlib/src/types/liveSessionTypes/__test__/LiveSessionContracts.test.ts
- AICode/src/sharedlib/src/types/aiTypes/aiStopSuffix.util.ts
- AICode/src/sharedlib/src/types/aiTypes/__test__/aiStopSuffix.util.test.ts
- AICode/src/sharedlib/src/types/busTypes/ui/SidebarStore.ts
- AICode/src/sharedlib/src/types/busTypes/ui/SidebarUiMessages.ts

- AICode/src/sharedlib/src/types/busTypes/global/MessageRegistry.ts
- AICode/src/extension/src/aiTools/orch/liveSession/liveSession.programming-doc-readme.md
- AICode/src/extension/src/aiTools/orch/liveSession/registry/LiveSessionRecord.type.ts
- AICode/src/extension/src/aiTools/orch/liveSession/registry/LiveSessionRegistry.service.ts
- AICode/src/extension/src/aiTools/orch/liveSession/registry/__test__/LiveSessionRegistry.service.test.ts
- AICode/src/extension/src/aiTools/orch/liveSession/core/LiveSessionService.ts
- AICode/src/extension/src/aiTools/orch/liveSession/core/__test__/LiveSessionService.test.ts
- AICode/src/extension/src/aiTools/orch/liveSession/open/LiveSessionOpen.service.ts
- AICode/src/extension/src/aiTools/orch/liveSession/open/__test__/LiveSessionOpen.service.test.ts
- AICode/src/extension/src/aiTools/orch/liveSession/replay/__test__/LiveSessionReplay.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/EasyAgentsBatchLiveSessionAdapter.service.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/liveSession/__test__/EasyAgentsBatchLiveSessionAdapter.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunExecution.service.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunExecution.live-activity-and-gating.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunChildExecution.service.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunChildExecution.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunManager.service.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunExecution.testHarness.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunLifecycle.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchLastViewerStop.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/easyAgents.programming-doc-readme.md
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/open.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/open.endpoint.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/stream.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/stream.endpoint.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/abort.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/__test__/abort.endpoint.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/stream.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/query/ask/__test__/stream.endpoint.test.ts
- AICode/src/extension/src/messageBus/handlers/ia/liveSessionStreamHandlers.ts
- AICode/src/extension/src/messageBus/handlers/ia/__test__/liveSessionStreamHandlers.test.ts
- AICode/src/extension/src/messageBus/handlers/ia/iaStreamHandlers.ts
- AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.queryDoneSound.test.ts
- AICode/src/extension/src/messageBus/handlers/ia/__test__/iaStreamHandlers.runIdPropagation.test.ts

- AICode/src/extension/src/extension/panels/prompt/PromptQueryDoneSound.manager.ts
- AICode/src/extension/src/extension/panels/prompt/__test__/
  PromptQueryDoneSound.manager.test.ts
- AICode/src/extension/src/messageBus/handlers/ia/SidebarQueryRunningBridge.ts
- AICode/src/extension/src/messageBus/handlers/ia/__test__/
  SidebarQueryRunningBridge.race.test.ts
- AICode/src/extension/src/messageBus/handlers/ia/__test__/
  SidebarQueryRunningBridge.usage.test.ts
- AICode/src/extension/src/extension/panels/prompt/PromptThreadCleanup.manager.ts
- AICode/src/extension/src/extension/panels/prompt/__test__/
  PromptThreadCleanup.manager.attachmentsCleanup.test.ts
- AICode/src/extension/src/extension/panels/core/openPanel.util.ts
- AICode/src/extension/src/extension/panels/core/__test__/openPanel.util.test.ts
- AICode/src/extension/src/messageBus/handlers/ui/uiGenericHandlers.ts
- AICode/src/extension/src/messageBus/handlers/ui/__test__/
  uiGenericHandlers.openExternalUrl.test.ts
- AICode/src/extension/src/extension/commands/NewChat.command.ts
- AICode/src/extension/src/extension/commands/__test__/NewChat.command.test.ts
- AICode/src/extension/src/extension/uriHandlers/registerDeepLinkUriHandler.ts
- AICode/src/extension/src/extension/uriHandlers/__test__/
  registerDeepLinkUriHandler.scope.test.ts
- AICode/src/extension/src/extension/panels/sidebar/
  SidebarStore.queryWindow.listeners.manager.ts
- AICode/src/extension/src/extension/panels/sidebar/
  SidebarStore.queryWindow.mutations.manager.ts
- AICode/src/extension/src/extension/panels/sidebar/SidebarStore.core.manager.ts
- AICode/src/extension/src/extension/panels/sidebar/SidebarStore.rehydrate.manager.ts
- AICode/src/extension/src/extension/panels/sidebar/__test__/
  SidebarStore.rehydrate.manager.test.ts
- AICode/src/extension/src/extension/panels/sidebar/__test__/
  SidebarStore.manager.queryRunningUpsert.test.ts
- AICode/src/extension/src/extension/panels/sidebar/SidebarStore.chatTitles.manager.ts
- AICode/src/extension/src/extension/panels/sidebar/SidebarStore.handlers.manager.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/delete.endpoint.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/
  list.endpoint.windowsLock.eperm.test.ts
- AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/
  diskAliases.lastWins.endpoints.test.ts
- AICode/src/extension/src/aiTools/orch/context/persistence/__test__/
  ChatPersistenceWriter.windowsFallback.test.ts
- AICode/src/extension/src/aiTools/orch/context/memory/core/mem/__test__/
  OrchMemory.naive_title.user_prompt_specif_multiline.e2e.repro.test.ts
- AICode/src/frontend/src/app/sidebar/sidebar-activity/SidebarActivityQuery.tsx
- AICode/src/frontend/src/app/sidebar/sidebar-activity/__test__/
  SidebarActivityQuery.test.tsx
- AICode/src/frontend/src/app/prompt/liveSession/core/PromptLiveSession.types.ts
- AICode/src/frontend/src/app/prompt/liveSession/core/usePromptLiveSession.hook.ts
- AICode/src/frontend/src/app/prompt/liveSession/core/__test__/
  usePromptLiveSession.hydration.test.tsx
- AICode/src/frontend/src/app/prompt/liveSession/core/__test__/
  usePromptLiveSession.handoff.test.tsx
- AICode/src/frontend/src/app/prompt/liveSession/adapters/
  EasyAgentsBatchLiveSession.adapter.ts
- AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/
  EasyAgentsBatchLiveSession.adapter.test.ts

- AICode/src/frontend/src/messageBus/AiStreamStartArgs.type.ts
- AICode/src/frontend/src/messageBus/useAiStream.hook.ts
- AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.liveSessionObserver.test.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/preparePromptStreamingTurn.util.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/preparePromptStreamingTurn.util.test.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptThinkingLifecycle.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptToolUse.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.finalization.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptRunCommands.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/Prompt.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.liveSessionBootstrap.test.tsx
- AICode/src/frontend/src/app/prompt/PromptInput/__test__/PromptInput.batchLiveViewerMode.test.tsx
- AICode/src/frontend/src/app/easy-agents/easy-agents-batch/core/__test__/EasyAgentsBatch.liveWatch.test.tsx
- AICode/src/frontend/src/app/easy-agents/easy-agents-item/core/batch/__test__/EasyAgentBatchSurface.test.tsx
- AICode/src/frontend/src/app/prompt/hydration/isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md
- AICode/src/frontend/src/app/prompt/thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md

Déplacés
Aucun.
Supprimés
Aucun.
Facteur de risque de régression

1. **Risque calculé : 92 %.** Criticité 5, couplage 5, blast radius 5, incertitude 4, faible confiance 3. Base = 10×5 + 3×5 + 2×5 + 2×4 + 3×3 = 92 ; risque = round(92 × 5/5) = 92 %.
2. **Zones complexes :** sources successives partageant un turn, retry transport de la même source, réservations avant await, callbacks stale, lifecycle Readable lazy, pipeline onChunk non attendue par le client généré, barrière de finalisation post-forwarding, premier header d'erreur réversible, EOF sans terminal, baseline continuation + enveloppe récente, thinking agrégé, races panel/open/cleanup, Activity source/observer, son exact, direct handoff, conservation draft/attachments, claims Stop pré-enregistrement, suppression autoritative et retraites terminales évincées après une source plus récente.
3. **Justification :** cette tranche modifie le chemin normal de toutes les requêtes Prompt, le dispose historique, la projection Activité et la coordination du frontend streaming. Une erreur peut perdre/réordonner un chunk, accepter une complétion stale, publier le handoff avant que le dernier callback onChunk soit drainé, clear la mémoire ou le son d'une nouvelle source, interrompre le mauvais run, figer un ancien brouillon de continuation, afficher un Stop EasyAgents détaché, produire deux bulles thinking ou ressusciter un ancien terminal. Les identités source+tentative, réservations, époques, ownerships stricts, queues par locator, pipeline FIFO, finalisation registre post-forwarding, encodage réversible et tests à trois chemins sont donc obligatoires.

Résumé

- Étendre liveSession aux sources Prompt normales en une implémentation complète.
- Ajouter sourceId de bout en bout et streamInstanceId côté frontend pour distinguer les retries transport de la même source.
- Réserver synchroniquement puis enregistrer les sources normales, bufferiser avant forwarding et partager le lifecycle lazy avec EasyAgents.
- Attendre une pipeline FIFO explicite dans le handler extension avant toute finalisation ou cleanup.
- Garder le run normal actif pendant tout le forwarding extension, puis publier atomiquement sourceSettled + handoffReady seulement après drainage complet ; annuler une réservation non enregistrée ou évincer un record sans baseline.

- Capturer une baseline immuable de messages pour `continueResponse`, tout en rechargeant l’enveloppe persistée la plus récente à chaque Open.
- Réutiliser assistant et thinking historique sans duplication, avec reset exact par tentative.
- Encoder réversiblement le premier `meta.error` Slimbk afin que buffer et frontend continu reçoivent le même terminal.
- Ne plus tuer une source normale active à la fermeture ; différer cleanup mémoire/attachments/son.
- Sérialiser ouverture, dispose et cleanup terminal par locator sans cycle de modules et avec release adapter-specific.
- Modéliser Activité avec Idle, Running observer/open et Running source/open|closed.
- Réutiliser le Stop normal dans Prompt et ajouter uniquement le Stop détaché Running/source/closed.
- Corréler également le son de fin au runRef exact.
- Utiliser le registre liveSession comme autorité pour deletion/Retry destructifs.
- Préserver intégralement EasyAgents, Queue A, Queue B, locks et sanitizers.
- Tester unités, courses et isomorphisme stream continu / replay / hydratation classique, notamment la fenêtre terminal backend produit mais forwarding extension non encore drainé.

Nom du commit
`feat(live-session): keep prompt runs alive across panel close and reopen`
SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_013420
Copy
Refine
Code
Verify

Sévérité	Description du changement	Pourquoi
🟥 majeure	Déplacement de la finalisation normale `sourceSettled + handoffReady` depuis le service source backend vers le handler extension, **après drainage complet de la pipeline FIFO `onChunk`**.	Le client Slimbk n’attend pas les callbacks asynchrones. Une finalisation backend anticipée retirait le run de l’index actif alors que des chunks, le son et l’Activité pouvaient encore être en transit, autorisant un cleanup destructif prématuré lors d’une fermeture de panneau.
🟥 majeure	Ajout d’une finalisation atomique post-forwarding dans le registre : annulation d’une réservation restante, éviction non terminale sans baseline, ou publication simultanée du settlement et du handoff avec baseline et vrai terminal.	Garantit qu’une source normale reste autoritativement active pendant tout le forwarding et empêche les états intermédiaires incohérents ou les anciens runs de finaliser une source plus récente.
🟥 majeure	Ajout des courses de forwarding aux tests du registre, du service, d’Open, du replay, du handler IA, du coordinateur, de la réhydratation Activité et de la suppression de chat.	Verrouille le scénario critique « terminal backend déjà produit, dernier callback extension pas encore drainé, panneau fermé entre les deux ».
🟨 mineure	Clarification de la séparation entre le settlement du lifecycle générique et la finalisation du registre : EasyAgents conserve	Évite qu’une extraction trop

Sévérité	Description du changement	Pourquoi
	son ordre historique, tandis que `normalPrompt` utilise la barrière post-forwarding.	générique du lifecycle modifie involontairement la policy EasyAgents ou réintroduise la finalisation prématurée côté Prompt normal.
🟧 mineure	Actualisation du commit courant réinspecté vers `7a39caa251158f63a6a34d6ca3cf2a0e82993425`.	Le dépôt courant a avancé depuis la précédente refinement ; le diff avec la tranche 1 reste limité aux quatre fichiers d’instructions SPECIF et ne modifie pas le sous-système concerné.

</ASSISTANT>
SHOW ALLPage 1 / 1