Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=f9e3dfb82772495da07b9b048994cbf5-RS-Am%C3%A9liorer-session
<USER>
[Refine]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_014742
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *reading project*: liveSession.programming-doc-readme.md, easyAgents.programming-doc-readme.md, isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md, thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md, 🔎 h "Architecture liveSession"
🔧 ✅ *scanning git*: ♈ "Vérifier les commits", ♈ "Historique liveSession", ♈ "Diff du refactor live", ♈ "Vérifier commit tranche", ♈ "Vérifier commit inspecté"
🔧 ✅ *reading project*: 📁 g "Fichiers liveSession", 👉 "Chemins live principaux"
🔧 ✅ *scanning git*: ♈ "Comparer les tranches", ♈ "Comparer au HEAD"
🔧 ✅ *reading project*: 🔎 h "Contrats et panel cleanup", 🔎 h "Stream backend et erreurs", LiveSessionRun.type.ts [L1-L220], LiveSessionPanel.type.ts [L1-L220], LiveSessionOpenResult.type.ts [L1-L240], LiveSessionRecord.type.ts [L1-L280], slimbkBuildHeader.ts [L200-L430], stream.endpoint.ts [L1-L190], EasyAgentsBatchRunChildExecution.service.ts [L80-L560], openPanel.util.ts [L220-L740], PromptThreadCleanup.manager.ts [L1-L260], SidebarStore.ts [L1-L300], MessageRegistry.ts [L1-L240], AiStreamStartArgs.type.ts [L1-L180], 🔎 h "Bus streaming fermeture", useAiStream.hook.ts [L100-L750], Prompt.tsx [L1-L360], EasyAgentsBatchLiveSession.adapter.ts [L1-L240], PromptLiveSession.types.ts [L1-L240], SidebarActivityQuery.tsx [L1-L260], SidebarStore.queryWindow.listeners.manager.ts [L1-L340], SidebarStore.queryWindow.mutations.manager.ts [L1-L300], useStreamListener.hook.ts [L1-L120], frontMessageBus.util.ts [L300-L390], extTypedMessageBus.util.ts [L1-L390], extMessageBus.util.ts [L1-L180], liveSessionStreamHandlers.ts [L1-L260], LiveSessionOpen.service.ts [L1-L360], LiveSessionService.ts [L1-L220], EasyAgentsBatchLiveSessionAdapter.service.ts [L1-L320], 🔎 e "Appels openCentralPanel", 🔎 e "Messages openPanel", 🔎 e "Nettoyage Prompt", 🔎 e "Fenêtres queryState", 🔎 e "Type QueryWindowItem", 🔎 e "RunRef liveSession", OrchAsk.ts [L1-L260], 🔎 h "Mémoire continuation", 🔎 h "Types AiQuery", AiQuery.type.ts [L1-L300], OrchMemory.ts [L1-L320], OrchMessagesMemory.ts [L1-L300], preparePromptStreamingTurn.util.ts [L1-L260], usePromptRunCommands.hook.ts [L1-L420], OrchAskStream.ts [L1-L240], 🔎 e "Accès mémoire OrchAskStream", 🔎 e "Instance OrchMemory stream", OrchContextBuilder.ts [L1-L220], OrchMemoryPersistenceQueue.ts [L1-L340], OrchMessagesMemoryStore.ts [L1-L240], usePromptRunCommands.hook.ts [L350-L520], usePromptThinkingLifecycle.hook.ts [L1-L340], 🔎 e "Snapshot mémoire publique", 🔎 e "Queue mémoire publique", 🔎 e "ChatPersistence queue", OrchMemory.ts [L300-L784], ChatPersistence.service.ts [L1-L220], ChatPersistenceWriteQueue.service.ts [L1-L260], registerAllBusHandlers.ts [L1-L220], SidebarStore.handlers.manager.ts [L1-L320], SidebarUiMessages.ts [L1-L220], UiMessages.ts [L1-L180], 🔎 e "Enregistrement handlers sidebar", SidebarStore.startup.manager.ts [L1-L260], SidebarStore.core.manager.ts [L1-L240], SidebarStore.rehydrate.manager.ts [L1-L220], 🔎 e "Fixtures windows", 🔎 e "Accès point windows", 🔎 e "Type fenêtre activité", 🔎 e "Initialisation queryState", callLocalApi.generated.ts [L1-L90], slimbkBuildHeader.ts [L1-L200], 🔎 e "SlimbkCallError exports", aiStopSuffix.util.ts [L1-L180], SlimbkResult.type.ts [L1-L180], VscButton.tsx [L1-L300], 🔎 e "Bouton SquareStop", 🔎 e "disableAfterClick usage", VscButton.module.css [L1-L260], 🔎 e "Icône SquareStop", 🔎 e "Littéraux runRef", 🔎 e "Appels registerRun", 🔎 e "Appels build run key", 🔎 e "Queries live turn", 🔎 e "Payload live turnId", 🔎 e "Enregistrement cleanup thread", 🔎 e "Clear cleanup mapping", uiAiPromptHandlers.ts [L1-L240], AiPromptUiMessages.ts [L1-L220], PromptLiveSessionBanner.tsx [L1-L240], 🔎 e "Usage LiveSessionBanner", PromptInput.tsx [L1-L260], 🔎 e "isViewerMode Prompt", 🔎 e "isPromptLocked dérivation", usePromptRunUiDerived.hook.ts [L1-L300], 🔎 e "PromptInput isLocked", Prompt.tsx [L980-L1320], 🔎 e "Calcul finalIsLocked", Prompt.tsx [L820-L990], LiveSessionReplay.service.ts [L1-L180], LiveSessionRegistry.service.ts [L220-

L640], abort.endpoint.ts [L1-L100], PromptQueryDoneSound.manager.ts [L1-L320], delete.endpoint.ts [L1-L150]

specifying "SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_022043"

SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_022043

Copy

Refine

Code

Verify

Markdown formattedMarkdown source

Spécification — Finaliser `liveSession` pour les sessions Prompt normales

Contexte

La tranche 1 correspond au sous-système fonctionnel introduit au commit `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`. La réinspection a été reprise depuis zéro sur le dépôt courant au commit `0b9f52cc`. Le commit `ed3bd94687cc34f6603730689bfc775818881bae`, présent dans l'ascendance courante, porte le même arbre fonctionnel `liveSession` que `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`; le diff entre ces deux commits est vide. Les commits ultérieurs inspectés ne modifient aucun contrat ni chemin d'exécution visé ici. La base fonctionnelle `liveSession` reste donc celle de la tranche 1, mais la référence au dépôt courant doit être `0b9f52cc` et non l'ancien commit d'inspection `7a39caa251158f63a6a34d6ca3cf2a0e82993425`.

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 active 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 livré atomiquement afin qu'aucun contrat intermédiaire incomplet ne soit introduit.

Le problème utilisateur 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 impose les invariants suivants :

- l'Activité distingue les runs `source` des runs `observer`, sinon une fermeture EasyAgents créerait à tort un item détaché avec bouton Stop ;
- le coordinateur de panneau n'importe jamais le low-level opener qui le compose, afin d'éviter un cycle ESM ;
- une query liveSession EasyAgents explicitement fournie par le caller est préservée et n'est jamais remplacée par une ouverture classique ;
- `sourceId` identifie la source backend exacte, tandis qu'un `streamInstanceId` frontend-only distingue chaque tentative de transport acceptée, y compris un retry de replay de la même source ;

- les callbacks frontend `onData`, `onComplete` et `onError`, y compris leurs continuations après `await`, sont gardés par `sourceId + streamInstanceId` ;
- le handoff direct normal exige un vrai terminal reçu comme donnée et une complétion transport normale ; une vraie erreur source transportée dans le premier header Slimbk doit donc être reconstruite par le handler extension, injectée dans sa même pipeline FIFO comme chunk terminal, puis considérée comme une fin transport traitée normalement ; elle ne doit pas remonter au frontend comme une erreur de transport ;
- le protocole privé du premier header d'erreur utilise exactement `errorCode='AICODE_LIVE_SESSION_SOURCE_ERROR_V1'` et un `errorMessage` commençant par `AICODE_LIVE_SESSION_SOURCE_ERROR_V1:` suivi du JSON strict `{ "version": 1, "value": AiErrorMetaValue }`; ce format est réversible sans dépendre de `Error.stack`, tandis que le décodeur peut aussi extraire le préfixe d'une chaîne ou d'une stack enveloppée ;
- une erreur de transport non encodée ou une complétion sans vrai terminal déclenche la récupération et ne simule jamais un passage à `normal` ;
- la finalisation post-forwarding ne laisse jamais un record `sourceSettled` sans terminal bloqué indéfiniment : une réservation restante est annulée, un record avec baseline et vrai terminal devient `handoffReady`, et tout record sans baseline ou sans vrai terminal est retiré/évincé exactement, réveille les subscriptions, retire son pointeur actif et conserve sa retraite/époque ;
- `continueResponse` réutilise la bulle thinking agrégée existante et son texte de base ; cette initialisation est corrélée au `streamInstanceId`, y compris lors d'un retry transport de la même source ;
- la baseline de continuation fige les messages ciblés, mais jamais le brouillon, les attachments, `nextTurnId`, le titre ou les autres champs d'enveloppe susceptibles d'évoluer après le début du stream ; chaque Open superpose les messages immuables sur l'enveloppe persistée la plus récente ;
- les erreurs source transformées en header Slimbk utilisent l'encodage réversible et versionné du `AiErrorMetaValue` défini ci-dessus ; le buffer et le forwarding extension reconstruisent exactement le même chunk ;
- le client Slimbk généré n'attend pas les callbacks `onChunk` asynchrones et appelle `onError` pour un header non-200 sans rejeter nécessairement sa promesse si le callback ne throw pas : le handler normal maintient donc une pipeline FIFO unique pour les chunks ordinaires et le terminal source décodé depuis le header, puis l'attend avant ses cleanups ;
- pour une source Prompt normale, `sourceSettled` et `handoffReady` ne sont pas publiés par l'endpoint backend avant que le handler extension ait drainé cette pipeline ;
- la finalisation normale du registre est atomique et exécutée par le handler extension après le drainage ; elle retourne un résultat strict distinguant `reservationCancelled`, `handoffReady`, `retiredMissingBaseline`, `retiredMissingTerminal` et `stale`, afin que l'Activité, le coordinateur et les tests ne déduisent jamais l'issue à partir d'un état partiel ;
- le son de fin est corrélé au run exact `{ locator, turnId, sourceId }`, y compris pour le terminal source décodé depuis le premier header Slimbk ;
- la libération de lease lors du dispose est dispatchée par adaptateur : EasyAgents conserve `last viewer → kill`, tandis que `normalPrompt` libère seulement la lease et laisse la source continuer ;
- le coordinateur et la réhydratation Activité observent tout run actif, source ou observer, tandis que deletion/Retry conservent l'autorité plus étroite « source normale active ou réservée » ;
- la retraite terminale conserve l'époque terminale du run ; une éviction tardive de l'ancien run après le démarrage d'une source plus récente ne rend jamais l'ancien terminal de nouveau éligible au fallback persistant ;
- les suppressions de chat et le Retry EasyAgents consultent le registre `liveSession` comme autorité, et non uniquement le SidebarStore de présentation.

La tranche préserve également :

- aucun changement 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 ;
- aucune notion de panneau « invisible » : le panneau est `open` ou `closed` ;
- restauration du compositeur normal dès que le snapshot de base est installé et le replay armé ; pendant la réponse, seul le verrou Send normal reste actif, la saisie du prochain brouillon et la préparation des attachments restant autorisées ;
- aucun `*.generated.*` édité manuellement ;
- aucun fichier du répertoire documentaire personnel interdit chargé ou modifié ;
- commentaires, tests et identifiants ajoutés ou modifiés en anglais ;
- aucun `any`, `eslint-disable`, TODO, cast inutile, import dynamique de production ni compatibilité implicite non demandée.

Objectifs décrits par l'utilisateur
Besoin initial exprimé par l'utilisateur

- Livrer toute la tranche finale en une seule implémentation.
- Permettre à une génération Prompt normale de continuer lorsque son panneau est fermé.
- Conserver l'item dans Activité si le panneau est ouvert, ou fermé avec processus encore en cours.
- Supprimer l'item d'Activité si le panneau est fermé et le processus terminé.
- Ajouter une troisième ligne avec un bouton Stop uniquement pour `Running/source/closed`.
- Utiliser l'icône Stop existante, `size="small"`, `variant="danger"`, label `Stop`, avec désactivation immédiate après clic.
- Faire disparaître cette ligne dès la réouverture du panneau ou la fin du processus.
- Réutiliser dans le panneau reconnecté le Stop normal du compositeur.
- Restaurer le Prompt normal au plus tôt, sans read-only prolongé propre à la reconnexion.
- Une fois le replay armé, autoriser saisie et attachments du prochain message, tout en conservant le verrou Send normal.
- Étendre SidebarStore pour représenter les activités de session et pas uniquement les fenêtres ouvertes.
- Conserver le buffer raw-live complet, sans compression ni limitation ajoutée.
- Tester tout code créé, déplacé ou modifié, y compris les courses fermeture/réouverture/arrêt/retry transport/sources successives.
- Mettre à jour la programming doc générique `liveSession` et les documentations d'isomorphisme et de thinking.
- Respecter l'architecture modulaire, le typage strict et toutes les règles de qualité du projet.

Clarifications implicites ajoutées par AICode

- Le dépôt courant inspecté est `0b9f52cc`; la base fonctionnelle de tranche 1 est inchangée par rapport à `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33` et `ed3bd94687cc34f6603730689bfc775818881bae`.
- `LiveSessionRunRef` devient `{ locator, turnId, sourceId }`. `turnId` reste la corrélation logique persistante ; `sourceId` identifie une invocation backend précise.
- `streamInstanceId` reste frontend-only, change à chaque tentative acceptée et ne traverse aucun contrat backend.
- `sourceId` est obligatoire dans toutes les routes/query strings liveSession et tous les guards exacts. Aucune opération exacte ne retombe sur le run actif d'un locator après échec d'identité.
- EasyAgents utilise `sourceId = turnId`, son contrat garantissant une source unique par tour.
- Pour Prompt normal, `useAiStream.start()` génère `sourceId` après acceptation locale du start et génère toujours un nouveau `streamInstanceId`.
- Les callbacks stale sont ignorés avant toute mutation, après chaque `await` et avant tout reset de queue/lock.
- Le registre ajoute des réservations de source entre l'entrée du handler extension et le retour réussi de `askStream(...)`. La réservation est la première mutation synchrone du handler.
- Une réservation n'est pas replayable et ne viole pas « seul `registerRun` crée un record ».
- Réservation → record conserve `sourceStartedAtMs` et `abortRequested`; un abort précoce est appliqué après priming/enregistrement, avant l'interprétation du premier chunk.
- Le registre conserve des retraites structurées. Seule une retraite terminale dont l'époque reste courante autorise un fallback absent-record.
- `baselineAndObservableChunk` reste la policy EasyAgents ; `baselineOnly` est utilisée pour les sources Prompt normales.
- `replayMode` vaut `replaceTurnAssistants` pour prompt/retry/edit/specif/compact-context et `continueAssistant` pour `continueResponse`.
- `continueResponse` capture, dans la section critique de démarrage de la source et sans `await` entre le snapshot et `askStream(...)`, une baseline immuable obtenue par `orchAsk.stream.memory.getThreadMessagesSnapshot(...)`, valide l'assistant ciblé et ne fige pas l'enveloppe entière. Aucun nouvel accès public à OrchMemory n'est nécessaire.
- À chaque Open `continueAssistant`, le backend recharge l'enveloppe persistée la plus récente et remplace uniquement `messages` par la baseline défensive.
- `markBaselineReady` accepte une baseline de messages uniquement pour `continueAssistant`, avec clones défensifs entrée/sortie.
- Le lifecycle `Readable` lazy est extrait du child executor EasyAgents dans un service générique partagé.
- Le stream normal est `ownership='source'`; un 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. Son forwarding ordinaire et le terminal source décodé depuis un header 500 encodé passent par la même pipeline FIFO.
- Chaque chunk normal est appendu au registre avant exposition.

- Le premier `meta.error` source est appendu tel quel au buffer. L'endpoint place le code privé V1 dans `errorCode` et l'encodage préfixé V1 dans `errorMessage`. Le handler extension reconnaît strictement cet encodage dans le `SlimbkCallError`, décode le chunk exact, l'enfile après tous les chunks précédents dans la pipeline FIFO, l'envoie à Activité/son/frontend, marque cette erreur comme source-terminal traitée et laisse l'appel stream se terminer normalement. Le frontend reçoit donc ce cas par `onData`, jamais par `onError`.
- Les erreurs de transport ordinaires non encodées restent canonisées côté frontend en `SYSTEM_ERROR` et donnent `transportError`. La finalisation retire exactement tout record qui ne possède pas de vrai terminal après drainage, afin qu'aucun replay ne puisse attendre indéfiniment un `handoffReady` impossible.
- Les throws source après baseline et EOF sans terminal deviennent un `meta.error` canonique bufferisé/exposé.
- Le service source ne retire pas le run de l'index actif. Le handler extension finalise le registre seulement après retour Slimbk et drainage FIFO.
- Tant que cette finalisation n'a pas eu lieu, le run/réservation reste autoritatif pour coordinateur, Activité, Stop, deletion et cleanup.
- Toute ouverture Prompt passe par une façade coordonnée commune. Une query live explicite valide garde priorité ; une query live explicite invalide ne se dégrade jamais silencieusement.
- Le coordinateur est une classe à dépendances injectées et n'importe pas le low-level opener.
- Ouverture, dispose et cleanup terminal sont sérialisés par locator avec queues FIFO supprimées à idle.
- Le low-level panneau retire synchroniquement le panneau du registre puis délègue les effets Prompt au coordinateur.
- Le coordinateur réévalue panneau et run actif immédiatement avant tout cleanup destructif.
- Le mapping cleanup Prompt est indexé par locator structurel complet.
- Le panneau reconnecté utilise une policy `normalPrompt`, distincte du viewer EasyAgents ; `liveSessionStreaming` n'implique plus automatiquement read-only.
- `not_ready` normalPrompt est retryé avec une seule requête Open en vol et un délai fixe annulable. EasyAgents conserve son échec actuel.
- `transportError` ou complétion sans vrai terminal conduit à `liveSessionReplayFailed`. Une erreur source encodée et décodée par le handler extension est au contraire un terminal de données avec complétion transport normale et peut suivre le handoff direct.
- Le Stop Prompt normal utilise une route source exacte ; l'ancien endpoint thread-only reste réservé aux nettoyages destructifs.
- Le Stop détaché valide synchroniquement item exact, panneau fermé, source exacte active/réservée, non-settled et claim non consommé, puis appelle Orch sans `await` intermédiaire.
- Le son garde l'option utilisateur par thread, mais corrèle l'état de run et la déduplication par runRef exact.
- SidebarStore devient une union stricte : Idle/open ; Running/observer/open ; Running/source/open|closed.
- `queryState.windows` devient `queryState.items` et `SidebarActivityQueryWindowItem` devient `SidebarActivityQueryItem`.
- La réhydratation fusionne panneaux ouverts, observers actifs et sources normales actives/réservées ; un observer sans panneau n'est pas projeté.
- Le registre, et non le SidebarStore, est l'autorité pour deletion et Retry destructifs.
- `continueResponse` réutilise une bulle assistant et une bulle thinking agrégée uniques, avec `baseThinkingTextRef`, séparateur canonique et reset par `streamInstanceId`.
- Aucun résultat généré n'est listé comme fichier à éditer.

Vue d'ensemble de la méthode choisie

Identité, réservations et machine d'état

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 ;
5. publication Activité Running/source avec état réel du panneau ;
6. armement du son exact ;
7. persistance éventuelle de la préférence SPECIF Code ;
8. chargement catalogue ;
9. capture synchrone et défensive des messages de continuation dans la section critique de démarrage, si nécessaire ;
10. retour réussi de `askStream(...)` ;
11. priming du `Readable` ;

12. réservation → record et application d'un abort précoce ;
13. append de chaque chunk avant exposition ;
14. baseline durable ;
15. replay possible ;
16. vrai terminal ou synthèse canonique ;
17. fin de consommation backend sans retrait du run actif ;
18. retour Slimbk et drainage de la pipeline FIFO extension ; si le premier résultat est un header 500 contenant le protocole privé V1, le handler le décode en chunk source terminal, l'enfile dans la même FIFO et ne le transforme pas en erreur transport ;
19. finalisation atomique post-forwarding : handoff si baseline + vrai terminal, sinon retraite/éviction exacte ;
20. Activité Idle si panneau ouvert et handoff réussi, suppression si fermé ou si le run a été retiré ;
21. cleanup mémoire/attachments/son/trivial uniquement sans panneau ni nouvelle source.

Replay, erreurs et baselines

`replaceTurnAssistants` retire les assistants du tour exact puis rejoue la source. `continueAssistant` conserve les messages exacts d'avant continuation, tout en rechargeant l'enveloppe persistée la plus récente à chaque Open.

Le protocole du premier `meta.error` est :

- le service source append le chunk original au buffer ;
- l'endpoint envoie `500_SERVER_ERROR`, `errorCode='AICODE_LIVE_SESSION_SOURCE_ERROR_V1'` et `errorMessage='AICODE_LIVE_SESSION_SOURCE_ERROR_V1:' + JSON.stringify({ version: 1, value })` ;
- le client généré transmet ce header au callback `onError` sous forme de `SlimbkCallError` et n'attend pas les callbacks async ;
- le handler extension tente le décodage strict ; si le payload privé est valide, il reconstruit le chunk exact avec le `turnId` du run, l'ajoute à la même pipeline FIFO que les `onChunk`, met à jour son/Activité, l'émet au frontend et n'effectue aucun throw pour ce cas traité ;
- l'appel Slimbk se termine alors normalement, ce qui permet au frontend d'observer le vrai terminal via `onData` puis une complétion transport normale ;
- si le payload n'est pas le protocole privé valide, le handler conserve la sémantique d'erreur transport ; le frontend crée le `SYSTEM_ERROR` canonique ;
- un EOF source sans terminal devient `INTERRUPTED` avant l'endpoint et suit le chemin de données terminales normal.

Le terminal bufferisé ne rend pas le run handoff-ready : la finalisation post-forwarding reste obligatoire. Un record qui ne possède toujours pas de vrai terminal après le drainage est évincé/retiré, et non laissé dans l'état `sourceSettled` en attente éternelle.

Cycle de vie du panneau

Le wiring low-level compose le coordinateur, qui reçoit opener, panneau registry, liveSession resolver, releases adapter-specific, transitions Activité, cleanup thread/son/trivial et UI chat. Les releases :

- `easyAgentsBatch + stopOnLastViewerClose` → policy EasyAgents dernier viewer = hard stop ;
- `normalPrompt + continueOnPanelClose` → release de lease sans abort ;
- descripteur invalide → fail-fast, sans fallback ni cleanup destructif d'un run actif.

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

Frontend reconnecté

Après installation du snapshot et armement du replay, le compositeur et les attachments redeviennent visibles. `isStreaming` maintient le verrou Send et le Stop normal. Chaque retry possède un nouveau `streamInstanceId`. Les callbacks antérieurs ne peuvent plus muter le runtime. Le handoff normal est direct uniquement après vrai terminal de données, drainage Queue A/Queue B, finalisation Prompt et outcome `completed` de la tentative courante. Si la subscription se termine par éviction sans vrai terminal, le hook entre dans `liveSessionReplayFailed` au lieu de passer à `normal`.

Activité, son et Stop

Le Stop détaché utilise `VscButton` : `icon="SquareStop"`, `size="small"`, `variant="danger"`, label `Stop`, `disableAfterClick`, `enabled={!stopRequested}`, testId stable basé sur `sourceId`. Le lien de titre

reçoit un testId stable basé sur le locator. Le son et les transitions Activité consomment le runRef exact et la même séquence de chunks que le frontend, y compris le terminal reconstruit depuis le premier header d'erreur source.

Liste exacte des fichiers avec détails d'implémentation

Contrats shared

1) LiveSessionReplay.type.ts — créé

- Définir `ZLiveSessionReplayMode` / `LiveSessionReplayMode` : `replaceTurnAssistants | continueAssistant`.
- Documenter la reconstruction frontend de chaque branche.

2) LiveSessionRun.type.ts — modifié

- Ajouter `LiveSessionSourceId` strict trim/non vide et `sourceId` obligatoire au runRef.
- Inclure `sourceId` dans la run key, sans modifier la locator key.
- Documenter `turnId` logique et `sourceId` exécutionnel.

3) LiveSessionPanel.type.ts — modifié

- Ajouter paramètres source/adaptateur et adaptateur strict `easyAgentsBatch | normalPrompt`.
- Réutiliser le schéma sourceId ; rendre `continueOnPanelClose` effectif pour normalPrompt.

4) LiveSessionOpenResult.type.ts — modifié

- Ajouter `sourceId` aux branches live/terminal et `replayMode` uniquement à live.
- Étendre transport et `superRefine`, y compris propriété explicitement présente à `undefined`.

5) LiveSessionContracts.test.ts — modifié

- Couvrir clés, collisions, query params, adaptateurs, replay modes, transports et sourceId obligatoire.

6) aiStopSuffix.util.ts — modifié

- Ajouter les fabriques canoniques des `meta.error` transport/source.
- Définir le protocole privé exact V1 : code `AICODE_LIVE_SESSION_SOURCE_ERROR_V1`, préfixe identique dans `errorMessage`, JSON strict `{ version: 1, value: AiErrorMetaValue }`.
- Ajouter encodeur/décodeur réversible ; le décodeur accepte `Error`, objet structurel `SlimbkCallError`, chaîne et stack enveloppée, mais exige le code lorsqu'il est disponible et valide strictement le payload.
- Produire `SYSTEM_ERROR` / `INTERRUPTED` canoniques pour les erreurs non encodées.
- Garantir que le chunk décodé par l'extension est identique au chunk bufferisé.

7) aiStopSuffix.util.test.ts — modifié

- Tester Error/string/status/errorCode, réversibilité, stacks/prefixes, détails multilignes, invalide, SYSTEM_ERROR, INTERRUPTED et identité du chunk.

8) SidebarStore.ts — modifié

- Remplacer le type fenêtre par l'union stricte `SidebarActivityQueryItem`.
- Branches Idle/open, Running observer/open, Running source/open|closed avec identités exactes et `stopRequested`.
- Préserver titre, elapsed, `updatedAt`, `openedAt`; renommer `windows` en `items`.

9) SidebarUiMessages.ts — modifié

- Ajouter `openQueryActivity(locator)` et `stopDetachedQuery(runRef) -> {accepted}`.

10) MessageRegistry.ts — modifié

- Ajouter `sourceId` aux streams normal et liveSession ; ne jamais transporter `streamInstanceId`.

Backend Orch `liveSession`

11) liveSession.programming-doc-readme.md — modifié

- Documenter identité, réservations, readiness, replay, continuation, protocole header V1 décodé dans le handler extension, ownership, lifecycle, Activity, son, retraites, retrait exact des runs sans terminal et barrière post-forwarding.

12) LiveSessionRecord.type.ts — modifié

- Ajouter `baselineOnly`, replayMode, baseline continuation, timestamp, abortRequested, viewerGroup optionnel, réservations/snapshots/retraites/époque terminale.
- Définir l'union de résultat de finalisation post-forwarding.
- Étendre permits/subscriptions à sourceId et interdire les baselines incohérentes.

13) LiveSessionRegistry.service.ts — modifié

- Réservations/cancel exacts, transformation atomique vers record, listings actifs génériques/source, existence et claims atomiques.
- Supprimer `canAbortSource`.
- Finalisation normale post-forwarding idempotente et exacte.
- Cette finalisation doit évincer/retirer les records sans baseline ou sans vrai terminal, réveiller les subscriptions et supprimer le pointeur actif ; aucun record settled non-handoff ne doit rester bloqué.
- Retraites structurées et époques empêchant la résurrection d'un ancien terminal.
- Préserver leases, retention et replay tranche 1.

14) LiveSessionRegistry.service.test.ts — modifié

- Couvrir sources successives, réservations, baselineOnly, baseline messages, abort, listings, claims, finalisation, retraites, époques, A-terminal/B-start/A-eviction tardive et baseline sans terminal retirée sans waiter bloqué.

15) LiveSessionService.ts — modifié

- Exposer toutes les nouvelles opérations du registre, supprimer `canAbortSource`, conserver Open/Replay/release/sweep/evict.
- Exposer le résultat strict de finalisation post-forwarding.

16) LiveSessionService.test.ts — modifié

- Couvrir parcours normal, réservations, réouverture, sources successives, claims, finalisation, retraite stale et finalisation sans terminal.

17) LiveSessionOpen.service.ts — modifié

- `not_ready` pour réservation exacte.
- `replaceTurnAssistants` : filtrage assistants du tour sur enveloppe actuelle.
- `continueAssistant` : enveloppe persistée récente + remplacement du seul tableau messages.
- Retourner sourceId/replayMode et contrôler le fallback terminal par retraite+époque.
- Un run retiré faute de terminal ne doit jamais être classé comme live ou handoff-ready.

18) LiveSessionOpen.service.test.ts — modifié

- Couvrir réservation, baselineOnly, continuation/enveloppe récente, mismatch, sources mêmes turn, retraites, retrait faute de terminal et état live avant finalisation post-forwarding.

19) LiveSessionReplay.service.test.ts — modifié

- Ajouter sourceId, isolation ancienne subscription/nouvelle source, attente de finalisation après terminal drainé et terminaison immédiate par éviction quand aucun vrai terminal n'existe.

20) LiveSessionSourceLifecycle.service.ts — créé

- Extraire le lifecycle lazy Readable OOP générique : création, first `next`, source-start sync, first result/tail, cleanup/abort/return/destroy, settlement produit exact et agrégation déterministe.
- Ne contenir aucune sémantique EasyAgents/Prompt ni publication registre normalPrompt.

21) LiveSessionSourceLifecycle.service.test.ts — créé

- Généraliser les tests priming/cleanup et vérifier que le callback produit reste l'unique frontière de settlement.

22) NormalPromptLiveSessionSource.service.ts — créé

- Mapper AiQuery → replayMode, readiness baselineOnly.
- Capturer sans `await` entre snapshot et `askStream(...)` la baseline messages de continuation via `orchAsk.stream.memory.getThreadMessagesSnapshot(...)`, puis valider l'assistant.
- Enregistrer après priming, transmettre timestamp, appliquer abort réservé.
- Parser, append-before-yield, baseline, vrai terminal.
- Premier meta.error appendu inchangé puis exposé à l'endpoint pour encodage header V1.
- Transformer throw/EOF sans terminal en erreur canonique.
- Ne pas finaliser le registre ; garantir qu'un échec registration ne touche aucun tiers.

23) NormalPromptLiveSessionSource.service.test.ts — créé

- Couvrir tous les query types, continuation, append order, abort, premier error/header, throw, EOF, absence de settlement/handoff avant finalisation et isolation registration.

Adaptation EasyAgents

24) EasyAgentsBatchLiveSessionAdapter.service.ts — modifié

- `replayMode='replaceTurnAssistants'`, `sourceId=turnId`, timestamp.
- Injecter lifecycle partagé ; supprimer seam `canAbortSource`.
- Préserver settlement/handoff métier EasyAgents.

25) EasyAgentsBatchLiveSessionAdapter.service.test.ts — modifié

- Vérifier identité, replayMode, timestamp, nouveau claim et ordre historique.

26) EasyAgentsBatchRunExecution.service.ts — modifié

- Construire tous les runRefs avec `sourceId=liveTurnId`, sans changer les transitions Batch.

27) EasyAgentsBatchRunExecution.live-activity-and-gating.service.test.ts — modifié

- Étendre assertions identité/replayMode et préserver Watchable/Handoff.

28) EasyAgentsBatchRunChildExecution.service.ts — modifié

- Remplacer lifecycle local par service partagé, conserver parsing/classification/callbacks.

29) EasyAgentsBatchRunChildExecution.service.test.ts — modifié

- Conserver intégration Batch et déplacer tests lifecycle purs.

30) EasyAgentsBatchRunManager.service.ts — modifié

- Injecter lifecycle, migrer `items`, consulter `hasActiveOrReservedSourceForLocator` pour Retry.

31) EasyAgentsBatchRunExecution.testHarness.ts — modifié

- Injecter lifecycle et runRefs sourceId.

32) EasyAgentsBatchRunLifecycle.service.test.ts — modifié

- Adapter construction au lifecycle partagé.

33) EasyAgentsBatchLastViewerStop.service.test.ts — modifié

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

34) easyAgents.programming-doc-readme.md — modifié

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

Endpoints et message bus

35) open.endpoint.ts — modifié

- Exiger sourceId et transporter sourceId/replayMode.

36) open.endpoint.test.ts — modifié

- Couvrir champs, branches et mismatch.

37) stream.endpoint.ts — modifié

- Exiger sourceId et relayer runRef complet.

38) stream.endpoint.test.ts — modifié

- Couvrir exactitude, mismatch, terminal, éviction sans terminal et attente de finalisation post-forwarding.

39) abort.endpoint.ts — modifié

- Exiger sourceId ; claim viewer atomique avec lease active ; claim et abort dans la même pile.

40) abort.endpoint.test.ts — modifié

- Couvrir viewer EasyAgents/normal, stale, double Stop, remplacement et idempotence.

41) abortSource.endpoint.ts — créé

- Stop exact source Prompt normal par locator+turn+source, claim puis abort sans await.

42) abortSource.endpoint.test.ts — créé

- Couvrir réservation, record, stale, double clic, absent et ordre.

43) stream.endpoint.ts — modifié

- Ajouter sourceId, préserver ordre SPECIF/catalogue, déléguer au service source.
- Premier meta.error : header 500 avec `errorCode` et `errorMessage` du protocole privé V1 ; ne jamais bufferiser une autre représentation.
- Ne pas finaliser le registre ; terminer après consommation/canonisation.

44) stream.endpoint.test.ts — modifié

- Tester délégation, ordre, premier chunk, header encodé exact, absence double forwarding et absence de finalisation prématurée.

45) liveSessionStreamHandlers.ts — modifié

- Relayer sourceId.
- Source ownership : aucune activité/son supplémentaire.
- Observer : ressources uniquement si panneau exact ouvert.
- RunRef exact au son ; conserver pipeline async et cleanup partiel.

46) liveSessionStreamHandlers.test.ts — modifié

- Couvrir sourceId, ownership, observer fermé, son exact, absence duplication, éviction sans terminal et ordre.

47) iaStreamHandlers.ts — modifié

- Recevoir sourceId et construire runRef exact.
- Réserver synchroniquement avant tout await, publier Activity et armer son exact.
- Transmettre sourceId à l'endpoint.
- Maintenir une FIFO unique. `onChunk` y enfile le chunk ordinaire. `onError` tente d'abord de décoder le protocole privé V1 : si valide, il enfile le chunk terminal exact dans cette même FIFO, le fait passer par Activity/son/emit, mémorise que l'erreur source a été traitée et retourne sans throw ; sinon il conserve l'erreur transport et la propage.
- Après retour Slimbk, attendre la FIFO. Une erreur source encodée traitée n'est pas un `transportError`; toute autre erreur l'est.
- Finaliser atomiquement le registre après FIFO, avant son Stop et Activity Stop ; retirer exactement les records sans vrai terminal.
- Exécuter tous les cleanups, agréger les erreurs dans l'ordre, puis coordinateur `onSourceSettled` avec le résultat strict de finalisation.

48) iaStreamHandlers.queryDoneSound.test.ts — modifié

- Vérifier runRef/sourceId, FIFO avant finalisation/son, fermeture sans cleanup prématuré, source suivante.
- Ajouter premier meta.error encodé : décodage extension, son d'erreur unique, émission data avant done, aucune erreur transport frontend.
- Ajouter terminal backend produit mais callback encore en transit et run sans terminal retiré après drainage.

49) iaStreamHandlers.runIdPropagation.test.ts — modifié

- Vérifier réservation, Activity, FIFO, finalisation, cleanup et ordre strict.
- Vérifier que le terminal décodé depuis header rejoint la FIFO après les chunks déjà reçus et avant finalisation.
- Vérifier qu'une erreur transport sans terminal retire le run et réveille un replay attaché.

50) PromptQueryDoneSound.manager.ts — modifié

- Corréler état actif/dédup par runRef exact, callbacks stale no-op, pas de création implicite.
- Préserver TOOL_USE/ABORTED/fréquences et traiter l'erreur source décodée comme le même terminal que le buffer.

51) PromptQueryDoneSound.manager.test.ts — modifié

- Couvrir mêmes turns/sources différentes, stale chunk/stop et absence de création tardive.

Activité runtime extension

52) SidebarQueryRunningBridge.ts — modifié

- Renommer classe interne `SidebarQueryActivityRuntimeService` sans déplacer le fichier.
- Stocker runRef, ownership, timer, startMs, runId par locator ; guards sourceId+runId.
- Source open/closed ; observer open seulement ; piloter transitions de l'union.

53) SidebarQueryRunningBridge.race.test.ts — modifié

- Couvrir sources successives, stale, observer fermé et panel source fermé/rouvert.

54) SidebarQueryRunningBridge.usage.test.ts — modifié

- Usage exact et transition Idle/suppression, y compris retrait sans terminal.

Coordination panneau Prompt

55) PromptLiveSessionPanelQuery.util.ts — créé

- Parser locator et descripteur complet, construire query normalPrompt, distinguer classique/explicite valide/invalide, retirer autoRun lors injection.

56) PromptLiveSessionPanelQuery.util.test.ts — créé

- Couvrir identité, adapter, ordre, policies, invalide, suppression autoRun.

57) PromptLiveSessionPanelCoordinator.service.ts — créé

- Classe OOP injectée, aucun import low-level.
- Queue FIFO par locator supprimée à idle.
- `openPromptPanel`, `onPanelDisposed`, `onSourceSettled` avec réévaluation avant cleanup.
- Préserver live explicite ; injecter normalPrompt seulement sans descripteur et avec source exacte active/réservée.
- Dispatch release EasyAgents/normalPrompt, detach source sans abort, ignorer settlement stale.
- Traiter le résultat `retiredMissingBaseline|retiredMissingTerminal` comme fin autoritative sans handoff : panneau fermé → cleanup ; panneau ouvert/replay attaché → laisser le frontend afficher `liveSessionReplayFailed` sans restaurer artificiellement `normal`.

58) PromptLiveSessionPanelCoordinator.service.test.ts — créé

- Couvrir priorités, releases, query invalide, races open/cleanup, pré-baseline, observer, source suivante, drainage post-terminal, retrait sans terminal et protection mémoire/son.

59) PromptLiveSessionDetachedStop.service.ts — créé

- Valider synchroniquement closed + Running/source exact + source active/réservée + claim non consommé ; mutation stopRequested puis abort sans await.

60) PromptLiveSessionDetachedStop.service.test.ts — créé

- Couvrir accept/refus, reopened, replacement, double clic, réservation, store stale.

61) PromptThreadCleanup.manager.ts — modifié

- Indexer par locator structurel ; séparer register, detach actif, cleanup historique, cleanup terminal sans abort/await et orphan cleanup.
- Ne jamais Abort/ClearMemory une source normale active.

62) PromptThreadCleanup.manager.attachmentsCleanup.test.ts — modifié

- Couvrir scopes, detach, idle, terminal, mapping absent et sérialisation attachments.

63) openPanel.util.ts — modifié

- Composer un coordinateur module-scoped et exposer façade Prompt async, conserver low-level autres pages.
- Capture descriptor initial ; dispose unregister panneau synchronique puis délégation complète.
- Adapter titre Running à `items/status`.

64) openPanel.util.test.ts — modifié

- Wiring sans cycle, identité/adaptateur, reveal, dispose unique, unregister avant coordinateur, ports release, pages non-Prompt et fixtures `items`.

65) uiGenericHandlers.ts — modifié

- Await façade coordonnée pour Prompt, low-level pour autres pages.

66) uiGenericHandlers.openExternalUrl.test.ts — modifié

- Mocker façade Prompt et préserver URL.

67) NewChat.command.ts — modifié

- Await ouverture coordonnée.

68) NewChat.command.test.ts — modifié

- Vérifier async et propagation erreur.

69) registerDeepLinkUriHandler.ts — modifié

- Deep links chat via façade coordonnée, non-bloquant avec catch explicite.

70) registerDeepLinkUriHandler.scope.test.ts — modifié

- Couvrir root/nested et guards.

SidebarStore et Activité

71) SidebarStore.queryWindow.listeners.manager.ts — modifié

- Transitions atomiques `panelOpened`, `sourceStarted`, `sourceSettled`; lecture panelState et sourceStarted sans await intermédiaire.
- `sourceSettled` reçoit le résultat strict de finalisation pour distinguer Idle/handoff et suppression/retrait.

72) SidebarStore.queryWindow.mutations.manager.ts — modifié

- `windows→items`, `panelClosed`, `forceRemove`, elapsed et stopRequested exacts ; conserver seulement Running/source fermé.

73) SidebarStore.core.manager.ts — modifié

- Initialiser items et appeler `rehydrateQueryActivities`.

74) SidebarStore.rehydrate.manager.ts — modifié

- Fusionner panneaux, observers actifs et sources/réservations ; dédupliquer et interdire observer/closed/idle closed.
- Elapsed source depuis timestamp ; terminal bufferisé non finalisé reste Running ; run retiré sans terminal n'est jamais re-projeté.

75) SidebarStore.rehydrate.manager.test.ts — modifié

- Couvrir toutes les projections, réservation, observer sans panneau, dédup, fenêtre post-backend/pré-forwarding et retrait sans terminal.

76) SidebarStore.manager.queryRunningUpsert.test.ts — modifié

- Tester machine d'état et identités.

77) SidebarStore.chatTitles.manager.ts — modifié

- Lire items et préserver fallback title source fermée/running.

78) SidebarStore.handlers.manager.ts — modifié

- Enregistrer open activity et Stop détaché, déléguer coordinateur/service.

79) SidebarStore.handlers.manager.test.ts — créé

- Vérifier payloads, accepted et délégation.

80) delete.endpoint.ts — modifié

- Lire items ; bloquer si projection Running ou source active/réservée ; fermer/supprimer après validation autoritative.

81) delete.endpoint.liveSession.test.ts — créé

- Couvrir open/closed/réservation/store stale/absent, terminal bufferisé avant drainage et run retiré sans terminal.

82) list.endpoint.windowsLock.eperm.test.ts — modifié

- Adapter fixtures items.

83) diskAliases.lastWins.endpoints.test.ts — modifié

- Adapter items et préserver delete disque.

84) ChatPersistenceWriter.windowsFallback.test.ts — modifié

- Adapter fixture items.

85) OrchMemory.naive_title.user_prompt_specif_multiline.e2e.repro.test.ts — modifié

- Adapter fixture items.

86) SidebarActivityQuery.tsx — modifié

- Utiliser items/union ; ouvrir via nouveau message ; Stop exact sur troisième ligne source/closed.
- `SquareStop`, small, danger, disableAfterClick, enabled, testIds ; sous-composant/callbacks mémorisés sans hooks dans map.

87) SidebarActivityQuery.test.tsx — modifié

- Couvrir locators, observer/source, Stop, disparition, payload, désactivation et testIds stables sans labels fragiles.

Frontend `liveSession` et Prompt

88) NormalPromptLiveSession.adapter.ts — créé

- Parser normalPrompt, sourceId, lease, close policy.
- Policies not_ready retry, handoff direct, replay recovery, surface interactive ; aucune action Stop banner.

89) NormalPromptLiveSession.adapter.test.ts — créé

- Couvrir parsing, policies et view models.

90) PromptLiveSessionBootstrap.adapter.ts — créé

- Dispatch strict EasyAgents/normalPrompt, aucun fallback.

91) PromptLiveSessionBootstrap.adapter.test.ts — créé

- Couvrir dispatch, partiels et inconnus.

92) PromptLiveSession.types.ts — modifié

- Ajouter policies et `liveSessionReplayFailed`; supprimer mode⇒viewer et helper mort ; runRef sourceId.

93) usePromptLiveSession.hook.ts — modifié

- Transporter sourceId/replayMode, retry not_ready annulable, replay selon mode.
- Handoff direct normal uniquement vrai terminal data + queues/finalisation + outcome `completed`.
- `transportError`, completed sans terminal ou subscription évincée sans terminal → replayFailed.
- Une erreur source encodée ne passe pas ici par `onError` : elle arrive comme chunk data décodé/forwardé par l'extension et peut donc satisfaire le handoff direct.
- Chaque retry recharge baseline et crée un nouveau streamInstanceId.

94) usePromptLiveSession.hydration.test.tsx — modifié

- Ajouter identité/replayMode et préserver EasyAgents.

95) usePromptLiveSession.handoff.test.tsx — modifié

- Vérifier EasyAgents terminal, normal direct, refus transportError/completed sans terminal/éviction sans terminal et acceptation d'un meta.error reçu comme data avec completed.

96) usePromptLiveSession.normalPrompt.test.tsx — créé

- Couvrir not_ready, surface interactive, draft/attachments, Stop, replay failure/retry, direct handoff, terminal, éviction sans terminal et close/reopen.

97) usePromptLiveSession.normalPromptIsomorphism.test.tsx — créé

- Comparer stream continu/replay/hydratation pour tous query types, thinking, tools, usage, edits, erreurs canoniques, EOF et transport interrompu sans terminal.

98) EasyAgentsBatchLiveSession.adapter.ts — modifié

- Ajouter sourceId=turnId/adaptateur, préserver UX.

99) EasyAgentsBatchLiveSession.adapter.test.ts — modifié

- Adapter query, sourceId et dispatcher.

100) AiStreamStartArgs.type.ts — modifié

- Garder promptQuery public sans sourceId ; état interne runRef+streamInstanceId ; outcome `pending|completed|transportError`.

101) useAiStream.hook.ts — modifié

- Générer identités après acceptation, callbacks par tentative et guards stale complets.
- Exposer runRef, streamInstanceId, outcome.
- Stop exact selon source/observer.
- Les erreurs source encodées ne sont pas décodées ici : elles sont déjà reçues via `onData` comme `meta.error` exact après décodage extension.
- `onError` est réservé aux erreurs de transport non traitées et utilise la fabrique canonique SYSTEM_ERROR.
- Ne modifier aucune Queue/lock/sanitizer/raw-live.

102) useAiStream.hook.liveSessionObserver.test.ts — modifié

- Ajouter identité, routes Stop, outcome, retry/stale ; vérifier qu'un meta.error reçu par data donne completed, tandis qu'un vrai onError donne transportError.

103) useAiStream.hook.promptSourceIdentity.test.ts — créé

- Vérifier unicité, stabilité, abort exact, callbacks stale et observer retry.

104) usePromptRunController.hook.ts — modifié

- Observer reçoit runRef+lease+replayMode ; base thinking + streamInstanceId ; exposer outcome/tentative ; reset complet.

105) usePromptRunController.liveSessionObserver.test.tsx — modifié

- Couvrir identité, modes, base thinking, tentative et outcome.

Continuation et thinking

106) preparePromptContinuationStreaming.util.ts — créé

- Trouver assistant non-thinking exact, capturer bases assistant/thinking, réutiliser thinking agrégé ou créer placeholder, réutiliser UID assistant, armer séparateur, reset refs/buffers/notices.

107) preparePromptContinuationStreaming.util.test.ts — créé

- Couvrir assistant, bases, thinking, séparateur, manquant, sources successives, aucune duplication.

108) preparePromptStreamingTurn.util.ts — modifié

- Reset explicite base thinking/séparateur pour replace.

109) preparePromptStreamingTurn.util.test.ts — modifié

- Vérifier resets.

110) usePromptThinkingLifecycle.hook.ts — modifié

- Recevoir base thinking et streamInstanceId ; initialiser avant effets transport ; séparateur unique ; préserver historique ; reset replace.

111) usePromptThinkingLifecycle.continuation.test.tsx — créé

- Couvrir base, append, tool/text avant thinking, réinsertion, reset et retry même source/nouvelle tentative.

112) usePromptToolUse.hook.ts — modifié

- Supprimer bulle vide uniquement si baseThinking vide.

113) usePromptRunController.finalization.hook.ts — modifié

- Reset base thinking/séparateur après terminal, ordre inchangé.

114) usePromptRunCommands.hook.ts — modifié

- Remplacer bootstrap continue inline par helper ; préserver autres commandes.

115) usePromptRunCommands.continueResponse.test.tsx — créé

- Vérifier helper, même turn, nouvelle source/tentative, thinking unique, assistant exact.

Intégration Prompt et documentation

116) Prompt.tsx — modifié

- Capturer source/adaptateur dans query figée, dispatcher, passer outcome/runRef/tentative.
- Déduire interactions de surface/policy ; draft/interactions normaux pendant replay normalPrompt, y compris la persistance debounced du brouillon lorsque le mode n'est pas encore `normal` mais que la policy autorise la surface interactive.
- Cleanup ownership différé jusqu'au handoff direct ; hydration classique off pour tout live bootstrap.

117) Prompt.liveSessionBootstrap.test.tsx — modifié

- Couvrir dispatch, identité, surface, draft, query figée, hydration off, replay failure, éviction sans terminal et ownership cleanup.

118) PromptInput.batchLiveViewerMode.test.tsx — modifié

- Préserver viewer EasyAgents et ajouter normalPrompt interactif avec Stop normal, sans banner Stop.

119) EasyAgentsBatch.liveWatch.test.tsx — modifié

- Adapter query exacte source/adaptateur.

120) EasyAgentBatchSurface.test.tsx — modifié

- Adapter query exacte sans changement UX.

121) isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md — modifié

- Ajouter troisième chemin fermeture/réouverture, continuation, thinking, retry, erreurs canoniques, vrai terminal, retrait sans terminal et barrière post-forwarding.
- Documenter que l'erreur source du premier header est transformée en chunk de données exact par le handler extension avant le frontend.

122) thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md — modifié

- Documenter placeholder vide, bulle agrégée, séparateur et reset 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, retry de même source, réservations, callbacks stale, lifecycle lazy, pipeline non attendue, protocole exact du premier header source-error, finalisation post-forwarding, retrait des records sans terminal, EOF, continuation, thinking, panneau/cleanup, Activity, son, handoff, draft/attachments, Stop pré-enregistrement, deletion et retraites.
3. **Justification :** la tranche modifie le chemin normal de toutes les requêtes Prompt et plusieurs coordinations asynchrones. Le traitement du premier header d'erreur est particulièrement critique : si le handler le laisse remonter comme erreur transport, le frontend ne peut pas satisfaire la règle « vrai terminal data + complétion normale », le son ne voit pas le terminal exact et le handoff direct devient impossible pour cette classe d'erreurs. De même, si une finalisation sans vrai terminal laisse un record simplement `sourceSettled`, le replay actuel attend explicitement `handoffReady` ou `evicted` et peut rester bloqué indéfiniment. La reconstruction extension dans la FIFO unique, l'éviction exacte des runs non-handoffables, les identités exactes, réservations, époques et tests d'isomorphisme sont donc obligatoires.

Résumé

- Étendre `liveSession` aux sources Prompt normales en une implémentation complète.
- Ajouter sourceId de bout en bout et streamInstanceId frontend.
- Réserver puis enregistrer les sources, append-before-forward et partager le lifecycle.
- Drainer une FIFO extension unique avant finalisation.
- Décoder le premier header d'erreur source via un protocole V1 exact, l'enfiler comme chunk terminal exact, l'émettre comme donnée et conserver une complétion transport normale.
- Finaliser atomiquement après drainage : `handoffReady` seulement avec baseline + vrai terminal ; sinon retraite/éviction exacte afin qu'aucun replay ne reste bloqué.
- Capturer les messages de continuation sans figer l'enveloppe.
- Réutiliser assistant/thinking sans duplication.
- Ne plus tuer une source normale à la fermeture et différer les cleanups.
- Sérialiser panneau/dispose/cleanup par locator.
- Modéliser Activité avec ownership et panneau open/closed.
- Ajouter uniquement le Stop détaché source/closed.
- Corréler son et contrôles au runRef exact.
- Utiliser le registre comme autorité deletion/Retry.
- Préserver EasyAgents, queues, locks et sanitizers.
- Tester stream continu, replay, hydratation, erreurs source encodées, transport réel et éviction sans terminal.

Nom du commit
`feat(live-session): keep prompt runs alive across panel close and reopen`
SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_022043
Copy
Refine
Code
Verify

Sévérité	Description du changement
🟥 majeure	La finalisation post-forwarding retire désormais explicitement tout record sans baseline ou sans vra… des subscriptions et suppression du pointeur actif.
🟥 majeure	Ajout d'un résultat strict de finalisation : `reservationCancelled`, `handoffReady`, `retiredMissingBaseline`, `retire…l` ou `stale`.
🟥 majeure	Définition exacte du protocole Slimbk source-error V1 avec `errorCode`, préfixe versionné et payl…

Sévérité	Description du changement
🟧 mineure	Clarification de la capture `continueResponse` via `getThreadMessagesSnapshot(...)`, sans `await` avant `askStream(...)` et sans nouvelle API OrchMemory.
🟧 mineure	Ajout des scénarios d’éviction sans terminal aux tests du registre, replay, handlers, coordinateur, A
🟧 mineure	Actualisation de la référence du dépôt inspecté au commit courant `0b9f52cc`, tout en confirmant l fonctionnelle de la tranche 1.

</ASSISTANT>
SHOW ALLPage 1 / 1