Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=8c84850903d647d19cc1c6caff7fd500-RS-Affiner-Finaliseu
<USER>
[Refine]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260711_232041
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
scanning git: "État Git courant", "Commit tranche 1"
reading project: t "Architecture liveSession", t "LiveSession frontend", t "Panneaux Prompt", liveSession.programming-doc-readme.md, easyAgents.programming-doc-readme.md, isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md, prompt-stream-dispatch-lock.programming-doc-readme.md, thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md
scanning git: "Révision courante"
reading project: fallback-persist-and-title.programming-doc-readme.md, h "Cœur liveSession", h "Lifecycle source Batch", e "Champ windows activité", e "Type activité fenêtre", e "Ouvertures Prompt", h "Fermeture Prompt", h "Transport stream Prompt", h "Continuation Prompt", e "Observateur liveSession", e "Barrière fallback",
LiveSessionRun.type.ts,
LiveSessionPanel.type.ts,
LiveSessionOpenResult.type.ts,
LiveSessionStream.util.ts, LiveSessionRecord.type.ts, LiveSessionRegistry.service.ts, LiveSessionService.ts,
LiveSessionOpen.service.ts,
LiveSessionReplay.service.ts, SidebarStore.ts, SidebarUiMessages.ts, MessageRegistry.ts [L1-L220], e "Usages run ref", e "Clé run liveSession", e "Paramètre turn live", e "Texte query turn live", e "Policy fermeture live", e "Service live production", e "Constructeurs runRef", e "Locator activité",
openPanel.util.ts,
PromptThreadCleanup.manager.ts,
OpenedPanelsRegistry.util.ts,
iaStreamHandlers.ts,
liveSessionStreamHandlers.ts,
SidebarQueryRunningBridge.ts,
SidebarStore.queryWindow.listeners.manager.ts,
SidebarStore.queryWindow.mutations.manager.ts,
SidebarStore.core.manager.ts,
SidebarStore.rehydrate.manager.ts,
SidebarStore.handlers.manager.ts,
SidebarActivityQuery.tsx,
AiStreamStartArgs.type.ts,
useAiStream.hook.ts,
PromptLiveSession.types.ts,
usePromptLiveSession.hook.ts,
EasyAgentsBatchLiveSession.adapter.ts,
PromptLiveSessionBanner.tsx, usePromptRunController.hook.ts [L430-L580], usePromptRunCommands.hook.ts [L320-L500],
preparePromptStreamingTurn.util.ts,
stream.endpoint.ts,
EasyAgentsBatchRunChildExecution.service.ts, EasyAgentsBatchLiveSessionAdapter.service.ts, e "Mode viewer Prompt", e "Ownership cleanup Prompt", e "Vue live Prompt", e "Compositeur viewer", e "Enregistrement thread", e "Bootstrap openMode", h "Interactions Prompt", Prompt.tsx, PromptInput.tsx [L100-L180], PromptInput.tsx [L1500-L1620], uiAiPromptHandlers.ts [L90-L150], OrchAsk.ts, OrchAskStream.ts [L1-

L220], OrchAskStreamTurnBootstrap.ts [L1-L360], OrchAskStreamWiring.ts [L360-L460], h "Snapshot mémoire Orch", e "Méthodes mémoire snapshot", e "Snapshot session mémoire", e "Persist continuation", e "Persistance snapshot", h "Clone session chat", VscButton.tsx, e "Bouton disableAfterClick", e "Icône SquareStop", SidebarActivityQuery.test.tsx, UiMessages.ts, SidebarUiMessages.ts, e "Réhydrate query windows", e "Ajout query window", e "Suppression query window", e "Running query window", e "Elapsed query window", e "État query Sidebar", e "Fenêtres dans tests", PromptQueryDoneSound.manager.ts, PromptQueryDoneSound.manager.test.ts, e "Nettoyage son Prompt", e "Son fin query", AiQuery.type.ts, e "AiQuery type checks", e "Specif retry sémantique", e "Compact retry sémantique", usePromptRunCommands.hook.ts [L100-L320], abort.endpoint.ts, open.endpoint.ts, stream.endpoint.ts, OrchAskStreamAbort.ts [L1-L140], e "Abort stream exact", e "Active controllers thread", OrchAskStreamWiring.ts [L1-L90], OrchAskStream.ts [L200-L320], e "Rendu banner live", e "Rendu showComposer", e "Rendu viewer banner", e "Mode viewer assigné", PromptInput.tsx [L340-L390], PromptInput.tsx [L1830-L2070], usePromptRunController.finalization.hook.ts, e "Finalisation stream Prompt", e "Set streaming false", e "fileEditsUiFinalized", e "Implémentation onStreamed", e "Émission stream extension", h "Cycle bus stream",
extTypedMessageBus.util.ts,
useStreamListener.hook.ts, frontMessageBus.util.ts, extMessageBus.util.ts [L220-L330], extMessageBus.util.ts [L330-L390], extMessageBus.util.ts [L100-L220], extMessageBus.util.ts [L1-L110], OrchMemory.ts [L430-L690], OrchMessagesMemory.ts [L100-L180], OrchMessagesMemoryStore.ts [L1-L90], OrchMessagesMemoryStore.ts [L200-L260],
ChatSessionFile.type.ts, ChatPersistence.service.ts, ChatPersistenceWriteQueue.service.ts, e "Queue session lecture", e "Queue session écriture", e "Erreur premier chunk", e "Stream ended before header", e "Erreur stream synthétique", callLocalApi.util.ts, h "Client Slimbk stream", stream.endpoint.test.ts [L1-L220], delete.endpoint.ts, g "Tests suppression chat", EasyAgentsBatchRunManager.service.ts [L220-L280], NewChat.command.ts, uiGenericHandlers.ts, registerDeepLinkUriHandler.ts [L100-L170], openPromptInNewSession.util.ts, openSidebarChatItem.util.ts, h "Queues FIFO par clé", SerialTaskQueue.ts, e "Usage SerialTaskQueue", e "Fabrique meta error", AiStreamedChunk.type.ts [L1-L260], aiStopSuffix.util.ts, g "Tests suffix stream",
usePromptQueryDoneSoundSync.hook.ts,
EasyAgentsBatchLastViewerStop.service.ts, EasyAgentsBatchRunManager.types.ts [L1-L240], EasyAgentsBatchRunManager.service.ts [L1-L230], e "Child executor construction", EasyAgentsBatchRunExecution.service.ts [L300-L420], e "Préservation promptDraft", OrchMemoryPersistenceQueue.ts [L80-L220], OrchMemoryPersistenceQueue.ts [L220-L430], OrchMemoryPersistenceQueue.ts [L650-L830], ChatPersistencePromptDraft.service.ts, ChatPersistenceWriter.service.ts [L1-L240], frontSessionToPromptMessages.util.ts [L220-L340], e "Miroir thinking stream", e "setStreamingThinkingTokens", e "Thinking separator stream", usePromptThinkingLifecycle.hook.ts [L1-L280], e "Hook thinking controller", usePromptRunController.hook.ts [L130-L300], usePromptRunController.hook.ts [L300-L440], e "Base assistant miroir", usePromptToolUse.hook.ts [L1-L180], g "Tests thinking lifecycle", e "Tests thinking mirror", usePromptCancelController.hook.ts [L90-L180], VscLink.tsx, usePromptSessionLocator.hook.ts [L1-L180]

specifying "SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260711_235909"

SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260711_235909

Copy

Refine

Code

Verify

Markdown formattedMarkdown source

Spécification — Finaliser `liveSession` pour les sessions Prompt normales

Contexte

La tranche 1 est exactement l'état courant du dépôt, au commit `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`. Elle a extrait le live watcher EasyAgents Batch dans un mécanisme générique `liveSession` réparti entre les contrats shared, le backend Orch et le frontend Prompt. 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 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 doit rouvrir 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 a confirmé l'architecture générale de la spécification initiale, mais a aussi mis en évidence plusieurs invariants indispensables qui doivent être explicitement intégrés au plan final :

- 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 ;
- les callbacks frontend d'un stream doivent être gardés par `sourceId`, car `turnId` seul ne permet pas de rejeter une complétion ou un frame tardif d'une invocation précédente du même tour ;
- 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 ne constitue pas un terminal source ;
- `continueResponse` doit réutiliser la bulle thinking agrégée existante et son texte de base, faute de quoi le streaming/replay produit deux bulles alors que l'hydratation classique en produit une seule ;
- les erreurs de source transformées en header Slimbk doivent être canonisées de manière identique dans le buffer et dans le frontend continu, faute de quoi l'isomorphisme diverge ;
- 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 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 exactement du commit `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33` et conserve intégralement le comportement EasyAgents visible.
- `LiveSessionRunRef` devient obligatoirement `{ locator, turnId, sourceId }`. Le `turnId` reste la corrélation logique persistante des messages ; le `sourceId` identifie une invocation précise du transport.
- Le `sourceId` est obligatoire dans toutes les routes/query strings liveSession et dans les guards frontend. 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. L'état actif interne, les callbacks `onData/onComplete/onError`, le Stop et le transport sont tous liés à ce même `sourceId`.
- Les callbacks d'une ancienne source dont le `sourceId` ne correspond plus à la source active sont ignorés, même lorsque le `turnId` est identique. Un ancien `onComplete` ne peut donc pas arrêter l'état streaming d'une source 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 terminaux évincés une métadonnée de retraite exacte avec l'époque locator terminale. Un fallback terminal persistant n'est accepté que si cette métadonnée existe et qu'aucune autre réservation/source du locator n'a fait évoluer l'époque depuis ce terminal.
- Une réservation annulée, un run pré-baseline évincé et une éviction destructive 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 un snapshot baseline immuable avant `askStream(...)`. Le snapshot reprend l'enveloppe persistée courante, y compris brouillon et attachments, et remplace son tableau de messages par le snapshot défensif courant d'OrchMemory afin de conserver exactement l'assistant ciblé.
- Cette capture s'exécute dans la sérialisation de persistance du locator afin d'attendre les écritures de brouillon déjà en file et d'éviter un read/write incohérent.
- `markBaselineReady` accepte le snapshot seulement pour `continueAssistant`; il est interdit pour `replaceTurnAssistants` et pour EasyAgents. Le registre clone défensivement le snapshot 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 : retour `askStream` → première lecture pendante → source-start synchrone → traitement du premier chunk → 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.
- Chaque chunk normal exposé au frontend est appendu au registre avant l'exposition. L'exception de framing du premier `meta.error` reste compatible avec le header Slimbk : le buffer reçoit alors le chunk d'erreur canonique que le frontend continu reconstruit depuis l'erreur transport, pas une représentation différente.
- 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.
- La construction canonique des erreurs transport est partagée entre backend et frontend ; elle normalise le message transport et produit la même valeur `AiErrorMetaValue` dans les deux chemins.
- 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 ne transporte pas d'options auto-run incompatibles.
- Le coordinateur est une classe à dépendances injectées et n'importe jamais le module low-level qui le compose. L'instance de production est créée dans le wiring du panneau avec une fonction low-level injectée, ce qui interdit le cycle ESM `coordinator ⇔ openPanel`.
- 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. Les queues sont 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 la source active exacte juste avant tout `ClearMemory`. Un settlement tardif d'une ancienne source ne peut jamais nettoyer la mémoire 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 et annulable jusqu'à `live`, `terminal`, `not_found` ou unmount.
- 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. Une erreur du transport replay ne compte pas comme terminal source et déclenche la branche de récupération, jamais un faux passage à `normal`.
- 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.
- 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`.
- 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` et le séparateur canonique inter-message garantissent que streaming continu, 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, réservations et machine d'état

`LiveSessionRunRef` devient `{ locator, turnId, sourceId }`. Le run key structurel inclut les trois valeurs. 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 de démarrage de la source ;
- le `replayMode` et, uniquement pour la continuation, un snapshot baseline immuable.

Une source normale suit :

1. création du `sourceId` frontend après acceptation du start ;
2. message bus vers l'extension ;
3. réservation exacte ownership `source` comme première mutation synchrone ;
4. publication Activité Running/source avec l'état réel du panneau ;
5. persistance éventuelle de la préférence SPECIF Code ;
6. chargement du catalogue ;
7. capture sérialisée de la baseline `continueResponse` si nécessaire ;
8. retour réussi de `askStream(...)` ;
9. priming du `Readable` lazy ;
10. transformation réservation → record et application immédiate d'un abort précoce ;
11. append de chaque chunk canonique avant exposition ;
12. baseline durable ;
13. replay possible ;
14. source settled ;
15. settlement + handoff normal atomique si une baseline existe, sinon éviction non terminale ;
16. transition Activité Idle si panneau ouvert, ou suppression si panneau fermé ;

17. cleanup mémoire/attachments/son/trivial chat 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 snapshot baseline immuable conserve l’assistant existant ; le frontend réutilise sa bulle assistant et sa bulle thinking agrégée, puis applique les nouveaux tokens/notices sur leurs textes de base.

Le premier `meta.error` converti en header Slimbk et les erreurs de transport suivent une fabrique canonique partagée. Ainsi :

- le frontend continu reçoit un `meta.error` synthétique déterministe ;
- le buffer liveSession contient exactement le même chunk logique ;
- un replay ultérieur produit le même texte, le même suffixe et les mêmes métadonnées ;
- un EOF sans terminal devient `INTERRUPTED` au lieu de laisser un replay bloqué ou un faux handoff.

Cycle de vie du panneau sans cycle de modules

La classe coordinateur ne connaît qu’un port low-level injecté. 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.

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

Une fermeture EasyAgents retire immédiatement l’activité observer du panneau et conserve la policy « dernier viewer = hard stop ». Elle ne crée jamais un item détaché source.

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 `isViewerMode=false` ;
- 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.

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 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 afin que les tests frontend n’utilisent pas les labels visibles comme sélecteurs.

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 le snapshot de base et la reconstruction frontend de chaque branche.

2) LiveSessionRun.type.ts — modifié

- Ajouter un 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) 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 ; ne pas dupliquer sa validation.
- Conserver les close policies et rendre `continueOnPanelClose` effectif pour `normalPrompt`.

4) 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) LiveSessionContracts.test.ts — modifié

- Couvrir la clé locator+turn+source, les collisions, les 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) aiStopSuffix.util.ts — modifié

- Ajouter une fabrique canonique de `meta.error` pour les erreurs de transport/source et une normalisation déterministe des détails.
- Produire `SYSTEM_ERROR` ou `INTERRUPTED` avec les messages canoniques existants.
- Garantir que le backend et le frontend construisent byte-for-byte le même chunk à partir du même message utile.

7) aiStopSuffix.util.test.ts — modifié

- Tester Error/string/status Slimbk, préfixes transport, détails, SYSTEM_ERROR, INTERRUPTED et stabilité du chunk produit.

8) 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) SidebarUiMessages.ts — modifié

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

10) MessageRegistry.ts — modifié

- Ajouter `sourceId` à `proxylocalapi:aiStream`.
- Ajouter `sourceId` à `proxylocalapi:liveSessionStream`.

Backend Orch `liveSession`

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

- Documenter sourceId, réservations, readiness, replay modes, snapshot continuation, canonicalisation des erreurs, ownership source/observer, lifecycle partagé, panel open/closed, Activity, direct handoff, tombstones et claims Stop.
- Ajouter les machines d'état et les races de sources successives/panel/cleanup.

12) LiveSessionRecord.type.ts — modifié

- Ajouter `baselineOnly` à `LiveSessionReadiness`.
- Ajouter `replayMode`, `baselineSession`, `sourceStartedAtMs`, `abortRequested` et `viewerGroupId?`.
- Définir la réservation source, le snapshot public actif/réservé et la retraite terminale/non terminale avec époque.
- Étendre permits/subscriptions à l'identité sourceId.

13) LiveSessionRegistry.service.ts — modifié

- Ajouter réservation/cancel exacts pour ownership source ; la réservation doit être synchrone et unique par locator.
- Exiger une réservation correspondante pour enregistrer une source normale ; conserver l'enregistrement direct EasyAgents.
- Transformer atomiquement la réservation en record et retourner l'état `abortRequested` au caller.

- Étendre readiness et baseline snapshot avec validations de replayMode.
- Ajouter des snapshots immuables `getActiveSourceRun(locator)` et `listActiveSourceRuns()` couvrant réservations et records source.
- Ajouter un test exact et un test locator pour `hasActiveOrReservedSource`.
- Ajouter les claims atomiques/idempotents viewer et source ; tous partagent `abortRequested`.
- Remplacer le simple set de retired keys par une retraite structurée ; autoriser le fallback persistant uniquement pour une retraite terminale dont l'époque est toujours courante.
- Faire évoluer l'époque sur reserve/register/cancel/handoff/evict afin de bloquer toutes les lectures ABA et les anciennes sources du même turn.
- Préserver toutes les garanties de leases, retention et replay de tranche 1.

14) LiveSessionRegistry.service.test.ts — modifié

- Couvrir sources successives même turn/sourceId distinct, reservation→record, cancel, conflict, baselineOnly, snapshot continuation, abort pré/post-registration, source listing, claims concurrents, tombstones et époques.
- Rejouer toutes les garanties tranche 1 avec sourceId explicite.

15) LiveSessionService.ts — modifié

- Exposer reserve/cancel, résolution/listing source, tests d'existence exact/locator, baseline snapshot, claims abort et settlement+handoff normal atomique.
- Conserver Open/Replay/release/sweep/evict.

16) LiveSessionService.test.ts — modifié

- Couvrir le parcours normal complet, les réservations, la réouverture avant/après baseline, sources successives, claims et settlement sans baseline.

17) LiveSessionOpen.service.ts — modifié

- Retourner `not_ready` immédiatement pour une réservation exacte.
- Pour `replaceTurnAssistants`, conserver le filtrage exact des assistants du tour.
- Pour `continueAssistant`, retourner une copie du snapshot baseline stocké sans supprimer l'assistant cible.
- Retourner sourceId et replayMode.
- Ne lire un fallback terminal absent-record que si une retraite terminale exacte et son époque l'autorisent.

18) LiveSessionOpen.service.test.ts — modifié

- Couvrir réservation, baselineOnly, snapshot continuation immuable, source mismatch, deux sources du même turn, fallbacks terminaux avant/après nouvelle source, retrait non terminal et toutes les courses existantes.

19) 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.

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.
- 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.

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

- Mapper chaque AiQuery vers replayMode ; readiness reste `baselineOnly`.
- Capturer la baseline continue sous la queue de persistance, avec enveloppe persistée et messages OrchMemory défensifs.
- Enregistrer ownership source après priming ; appliquer immédiatement un abort déjà réclamé.
- Parser les chunks, les canoniser lorsque le framing transport l'exige, les append avant exposition, marquer fallback baseline et suivre le vrai terminal.
- Transformer les throws/EOF sans terminal en erreur canonique bufferisée/exposée.
- Settlement+handoff atomique après consommation ; pas de handoff si aucune baseline n'a jamais existé.
- Exposer au endpoint un résultat/event strict permettant de conserver le header 500 du premier meta.error sans divergence du buffer.

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

- Couvrir prompt/retry/edit/specif/compact/continue, capture baseline, ordre append-before-yield, même turn/source différente, abort réservé, premier meta.error/header, throw après baseline, EOF sans terminal, settlement et handoff.

Adaptation EasyAgents

24) **EasyAgentsBatchLiveSessionAdapter.service.ts** **— modifié**

- Enregistrer `replayMode='replaceTurnAssistants'` et `sourceId=turnId`.
- Injecter le lifecycle générique dans le wiring Batch sans modifier la policy produit.

25) **EasyAgentsBatchLiveSessionAdapter.service.test.ts** **— modifié**

- Vérifier sourceId=turnId, replayMode et comportement historique.

26) **EasyAgentsBatchRunExecution.service.ts** **— modifié**

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

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

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

28) **EasyAgentsBatchRunChildExecution.service.ts** **— modifié**

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

29) **EasyAgentsBatchRunChildExecution.service.test.ts** **— modifié**

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

30) **EasyAgentsBatchRunManager.service.ts** **— modifié**

- Injecter le lifecycle partagé dans le child executor.
- Remplacer la lecture `queryState.windows` par `items`.
- Pour le guard de Retry manuel, consulter aussi l'autorité `liveSession.hasActiveOrReservedSourceForLocator(locator)` afin qu'une source détachée ou une projection Sidebar en retard ne puisse jamais être supprimée.

31) **EasyAgentsBatchRunExecution.testHarness.ts** **— modifié**

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

32) **EasyAgentsBatchRunLifecycle.service.test.ts** **— modifié**

- Adapter la construction du child executor au lifecycle partagé.

33) **EasyAgentsBatchLastViewerStop.service.test.ts** **— modifié**

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

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

- Documenter sourceId=turnId, lifecycle partagé et la branche Activity observer/open uniquement.
- Conserver la policy 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 nouveaux champs, branches et source mismatch.

37) **stream.endpoint.ts** **— modifié**

- Exiger sourceId et relayer le run exact complet.

38) **stream.endpoint.test.ts** **— modifié**

- Couvrir sourceId exact, mismatch et replay terminal.

39) **abort.endpoint.ts** **— modifié**

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

40) **abort.endpoint.test.ts** **— modifié**

- Couvrir viewer EasyAgents, viewer Prompt normal, source stale, 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) query ask stream.endpoint.ts — modifié

- Ajouter sourceId au schéma.
- Conserver la persistance SPECIF Code avant le catalogue.
- Déléguer création/consommation/instrumentation à `NormalPromptLiveSessionSourceService`.
- Conserver le header Slimbk du premier meta.error en utilisant l'événement canonique fourni par le service ; ne jamais bufferiser une représentation différente de celle du frontend.

44) query ask stream.endpoint.test.ts — modifié

- Tester délégation, sourceId, ordre SPECIF, premier chunk normal, premier meta.error/header, absence de double forwarding et erreurs pré-source.

45) liveSessionStreamHandlers.ts — modifié

- Relayer sourceId.
- Ownership source : aucune activité/son supplémentaire.
- Ownership observer : activité uniquement si le panneau exact est encore ouvert ; jamais de branche observer/closed.
- Préserver l'ordre des chunks et les cleanups partiels.

46) liveSessionStreamHandlers.test.ts — modifié

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

47) iaStreamHandlers.ts — modifié

- Recevoir le sourceId frontend.
- Réserver synchroniquement le run avant tout await.
- Démarrer l'activité source avec runRef exact et panelState réel, puis démarrer le son.
- Transmettre sourceId à l'endpoint.
- En finally : son stop, activité stop exacte, cancel réservation idempotent, puis coordinateur `onSourceSettled(runRef)`.
- Préserver les erreurs primaires tout en exécutant tous les cleanups.

48) iaStreamHandlers.queryDoneSound.test.ts — modifié

- Vérifier que fermer le panneau source ne supprime pas l'état son avant terminal, et qu'un replay source-owned ne joue rien.

49) iaStreamHandlers.runIdPropagation.test.ts — modifié

- Étendre runRef à locator+turn+source, vérifier réservation avant await, Activity exacte, cancel idempotent et settlement coordonné.

Activité runtime extension

50) SidebarQueryRunningBridge.ts — modifié

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

51) SidebarQueryRunningBridge.race.test.ts — modifié

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

52) SidebarQueryRunningBridge.usage.test.ts — modifié

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

Coordination panneau Prompt

53) `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 la query normalPrompt exacte avec lease fraîche.
- Distinguer : query classique, live explicite valide, live explicite invalide.
- Préserver l'ordre déterministe et interdire toute dégradation silencieuse.

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

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

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

- Classe OOP à dépendances injectées ; elle n'importe jamais le low-level opener.
- Map locatorKey → `SerialTaskQueue`, nettoyée lorsque la queue est vide.
- `openPromptPanel` : dans la queue, révéler l'existant ; sinon préserver un live explicite ; sinon résoudre une source normale active/réservée et injecter la query ; sinon ouvrir classique.
- `onPanelDisposed` : libérer la lease du descripteur initial, réévaluer la source active actuelle, appliquer la transition Activity, détacher le mapping exact et choisir cleanup différé ou historique.

- Associer le descriptor de cleanup détaché au runRef exact qui était actif au moment de la fermeture.
- `onSourceSettled` : ignorer un settlement stale si une nouvelle source exacte est active ; si panneau ouvert, supprimer l'état détaché ; sinon cleanup terminal sans abort.
- Réévaluer panneau/source immédiatement avant ClearMemory.

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

- Couvrir priorité live explicite EasyAgents, normal auto-injecté, query invalide non dégradée, absence de cycle via injection, open/cleanup dans les deux ordres, close pré-baseline, settlement stale après nouvelle source, cleanup gagné/perdu et aucune mémoire effacée sous un panneau ouvert.

57) `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 puis `abortStream` sans await intermédiaire.
- Marquer `stopRequested` par runRef exact et retourner `{accepted}`.

58) `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.

59) PromptThreadCleanup.manager.ts — modifié

- Remplacer la map sessionId par une map locator structurelle.
- Exposer des opérations distinctes et testables : register, detach sans abort, cleanup destructif historique, cleanup terminal déjà settled sans abort/await.
- Partager le cleanup orphan attachments.
- Ne jamais Abort/ClearMemory une source normale active lors du dispose.

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

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

61) openPanel.util.ts — modifié

- Composer l'instance coordinateur avec un port low-level injecté ; exposer une fonction publique async coordonnée pour Prompt.
- Conserver le low-level create/reveal et les pages non-Prompt.
- Le low-level Prompt capture le descriptor initial uniquement pour le transmettre au dispose.
- Sur dispose : retirer synchroniquement le panneau du registry, puis déléguer tous les effets Prompt au coordinateur ; conserver seulement le bookkeeping générique sélection/readyViews.
- Adapter le préfixe tab Running à `queryState.items/status`.

62) openPanel.util.test.ts — modifié

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

63) uiGenericHandlers.ts — modifié

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

64) uiGenericHandlers.openExternalUrl.test.ts — modifié

- Mocker la façade Prompt et préserver les tests URL.

65) NewChat.command.ts — modifié

- Await l'ouverture coordonnée de la nouvelle session.

66) NewChat.command.test.ts — modifié

- Vérifier l'appel coordonné async et la propagation d'erreur.

67) registerDeepLinkUriHandler.ts — modifié

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

68) registerDeepLinkUriHandler.scope.test.ts — modifié

- Vérifier ouverture coordonnée root/nested et préserver les guards de scope.

SidebarStore et Activité

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

- Remplacer add/running génériques par transitions atomiques panelOpened, sourceStarted et sourceSettled.
- Respecter les trois branches strictes de l'union.
- Conserver les listeners title/running des tabs Prompt.

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

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

- Conserver rename ephemeral.

71) SidebarStore.core.manager.ts — modifié

- Initialiser `queryState.items`.
- Appeler la nouvelle réhydratation des activités.

72) SidebarStore.rehydrate.manager.ts — modifié

- Renommer en `rehydrateQueryActivities`.
- Fusionner panneaux ouverts et sources ownership source actives/réservées sans doublons.
- Si locator ouvert+source : Running/source/open ; source seule : Running/source/closed ; panneau seul : Idle/open.
- Ne jamais projeter observer/closed ni closed/idle.
- Calculer elapsed depuis sourceStartedAtMs.

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

- Couvrir idle ouvert, source ouverte, source fermée, réservation, observer sans panneau ignoré et déduplication.

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

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

75) SidebarStore.chatTitles.manager.ts — modifié

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

76) SidebarStore.handlers.manager.ts — modifié

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

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

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

78) delete.endpoint.ts — modifié

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

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

- Couvrir source ouverte, source fermée, réservation, store stale, source absente et absence de side effects lorsque la suppression est bloquée.

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

- Adapter les fixtures `items`.

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

- Adapter les fixtures `items` et conserver les scénarios delete disque.

82) ChatPersistenceWriter.windowsFallback.test.ts — modifié

- Adapter la fixture SidebarStore à `items`.

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

- Adapter la fixture SidebarStore à `items`.

84) SidebarActivityQuery.tsx — modifié

- Utiliser `items` et l'union stricte.
- Ouvrir par `sidebar->ext:openQueryActivity` avec locator seulement.
- Ajouter callback Stop par runRef exact.
- Rendre la troisième ligne seulement pour Running/source/closed.
- Utiliser SquareStop/small/danger/disableAfterClick, `enabled={!stopRequested}` et testId sourceId.
- Ajouter un testId stable au lien d'ouverture.
- Toutes les fonctions/objets React modifiés utilisent useCallback/useMemo.

85) SidebarActivityQuery.test.tsx — modifié

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

Frontend `liveSession` et Prompt

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

- Parser adapter normal, sourceId, lease et continueOnPanelClose.
- Définir policies : retry not_ready, direct handoff, replay retry sur erreur transport, surface interactive pendant streaming.
- Streaming : view model non-viewer, aucune action Stop banner.
- Failure transport : surface de récupération explicite, sans faux handoff normal.

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

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

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

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

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

- Couvrir dispatch, configurations partielles et zéro fallback.

90) **PromptLiveSession.types.ts** **— modifié**

- Ajouter policies not-ready, handoff, transport failure et surface interactive.
- Supprimer l'inférence mode!=normal⇒viewer et le helper mort associé.
- Étendre bootstrap/runRef à sourceId.

91) **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 à l'unmount.
- Démarrer replay selon replayMode.
- Normal direct handoff uniquement après terminal reçu via data + queues drainées + complétion transport normale.
- Sur erreur transport replay normal : ne pas passer à normal ; exposer la récupération et permettre une reprise exacte depuis Open/baseline.
- EasyAgents conserve Open terminal/hydratation/Retry.

92) **usePromptLiveSession.hydration.test.tsx** **— modifié**

- Ajouter sourceId/replayMode et conserver tous les scénarios EasyAgents.

93) **usePromptLiveSession.handoff.test.tsx** **— modifié**

- Vérifier EasyAgents terminal Open, normal sans Open terminal, et refus du direct handoff sur transport error.

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

- Couvrir not-ready retry, surface interactive, draft/attachments, Stop normal, transport failure/retry, direct handoff, bootstrap terminal et close/reopen.

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

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

96) **EasyAgentsBatchLiveSession.adapter.ts** **— modifié**

- Émettre sourceId=turnId et adapter easyAgentsBatch.
- Parser nouvelle identité sans changer labels/UX.

97) **EasyAgentsBatchLiveSession.adapter.test.ts** **— modifié**

- Adapter ordre query, sourceId et dispatcher.

98) **AiStreamStartArgs.type.ts** **— modifié**

- Garder l'argument public promptQuery sans sourceId.
- Définir un état actif interne strict dont chaque branche transporte un runRef complet.
- La branche observer reçoit directement le runRef exact.

99) **useAiStream.hook.ts** **— modifié**

- Générer sourceId après le guard d'acceptation du promptQuery.
- Conserver currentSourceId synchroniquement et lier chaque callback de channel au sourceId capturé.
- Ignorer onData/onComplete/onError stale même turn/source différente.
- Exposer un outcome transport strict `pending|completed|transportError`.
- Stop promptQuery via abortSource exact ; Stop observer via abort viewer exact.
- Utiliser la fabrique canonique d'erreur transport.
- Ne modifier ni Queue A, ni Queue B, ni locks, ni sanitizers, ni raw-live routing.

100) **useAiStream.hook.liveSessionObserver.test.ts** **— modifié**

- Ajouter runRef sourceId, routes Stop exactes, outcome transport et rejet de callbacks stale.

101) `AICode/src/frontend/src/messageBus/__test__/useAiStream.hook.promptSourceIdentity.test.ts` **— créé**

- Vérifier sourceId unique par start accepté, stabilité, transport, abort exact, stale data/completion/error et même turn réutilisé.

102) usePromptRunController.hook.ts — modifié

- `startLiveObserver` reçoit runRef+lease+replayMode.
- Ajouter `baseThinkingTextRef` et transmettre les guards de continuation.
- Exposer l'outcome transport à `Prompt`/liveSession.
- Utiliser le bootstrap replace ou continuation approprié.

103) usePromptRunController.liveSessionObserver.test.tsx — modifié

- Couvrir sourceId, replay modes, base thinking et outcome transport.

Continuation et thinking isomorphes

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

- Trouver l'assistant message exact du turn demandé ; fail-fast s'il manque.
- Capturer base assistant text et base thinking text.
- Réutiliser la bulle thinking existante du turn ou créer un placeholder unique avant l'assistant.
- Réutiliser l'UID assistant hydraté.
- Armer le séparateur canonique uniquement si du thinking historique existe.
- Réinitialiser buffers, refs, notices, guards et `fileEditsUiFinalized` sans cast inutile.

105) `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.

106) preparePromptStreamingTurn.util.ts — modifié

- Réinitialiser explicitement base thinking et pending separator pour les branches replaceTurnAssistants.
- Conserver le contrat historique des nouveaux turns.

107) preparePromptStreamingTurn.util.test.ts — modifié

- Vérifier reset des bases assistant/thinking et du separator.

108) usePromptThinkingLifecycle.hook.ts — modifié

- Initialiser le transcript thinking depuis `baseThinkingTextRef` lorsque la cible de stream change.
- Ajouter le separator inter-message une seule fois avant le premier nouveau thinking.
- Ne pas supprimer une bulle contenant du thinking historique lorsque le segment courant commence par texte/tool.
- Réinitialiser proprement pour les turns replace.

109) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptThinkingLifecycle.continuation.test.tsx` — créé

- Couvrir base thinking conservée sans nouveaux tokens, append avec separator, tool/text avant thinking, réinsertion et reset nouveau turn.

110) usePromptToolUse.hook.ts — modifié

- Appliquer la suppression de bulle vide uniquement lorsque `baseThinkingTextRef` est vide.
- Préserver le thinking historique pendant les tool calls de continuation.

111) usePromptRunController.finalization.hook.ts — modifié

- Réinitialiser base thinking et pending separator après finalisation terminale, y compris meta-only.
- Conserver l'ordre actuel de finalisation et l'attachement des métadonnées.

112) usePromptRunCommands.hook.ts — modifié

- Remplacer le bootstrap inline continueResponse par le helper partagé.
- Conserver Retry/Edit/Send/Specif inchangés hors passage des nouveaux refs.

113) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.continueResponse.test.tsx` — créé

- Vérifier helper partagé, même turn logique, nouvelle sourceId générée par useAiStream, thinking unique et assistant cible exact.

Intégration Prompt et documentation

114) Prompt.tsx — modifié

- Capturer sourceId/adaptateur dans la query figée et utiliser le dispatcher.
- Passer outcome transport au hook liveSession.
- Déduire `interactionsEnabled` de la surface/policy, pas du mode seul.
- Autoriser draft persistence et interactions normales pendant replay normalPrompt.
- Garder `ownsThreadForCleanup=false` jusqu'au direct handoff normal ; ensuite réenregistrer le thread.
- Préserver classic hydration off pour tout live bootstrap.

115) Prompt.liveSessionBootstrap.test.tsx — modifié

- Couvrir dispatch, sourceId, surface interactive, draft, query figée, hydration off, transport failure et ownership cleanup.

116) 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, pas les labels visibles.

117) EasyAgentsBatch.liveWatch.test.tsx — modifié

- Adapter query sourceId/adaptateur sans changement UX.

118) EasyAgentBatchSurface.test.tsx — modifié

- Adapter query live exacte sans changement UX.

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

- Ajouter fermeture/réouverture comme troisième chemin isomorphe.
- Documenter continuation assistant+thinking, erreurs transport canoniques et vrai terminal requis pour direct handoff.

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

- Documenter que les règles de suppression concernent le placeholder vide du segment courant et ne suppriment jamais le thinking historique d'une continuation.
- Documenter la bulle agrégée unique et le separator canonique entre segments assistant.

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

- LiveSessionRun.type.ts
- LiveSessionPanel.type.ts
- LiveSessionOpenResult.type.ts
- LiveSessionContracts.test.ts
- aiStopSuffix.util.ts
- aiStopSuffix.util.test.ts
- SidebarStore.ts
- SidebarUiMessages.ts
- MessageRegistry.ts
- liveSession.programming-doc-readme.md
- LiveSessionRecord.type.ts
- LiveSessionRegistry.service.ts
- LiveSessionRegistry.service.test.ts
- LiveSessionService.ts
- LiveSessionService.test.ts
- LiveSessionOpen.service.ts
- LiveSessionOpen.service.test.ts
- LiveSessionReplay.service.test.ts
- EasyAgentsBatchLiveSessionAdapter.service.ts
- EasyAgentsBatchLiveSessionAdapter.service.test.ts
- EasyAgentsBatchRunExecution.service.ts
- EasyAgentsBatchRunExecution.live-activity-and-gating.service.test.ts
- EasyAgentsBatchRunChildExecution.service.ts
- EasyAgentsBatchRunChildExecution.service.test.ts
- EasyAgentsBatchRunManager.service.ts
- EasyAgentsBatchRunExecution.testHarness.ts
- EasyAgentsBatchRunLifecycle.service.test.ts
- EasyAgentsBatchLastViewerStop.service.test.ts
- easyAgents.programming-doc-readme.md
- open.endpoint.ts
- open.endpoint.test.ts
- stream.endpoint.ts
- stream.endpoint.test.ts
- abort.endpoint.ts
- abort.endpoint.test.ts
- query ask stream.endpoint.ts
- query ask stream.endpoint.test.ts
- liveSessionStreamHandlers.ts
- liveSessionStreamHandlers.test.ts
- iaStreamHandlers.ts
- iaStreamHandlers.queryDoneSound.test.ts

- iaStreamHandlers.runIdPropagation.test.ts
- SidebarQueryRunningBridge.ts
- SidebarQueryRunningBridge.race.test.ts
- SidebarQueryRunningBridge.usage.test.ts
- PromptThreadCleanup.manager.ts
- PromptThreadCleanup.manager.attachmentsCleanup.test.ts
- openPanel.util.ts
- openPanel.util.test.ts
- uiGenericHandlers.ts
- uiGenericHandlers.openExternalUrl.test.ts
- NewChat.command.ts
- NewChat.command.test.ts
- registerDeepLinkUriHandler.ts
- registerDeepLinkUriHandler.scope.test.ts
- SidebarStore.queryWindow.listeners.manager.ts
- SidebarStore.queryWindow.mutations.manager.ts
- SidebarStore.core.manager.ts
- SidebarStore.rehydrate.manager.ts
- SidebarStore.rehydrate.manager.test.ts
- SidebarStore.manager.queryRunningUpsert.test.ts
- SidebarStore.chatTitles.manager.ts
- SidebarStore.handlers.manager.ts
- delete.endpoint.ts
- list.endpoint.windowsLock.eperm.test.ts
- diskAliases.lastWins.endpoints.test.ts
- ChatPersistenceWriter.windowsFallback.test.ts
- OrchMemory.naive_title.user_prompt_specif_multiline.e2e.repro.test.ts
- SidebarActivityQuery.tsx
- SidebarActivityQuery.test.tsx
- PromptLiveSession.types.ts
- usePromptLiveSession.hook.ts
- usePromptLiveSession.hydration.test.tsx
- usePromptLiveSession.handoff.test.tsx
- EasyAgentsBatchLiveSession.adapter.ts
- EasyAgentsBatchLiveSession.adapter.test.ts
- AiStreamStartArgs.type.ts
- useAiStream.hook.ts
- useAiStream.hook.liveSessionObserver.test.ts
- usePromptRunController.hook.ts
- usePromptRunController.liveSessionObserver.test.tsx
- preparePromptStreamingTurn.util.ts
- preparePromptStreamingTurn.util.test.ts
- usePromptThinkingLifecycle.hook.ts
- usePromptToolUse.hook.ts
- usePromptRunController.finalization.hook.ts
- usePromptRunCommands.hook.ts
- Prompt.tsx
- Prompt.liveSessionBootstrap.test.tsx
- PromptInput.batchLiveViewerMode.test.tsx
- EasyAgentsBatch.liveWatch.test.tsx
- EasyAgentBatchSurface.test.tsx
- 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

Déplacés
Aucun.
Supprimés
Aucun.
Facteur de risque de régression

1. **Risque calculé : 89 %.** Criticité `5`, couplage `5`, blast radius `5`, incertitude `4`, faible confiance `2`. Base = `10×5 + 3×5 + 2×5 + 2×4 + 3×2 = 89`; risque = `round(89 × 5/5) = 89 %`.
2. **Zones complexes :** sources successives partageant un turn, réservations avant tout await, callbacks frontend stale, lifecycle Readable lazy, premier header d'erreur, EOF sans terminal, baseline `continueResponse`, thinking agrégé, races panel open/close/cleanup, Activity source/observer, direct handoff, conservation draft/attachments, claims Stop pré-enregistrement, suppression autoritative, tombstones après source ultérieure et coexistence EasyAgents/Prompt normal.
3. **Justification :** la fondation tranche 1 est solide et fortement testée, mais cette tranche modifie le chemin normal de toutes les requêtes Prompt, la politique historique de dispose et la projection Activité. Une erreur peut perdre/dupliquer un chunk, accepter une complétion stale, clear la mémoire d'une nouvelle source, interrompre le mauvais run, afficher un Stop EasyAgents détaché, produire deux bulles thinking ou violer l'isomorphisme. Les identités exactes, réservations, epochs, ownerships stricts, queues par locator, canonisation des erreurs et tests à trois chemins sont donc obligatoires.

Résumé

- Étendre `liveSession` aux sources Prompt normales en une seule implémentation complète.
- Ajouter `sourceId` de bout en bout et l'utiliser aussi pour garder les callbacks frontend contre les frames/completions tardifs.
- Réserver synchroniquement puis enregistrer les sources normales, bufferiser avant forwarding, et partager le lifecycle lazy avec EasyAgents.
- Capturer une baseline immuable pour `continueResponse`, avec assistant et thinking de base strictement isomorphes.
- Canoniser les erreurs/EOF afin que stream continu et replay reçoivent le même terminal.
- Ne plus tuer une source normale active à la fermeture du panneau ; différer cleanup mémoire/attachments/son.
- Sérialiser ouverture, dispose et cleanup terminal par locator sans cycle de modules.
- Préserver les queries live explicites EasyAgents et injecter uniquement les queries normalPrompt automatiques nécessaires.
- Modéliser Activité avec les branches Idle, Running observer/open et Running source/open|closed.
- Réutiliser le Stop normal dans Prompt et ajouter uniquement le Stop détaché Running/source/closed.
- Utiliser le registre liveSession comme autorité pour deletion/Retry destructifs.
- Préserver intégralement l'UX EasyAgents et les algorithmes Queue A/Queue B.
- Tester les unités, les courses et l'isomorphisme stream continu / replay / hydratation classique.

Nom du commit
`feat(live-session): keep prompt runs alive across panel close and reopen`
SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260711_235909
Copy
Refine
Code
Verify

[... response truncated; reason: aborted by user]
Generation aborted by user.
Restart response
</ASSISTANT>
SHOW ALLPage 1 / 1