Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=e452b8bcc54e49c7abbf603ec5d4e525-RS-Optimiser-session
<USER>
[Refine]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_022043
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *reading project*: 🗂️ g *"Docs liveSession proches"*, 🗂️ t *"Architecture liveSession"*, 🗂️ t *"Frontend liveSession"*, 🗂️ t *"Panneaux Prompt backend"*
🔧 ✅ *scanning git*: 🌱 *"État et commit courant"*, 🌱 *"Historique récent"*
🔧 ✅ *reading project*: 🔎 h *"Architecture liveSession"*, 📄 *liveSession.programming-doc-readme.md*, 📄 *easyAgents.programming-doc-readme.md* [L930-L1105], 📄 *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*
🔧 ✅ *scanning git*: 🌱 *"Vérification commit cible"*, 🌱 *"Vérification commit base"*, 🌱 *"Diff arbres liveSession"*, 🌱 *"Diff depuis inspection"*
🔧 ✅ *reading project*: 👉 *"Preuves architecture cœur"*,
🔎 r *"LiveSessionRunRef"*,
🔎 r *"LiveSessionPanelDescriptor"*,
🔎 r *"SidebarActivityQueryWindowItem"*, 🔎 r *"markSourceSettled"*, 🔎 r *"canAbortSource"*, 🔎 e *"Accès queryState windows"*, 🔎 e *"Type activité fenêtre"*, 🔎 e *"Run refs liveSession"*,
📄 *LiveSessionRun.type.ts*,
📄 *LiveSessionPanel.type.ts*, 📄 *LiveSessionOpenResult.type.ts*, 📄 *LiveSessionStream.util.ts*, 📄 *SidebarStore.ts* [L100-L175]
🔧 ✅ *scanning git*: 🌱 *"Usages runRef liveSession"*, 🌱 *"Usages descripteur panel"*, 🌱 *"Usages activité windows"*, 🌱 *"Usages arrêt liveSession"*
🔧 ✅ *reading project*: 📄 *LiveSessionRecord.type.ts*, 📄 *LiveSessionRegistry.service.ts* [L1-L220], 📄 *LiveSessionRegistry.service.ts* [L220-L660],
📄 *LiveSessionOpen.service.ts*,
📄 *LiveSessionReplay.service.ts*,
📄 *open.endpoint.ts*, 📄 *stream.endpoint.ts*, 📄 *abort.endpoint.ts*, 📄 *stream.endpoint.ts* [L1-L240], 🔎 e *"Contrat Slimbk errorCode"*, 🔎 h *"Client stream Slimbk"*
🔧 ✅ *scanning git*: 🌱 *"Usages SlimbkCallError"*, 🌱 *"Champs errorCode endpoint"*
🔧 ✅ *reading project*: 👉 *"Preuves transport Slimbk"*, 📄 *aiStopSuffix.util.ts*, 📄 *AiStreamedChunk.type.ts*, 📄 *AiStopErrorReason.type.ts*, 🔎 e *"Type AiErrorMetaValue"*
🔧 ✅ *scanning git*: 🌱 *"Usages suffix arrêt"*
🔧 ✅ *reading project*: 🔎 e *"Snapshot messages Orch"*, 🔎 r *"getThreadMessagesSnapshot"*, 📄 *OrchAsk.ts* [L1-L220], 🔎 h *"Continuation Orch memory"*, 📄 *OrchAskStream.ts* [L1-L180], 📄 *OrchMemory.ts* [L430-L480], 📄 *ChatSessionFile.type.ts*, 📄 *ChatPersistenceReader.service.ts* [L1-L180], 📄 *AiQuery.type.ts*, 📄 *usePromptRunCommands.hook.ts* [L1-L260], 📄 *usePromptRunCommands.hook.ts* [L260-L620], 📄 *preparePromptStreamingTurn.util.ts*, 📄 *EasyAgentsBatchRunChildExecution.service.ts* [L1-L360], 📄 *EasyAgentsBatchRunChildExecution.service.ts* [L360-L760], 📄 *EasyAgentsBatchRunExecution.service.ts* [L1-L260], 📄 *EasyAgentsBatchRunExecution.service.ts* [L260-L620], 📄 *EasyAgentsBatchRunManager.service.ts* [L1-L160], 📄 *EasyAgentsBatchRunManager.service.ts* [L220-L480], 📄 *EasyAgentsBatchLastViewerStop.service.ts*, 📄 *EasyAgentsBatchRunManager.types.ts* [L1-L220], 📄 *openPanel.util.ts* [L1-L260], 📄 *openPanel.util.ts* [L260-L560],
📄 *closePromptPanel.util.ts*,

OpenedPanelsRegistry.util.ts,
PromptThreadCleanup.manager.ts, PromptQueryDoneSound.manager.ts, openPanel.util.ts [L560-L741]
🔧 ✅ *scanning git*: ᛘ "Appels ouverture panneau", ᛘ "Appels cleanup Prompt", ᛘ "Appels close Prompt", ᛘ "Accès panneaux ouverts"
🔧 ✅ *reading project*: uiAiPromptHandlers.ts [L100-L145], SidebarQueryRunningBridge.ts, SidebarStore.queryWindow.listeners.manager.ts [L1-L340], SidebarStore.queryWindow.mutations.manager.ts [L1-L220], SidebarStore.rehydrate.manager.ts [L1-L280], SidebarActivityQuery.tsx [L1-L260], SidebarUiMessages.ts [L1-L220], SidebarStore.handlers.manager.ts [L1-L260], SidebarStore.core.manager.ts [L1-L240], registerAllBusHandlers.ts [L1-L180], 🔎 e "Message openPanel générique", 🔎 e "Canaux aiStream", 🔎 e "Canal liveSessionStream", AiStreamStartArgs.type.ts, useAiStream.hook.ts [L1-L250], useAiStream.hook.ts [L250-L520], useAiStream.hook.ts [L520-L750], useStreamListener.hook.ts [L1-L260], useAiStreamQueue.hook.ts [L1-L340], usePromptStreamDispatchGateQueue.hook.ts [L1-L300], PromptLiveSession.types.ts, usePromptLiveSession.hook.ts [L1-L260], usePromptLiveSession.hook.ts [L260-L620], EasyAgentsBatchLiveSession.adapter.ts, Prompt.tsx [L1-L220], Prompt.tsx [L220-L520], Prompt.tsx [L520-L840], Prompt.tsx [L840-L1337]
🔧 ✅ *scanning git*: ᛘ "Usage interactionsEnabled"
🔧 ✅ *reading project*: 🔎 e "Verrou input Prompt", 🔎 e "État input running", PromptHistory.tsx [L300-L370], 🔎 e "Affordances verrouillées", 👉 "Verrou actions historique", 🔎 e "Calcul idle historique", PromptHistory.tsx [L650-L790], PromptHistory.tsx [L500-L650], iaStreamHandlers.ts, liveSessionStreamHandlers.ts [L110-L260], 🔎 e "Mémoire getMessages", OrchMessagesMemory.ts [L1-L180], OrchMessagesMemoryCloner.ts [L1-L220], EasyAgentsBatchRunRetry.service.ts [L250-L330], EasyAgentsBatchRunRetry.service.ts [L190-L270], LiveSessionService.ts [L110-L220]
🔧 ✅ *scanning git*: ᛘ "Usages mutations activité", ᛘ "Usages état activité type"
🔧 ✅ *reading project*: 🔎 e "Implémentation onStreamed", extTypedMessageBus.util.ts [L1-L340], extMessageBus.util.ts [L1-L420]
🔧 ✅ *scanning git*: ᛘ "Retries frontend temporés"
🔧 ✅ *reading project*: 🔎 h "Polling annulable frontend", usePromptThinkingLifecycle.hook.ts [L1-L360], usePromptToolUse.hook.ts [L1-L260], usePromptRunController.finalization.hook.ts [L1-L340], usePromptRunController.hook.ts [L1-L260], usePromptRunController.hook.ts [L260-L620],
applyHydratedPromptSessionState.util.ts,
usePromptRunController.uiState.hook.ts, usePromptRunUiDerived.hook.ts, 🔎 e "Usage isLocked PromptInput", PromptInput.tsx [L680-L900], 🔎 e "Édition textarea verrou", 👉 "Contrôle édition input", PromptLiveSessionBanner.tsx, PromptLiveSessionBanner.module.css, PromptInput.tsx [L1-L180], 🔎 e "Type liveSessionViewModel", 🔎 e "Styles container PromptInput", PromptInput.module.css [L1-L140], PromptInput.tsx [L300-L430], 🔎 e "Props live session input", PromptInput.tsx [L180-L310], 🔎 e "Fallback continueResponse", 🔎 h "Durabilité continuation", fallback-persist-and-title.programming-doc-readme.md, 👉 "Barrière fallback par ui", VscButton.tsx [L1-L260], 🔎 e "VscLink testId"
🔧 ✅ *scanning git*: ᛘ "Usages query EasyAgents", ᛘ "Fixtures query windows", ᛘ "Fixtures isRunning query", ᛘ "Ascendance depuis 0b9", ᛘ "Diff complet des bases", ᛘ "Diff complet depuis 0b9", ᛘ "Métadonnées commit 0b9", ᛘ "Base commune commits", ᛘ "Métadonnées HEAD"
🔧 ✅ *reading project*: 📁 g "Tests finalisation controller", MessageRegistry.ts [L1-L80], usePromptSessionLocator.hook.ts [L1-L220], usePromptSessionLocator.hook.ts [L220-L320], frontMessageBus.util.ts [L240-L380], frontMessageBus.util.ts [L120-L245]
🔧 ✅ *scanning git*: ᛘ "Usages OpenResult live", ᛘ "Usages close policy", ᛘ "Usages stream liveSession", ᛘ "Usages release viewer", ᛘ "Usages mark handoff"
🔧 ✅ *reading project*: EasyAgentsBatchRunExecution.commit-and-recovery-failures.service.test.ts [L170-L235], EasyAgentsBatchRunExecution.kill.service.test.ts [L410-L550], EasyAgentsBatchRunExecution.post-commit-persistence.service.test.ts [L170-L225], 🔎 e "Assertions runRef kill"

🔧 ✅ *scanning git*: 🌱 *"Assertions handoff runRef"*, 🌱 *"Assertions source liveRun"*, 🌱 *"Tests canal aiStream"*, 🌱 *"Tests commandes start"*, 🌱 *"Types start AiStream"*, 🌱 *"Usages prepare streaming"*, 🌱 *"Mocks useAiStream"*, 🌱 *"Objets runRef inline"*, 🌱 *"Bootstraps liveSession en"*, 🌱 *"Appels gestionnaire son"*, 🌱 *"Tests directs useAiStream"*

🔧 ✅ *reading project*: 📄 *useAiStream.hook.rawLiveSideband.test.ts* [L50-L130]

🔧 ✅ *scanning git*: 🌱 *"Assertions args useStream"*, 🌱 *"Tests directs tool use"*

🔧 ✅ *reading project*: 🔎 e *"Live session dans Batch test"*, 🔎 e *"Live session table tests"*

🔧 ✅ *specifying*

"SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_025528"

SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_025528

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 présente réinspection a été reprise depuis zéro sur le dépôt courant au commit `b16c5bdde3b4a27cec2ee3f44c9a471b44465ac6` (`b16c5bdd`). Le commit `0b9f52cc402ce6d0ae19372fa62a432851c248b0`, utilisé lors d'une inspection précédente, n'est pas l'ancêtre direct du HEAD courant : les deux commits ont pour base commune `ed3bd94687cc34f6603730689bfc775818881bae`. Le diff complet entre `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33` et `ed3bd94687cc34f6603730689bfc775818881bae` est vide, et le diff entre `0b9f52cc402ce6d0ae19372fa62a432851c248b0` et le HEAD courant ne touche aucun fichier source ciblé par cette tranche. La base fonctionnelle `liveSession` reste donc exactement celle de la tranche 1, mais la référence correcte au dépôt courant est désormais `b16c5bdd`.

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, finalisation frontend 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, y compris lorsque le record `liveSession` a dû être retiré faute de baseline ou de vrai terminal ;
- 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 ;
- la présence de n'importe quel paramètre privé liveSession classe la query comme explicite : un descripteur partiel, incohérent ou avec adaptateur inconnu est invalide et ne se dégrade jamais silencieusement en 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 ;
- `streamInstanceId` remplace l'actuel identifiant de listener `streamKey` : il n'existe pas deux identités frontend concurrentes pour la même tentative ;
- les callbacks frontend `onData`, `onComplete` et `onError`, leurs continuations après `await`, ainsi que le `runMerged` déjà retiré de Queue A mais encore en cours dans Queue B ou dans les effets Prompt, sont gardés par `sourceId + streamInstanceId` ;
- le handoff direct normal exige un vrai terminal reçu comme donnée, une complétion transport normale, le drainage Queue A/Queue B et un accusé explicite de finalisation Prompt pour le même `streamInstanceId` ;
- une vraie erreur source transportée dans le premier header Slimbk doit 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 de la première ligne utile 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 ;
- cette finalisation post-forwarding est réservée aux sources `ownership='source'` ; le settlement/handoff EasyAgents reste piloté par son adaptateur métier historique ;
- `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 ;
- lors d'un retry frontend de replay, la nouvelle réponse Open remplace la baseline des messages et les stores d'hydratation, mais ne doit jamais écraser le brouillon, les attachments ni `nextTurnId` déjà modifiés localement dans le panneau reconnecté ;
- pour `continueAssistant`, la baseline est déclarée prête immédiatement après la promotion réservation → record, à partir du snapshot défensif capturé avant `askStream(...)` ; pour tous les modes `replaceTurnAssistants`, la baseline devient prête uniquement sur le chunk `fallback_persisted` appendu au registre ;
- 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 ;

- pour l’Activité, tout résultat non-stale signifie que la source exacte est terminée : panneau ouvert → projection Idle conservée ; panneau fermé → suppression ; `stale` → aucune mutation de la projection plus récente ;
- le son de fin est corrélé au run exact `{ locator, turnId, sourceId }`, y compris pour le terminal source décodé depuis le premier header Slimbk ; chaque appel `onAiStreamStart` retourne en plus un identifiant interne de tentative sonore, exigé par `onAiStreamChunk` et `onAiStreamStop`, afin qu’un ancien observer fermé ne puisse pas arrêter ou muter la tentative plus récente du même `sourceId` ;
- 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 Prompt normale active ou réservée » en complément de la projection UI ;
- l’elapsed d’une source Prompt normale vient de `sourceStartedAtMs` et continue panneau fermé ; l’elapsed EasyAgents observer reste ancré au démarrage de la tentative viewer, afin de préserver son UX historique ;
- 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 ; seules leurs entrées/sorties sont gardées par l’identité de tentative courante ;
- 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, seule la soumission Send reste verrouillée, tandis que la saisie du prochain brouillon et la préparation des attachments restent autorisées ;
- les actions destructives de l’historique, le cleanup classique et l’ouverture d’une nouvelle source restent verrouillés jusqu’au handoff direct réussi ;
- 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.
- Conserver l’item Idle si le panneau reste ouvert lorsque le processus se termine, quelle que soit l’issue finale exacte du record liveSession.
- 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 `b16c5bdd`; les arbres source ciblés sont inchangés par rapport à `0b9f52cc`, `ed3bd94687cc34f6603730689bfc775818881bae` et à la base fonctionnelle `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33`.
- `LiveSessionRunRef` devient `{ locator, turnId, sourceId }`. `turnId` reste la corrélation logique persistante ; `sourceId` identifie une invocation backend précise.
- Les paramètres query string exacts ajoutés sont `liveSessionSourceId` et `liveSessionAdapter`; l'adaptateur strict vaut `easyAgentsBatch | normalPrompt`.
- `streamInstanceId` reste frontend-only, change à chaque tentative acceptée, sert directement de clé `useStreamListener` 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 synchrone du start et génère toujours un nouveau `streamInstanceId`. Un ref d'acceptation synchrone empêche deux appels successifs dans le même tick de contourner le single-flight avant le prochain rendu React.
- Les callbacks stale sont ignorés avant toute mutation, après chaque `await`, avant tout reset de queue/lock et dans le runner de chunks déjà sorti de Queue A.
- 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 ; pour `replaceTurnAssistants`, fournir une baseline de messages est interdit.
- 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 deviennent un `meta.error` canonique `SYSTEM_ERROR`; un EOF sans vrai terminal devient un `meta.error` canonique `INTERRUPTED`; les deux sont bufferisés et exposés.
- 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 chaque étape destructive et après chaque `await` de cleanup.
- 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.
- Pour normalPrompt, `not_found` après une query exacte active/réservée, y compris après une séquence `not_ready`, mène à `liveSessionReplayFailed` dans le panneau ; il ne déclenche pas un fallback classique ni une fermeture silencieuse.
- `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.
- Un retry de replay recharge la baseline des messages et les stores d’hydratation, tout en préservant les champs du compositeur déjà modifiés localement.
- 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, avec un identifiant interne de tentative pour les reconnects du même run.
- 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. Le texte thinking de base, le séparateur de continuation et l’accusé de finalisation sont corrélés au `streamInstanceId`.
- Les nettoyages « premier token texte » et « premier tool use » ne retirent la bulle thinking que si le thinking de base est vide.
- 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 synchrone 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 et de son identifiant interne de tentative ;
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 ; pour `continueAssistant`, publication immédiate de la baseline messages ;

13. append de chaque chunk avant exposition ; pour `replaceTurnAssistants`, le chunk `fallback_persisted` rend la baseline prête ;
14. replay possible ;
15. vrai terminal ou synthèse canonique ;
16. fin de consommation backend sans retrait du run actif ;
17. 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 ;
18. finalisation atomique post-forwarding : handoff si baseline + vrai terminal, sinon retraite/éviction exacte ;
19. finalisation frontend de la tentative courante, avec publication de `finalizedStreamInstanceId` ;
20. Activité Idle si panneau ouvert, suppression si panneau fermé ; un résultat stale ne touche jamais l'activité d'une source plus récente ;
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
source terminée	open	aucun	Idle, même si le record a été retiré sans handoff
source terminée	closed	aucun	aucun item

Frontend reconnecté

Après installation du snapshot et armement du replay, le compositeur et les attachments redeviennent visibles. Le verrou Send et le Stop normal du compositeur restent actifs sans transformer le Prompt en viewer read-only. Les actions de l'historique et le cleanup classique restent désactivés jusqu'au handoff direct.

Chaque retry possède un nouveau `streamInstanceId`. Les callbacks antérieurs, les continuations async et les chunks déjà sortis de Queue A ne peuvent plus muter le runtime. Le handoff normal est direct uniquement après vrai terminal de données, drainage Queue A/Queue B, `outcome='completed'` et `finalizedStreamInstanceId ===`

`streamInstanceId` pour la tentative courante. Si la subscription se termine par éviction sans vrai terminal, le hook entre dans `liveSessionReplayFailed` au lieu de passer à `normal`.

En `liveSessionReplayFailed`, le compositeur reste visible et éditable, Send reste verrouillé, le Stop normal reste la seule action Stop et un bandeau sans bouton Stop expose le Retry de replay. Le retry remplace uniquement l'état de messages/hydratation requis par la nouvelle baseline et conserve le brouillon, les attachments et `nextTurnId` locaux.

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.

Le son maintient deux identités distinctes :

- le runRef exact pour la déduplication fonctionnelle du terminal ;
- un `soundAttemptId` process-local retourné par `onAiStreamStart`, pour ignorer les chunks/stop d'une ancienne tentative viewer du même run.

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 exactement `LIVE_SESSION_SOURCE_ID_PARAM='liveSessionSourceId'` et `LIVE_SESSION_ADAPTER_PARAM='liveSessionAdapter'`.
- Ajouter le schéma strict adaptateur `easyAgentsBatch | normalPrompt`.
- Réutiliser le schéma `sourceId`; rendre `continueOnPanelClose` effectif pour normalPrompt.
- Documenter les couples valides : EasyAgents + `stopOnLastViewerClose`; normalPrompt + `continueOnPanelClose`.

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, noms exacts des query params, adaptateurs, couples close-policy, 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 et l'absence de propriétés étrangères.
- Pour une stack, extraire le JSON de la ligne contenant le préfixe afin que les frames suivantes ne contaminent pas le parse.
- 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, propriétés supplémentaires invalides, 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` uniquement sur la branche source.
- 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, barrière post-forwarding et distinction runRef/soundAttemptId/streamInstanceId.

12) LiveSessionRecord.type.ts — modifié

- Ajouter `baselineOnly`, replayMode, baseline continuation, timestamp, abortRequested, claim d'abort, viewerGroup optionnel, réservations/snapshots/retraites/époque terminale.
- Définir l'union `LiveSessionPostForwardingFinalizationResult` : `reservationCancelled | handoffReady | retiredMissingBaseline | retiredMissingTerminal | stale`.
- 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` et exposer des claims exacts distinguant viewer observer et source normale.
- Finalisation post-forwarding idempotente, exacte et autorisée uniquement pour `ownership='source'`.
- 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.
- Conserver l'elapsed source depuis `sourceStartedAtMs`; ne pas imposer cet ancrage aux observers EasyAgents.

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.
- Vérifier l'idempotence des cinq résultats de finalisation et le refus de cette finalisation sur un record observer.

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, les snapshots actifs/réservés et les claims exacts.

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 par un clone défensif de la baseline.
- 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.
- Permettre au caller de choisir explicitement si un abort thread-level est autorisé sur échec post-return, afin qu'un échec de promotion normalPrompt ne puisse jamais tuer un tiers.

21) LiveSessionSourceLifecycle.service.test.ts — créé

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

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.
- Pour `continueAssistant`, publier immédiatement la baseline messages ; pour `replaceTurnAssistants`, publier la baseline uniquement lors de `fallback_persisted`.
- Parser, append-before-yield, vrai terminal.
- Premier meta.error appendu inchangé puis exposé à l'endpoint pour encodage header V1.
- Transformer throw en SYSTEM_ERROR canonique et EOF sans terminal en INTERRUPTED canonique.
- Ne pas finaliser le registre ; garantir qu'un échec registration/promotion ne touche aucun tiers.

23) NormalPromptLiveSessionSource.service.test.ts — créé

- Couvrir tous les query types, continuation, readiness immédiate/ack, 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` au profit du claim viewer exact.
- 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) EasyAgentsBatchRunExecution.commit-and-recovery-failures.service.test.ts — modifié

- Ajouter `sourceId=turnId` aux assertions exactes de `markHandoffReady` sans modifier les invariants d'ordre commit/cleanup/publication.

29) EasyAgentsBatchRunExecution.post-commit-persistence.service.test.ts — modifié

- Ajouter `sourceId=turnId` aux assertions exactes de handoff et préserver l'ordre patch parent → publication → handoff.

30) EasyAgentsBatchRunChildExecution.service.ts — modifié

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

31) EasyAgentsBatchRunChildExecution.service.test.ts — modifié

- Conserver intégration Batch et déplacer tests lifecycle purs.

32) EasyAgentsBatchRunManager.service.ts — modifié

- Injecter lifecycle, migrer `items`, consulter `hasActiveOrReservedSourceForLocator` en complément de la projection Activité pour Retry.

33) EasyAgentsBatchRunExecution.testHarness.ts — modifié

- Injecter lifecycle et runRefs sourceId.

34) EasyAgentsBatchRunLifecycle.service.test.ts — modifié

- Adapter construction au lifecycle partagé.

35) EasyAgentsBatchLastViewerStop.service.test.ts — modifié

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

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

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

Endpoints et message bus

37) open.endpoint.ts — modifié

- Exiger sourceId et transporter sourceId/replayMode.

38) open.endpoint.test.ts — modifié

- Couvrir champs, branches et mismatch.

39) stream.endpoint.ts — modifié

- Exiger sourceId et relayer runRef complet.

40) stream.endpoint.test.ts — modifié

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

41) abort.endpoint.ts — modifié

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

42) abort.endpoint.test.ts — modifié

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

43) abortSource.endpoint.ts — créé

- Stop exact source Prompt normal par locator+turn+source, claim puis abort sans await.
- Autoriser réservation et record ; refuser ownership observer, settled, stale et claim consommé.

44) abortSource.endpoint.test.ts — créé

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

45) 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.

46) stream.endpoint.test.ts — modifié

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

47) liveSessionStreamHandlers.ts — modifié

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

48) liveSessionStreamHandlers.test.ts — modifié

- Couvrir sourceId, ownership, observer fermé, son exact, reconnect même source avec ancien Stop sonore stale, absence duplication, éviction sans terminal et ordre.

49) iaStreamHandlers.ts — modifié

- Recevoir sourceId et construire runRef exact.
- Réserver synchroniquement avant tout await, publier Activity et armer son exact avec `soundAttemptId`.
- 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.

50) iaStreamHandlers.queryDoneSound.test.ts — modifié

- Vérifier runRef/sourceId/soundAttemptId, 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.

51) 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.

52) PromptQueryDoneSound.manager.ts — modifié

- Corréler état actif/dédup par runRef exact, callbacks stale no-op, pas de création implicite par chunk/stop.
- Faire retourner par `onAiStreamStart` un `soundAttemptId` monotone/process-local ; exiger runRef + attempt pour chunk/stop.
- Ne jamais réarmer la déduplication d'un run déjà terminal lors d'une reconnexion du même sourceId.
- Préserver TOOL_USE/ABORTED/fréquences et traiter l'erreur source décodée comme le même terminal que le buffer.

53) PromptQueryDoneSound.manager.test.ts — modifié

- Couvrir mêmes turns/sources différentes, deux attempts du même source, stale chunk/stop, dédup terminal et absence de création tardive.

Activité runtime extension

54) 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.
- Pour source, utiliser `sourceStartedAtMs`; pour observer, conserver un startMs de tentative viewer.

55) SidebarQueryRunningBridge.race.test.ts — modifié

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

56) SidebarQueryRunningBridge.usage.test.ts — modifié

- Usage exact et transition Idle/suppression, y compris retrait sans terminal : panneau ouvert → Idle ; panneau fermé → suppression.

Coordination panneau Prompt

57) PromptLiveSessionPanelQuery.util.ts — créé

- Parser locator et descripteur complet, construire query normalPrompt, distinguer classique/explicite valide/invalide, retirer autoRun lors injection.
- Classer comme explicite toute query contenant au moins un paramètre liveSession, même sans `openMode`.
- Rejeter doublons/incohérences des paramètres critiques au lieu de choisir implicitement une occurrence.

58) PromptLiveSessionPanelQuery.util.test.ts — créé

- Couvrir identité, noms exacts des paramètres, adapter, ordre, policies, partiels, doublons, invalide et suppression autoRun.

59) PromptLiveSessionPanelCoordinator.service.ts — créé

- Classe OOP injectée, aucun import low-level.
- Queue FIFO par locator supprimée à idle.
- `openPromptPanel`, `onPanelDisposed`, `onSourceSettled` avec réévaluation avant chaque cleanup et après chaque await.
- Préserver live explicite ; injecter normalPrompt seulement sans descripteur et avec source Prompt exacte active/réservée.
- Dispatch release EasyAgents/normalPrompt, detach source sans abort, ignorer settlement stale.
- Traiter `retiredMissingBaseline|retiredMissingTerminal` comme fin autoritative sans handoff : panneau fermé → cleanup ; panneau ouvert/replay attaché → conserver le panneau et laisser le frontend afficher `liveSessionReplayFailed` sans restaurer artificiellement `normal`.
- Ne jamais supprimer l'item Idle d'un panneau encore ouvert uniquement parce que le record a été retiré.

60) PromptLiveSessionPanelCoordinator.service.test.ts — créé

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

61) PromptLiveSessionDetachedStop.service.ts — créé

- Valider synchroniquement closed + Running/source exact + source active/réservée + claim non consommé ; mutation stopRequested puis abort sans await.
- Sur refus dû à une projection stale, ne jamais abort une autre source et laisser la prochaine transition/rehydratation corriger l'item.

62) PromptLiveSessionDetachedStop.service.test.ts — créé

- Couvrir accept/refus, reopened, replacement, double clic, réservation, store stale et ordre mutation/abort.

63) 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/réservée ni un observer actif.
- Revalider l'absence de source/panneau après chaque await du cleanup historique.

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

- Couvrir scopes, detach, idle, terminal, mapping absent, nouvelle source pendant cleanup et sérialisation attachments.

65) 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`.
- Exporter une notification coordonnée de settlement utilisable par le handler source sans que le coordinateur importe l'opener.

66) openPanel.util.test.ts — modifié

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

67) uiGenericHandlers.ts — modifié

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

68) uiGenericHandlers.openExternalUrl.test.ts — modifié

- Mocker façade Prompt et préserver URL.

69) NewChat.command.ts — modifié

- Await ouverture coordonnée.

70) NewChat.command.test.ts — modifié

- Vérifier async et propagation erreur.

71) registerDeepLinkUriHandler.ts — modifié

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

72) registerDeepLinkUriHandler.scope.test.ts — modifié

- Couvrir root/nested, façade coordonnée et guards.

SidebarStore et Activité

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

- Transitions atomiques `panelOpened`, `sourceStarted`, `sourceSettled`; lecture panelState et source active sans await intermédiaire.
- `sourceSettled` reçoit le résultat strict de finalisation.
- Mapping obligatoire : résultat non-stale + panneau ouvert → Idle ; résultat non-stale + panneau fermé → suppression ; stale → no-op.

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

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

75) SidebarStore.core.manager.ts — modifié

- Initialiser items et appeler `rehydrateQueryActivities`.

76) SidebarStore.rehydrate.manager.ts — modifié

- Fusionner panneaux, observers actifs et sources/réservations ; dédupliquer et interdire observer/closed/idle closed.
- Elapsed des sources normales depuis timestamp ; elapsed observer depuis la tentative viewer lorsque le runtime existe.
- Terminal bufferisé non finalisé reste Running ; run retiré sans terminal n'est jamais re-projeté comme actif, mais un panneau ouvert reste projeté Idle.

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

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

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

- Tester machine d'état, identités, panneau ouvert/fermé et mapping des outcomes stricts.

79) SidebarStore.chatTitles.manager.ts — modifié

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

80) SidebarStore.handlers.manager.ts — modifié

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

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

- Vérifier payloads, accepted et délégation.

82) delete.endpoint.ts — modifié

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

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

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

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

- Adapter fixtures items.

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

- Adapter items et préserver delete disque.

86) ChatPersistenceWriter.windowsFallback.test.ts — modifié

- Adapter fixture items.

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

- Adapter fixture items.

88) 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.

89) SidebarActivityQuery.test.tsx — modifié

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

Frontend `liveSession` et Prompt

90) NormalPromptLiveSession.adapter.ts — créé

- Parser normalPrompt, sourceId, lease, close policy et couple adaptateur/policy exact.
- Policies not_ready retry, not_found/replay failure in-place, handoff direct, replay recovery, surface composer interactive ; aucune action Stop dans le bandeau.
- Pendant streaming/replayFailed, forcer uniquement le Stop normal du compositeur et le verrou Send.

91) NormalPromptLiveSession.adapter.test.ts — créé

- Couvrir parsing, paramètres partiels, policies, view models composer/bandeau et absence de Stop banner.

92) PromptLiveSessionBootstrap.adapter.ts — créé

- Dispatch strict EasyAgents/normalPrompt, aucun fallback.
- Toute présence partielle de paramètre liveSession doit produire `invalid`, même si `openMode` manque.

93) PromptLiveSessionBootstrap.adapter.test.ts — créé

- Couvrir dispatch, partiels, doublons et inconnus.

94) PromptLiveSession.types.ts — modifié

- Ajouter policies et `liveSessionReplayFailed`; supprimer mode⇒viewer et helper mort ; runRef sourceId.
- Séparer explicitement : surface composer, verrou Send, actions historique, cleanup classique, Stop compositeur et bandeau optionnel.
- Permettre un bandeau Retry sans Stop au-dessus d'un compositeur normal visible.

95) usePromptLiveSession.hook.ts — modifié

- Transporter sourceId/replayMode, retry not_ready annulable, replay selon mode.
- Handoff direct normal uniquement vrai terminal data + queues + outcome completed + finalisation Prompt du même streamInstanceId.
- `transportError`, completed sans terminal, `not_found` normalPrompt 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.
- Sur retry, utiliser les setters d'hydratation de façon à préserver le brouillon, les attachments et `nextTurnId` locaux tout en remplaçant les messages/stores de baseline.

96) usePromptLiveSession.hydration.test.tsx — modifié

- Ajouter identité/replayMode, comportements not_ready/not_found par adaptateur et préserver EasyAgents.

97) usePromptLiveSession.handoff.test.tsx — modifié

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

98) usePromptLiveSession.normalPrompt.test.tsx — créé

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

99) 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.

100) EasyAgentsBatchLiveSession.adapter.ts — modifié

- Ajouter sourceId=turnId/adaptateur, préserver UX, elapsed viewer et bandeau read-only historique.

101) EasyAgentsBatchLiveSession.adapter.test.ts — modifié

- Adapter query, sourceId et dispatcher.

102) AiStreamStartArgs.type.ts — modifié

- Garder promptQuery public sans sourceId ; état interne runRef+streamInstanceId+abort policy ; outcome `pending|completed|transportError`.
- Le live observer transporte le runRef complet, la lease, le replayMode et la policy Stop, sans fonction ni donnée frontend-only envoyée au backend.

103) useAiStream.hook.ts — modifié

- Générer identités après acceptation synchrone, avec ref single-flight immédiatement muté pour bloquer les doubles starts du même tick.
- Utiliser streamInstanceId comme clé listener à la place de streamKey.
- 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.
- Garder onData/onComplete/onError et toutes leurs continuations async par runRef+streamInstanceId.
- Lier également le `runMerged` de la queue à l'instance capturée et revalider avant/après ses awaits afin qu'un vieux drain déjà extrait ne puisse pas muter la nouvelle tentative.
- Ne modifier aucun algorithme Queue/lock/sanitizer/raw-live.

104) 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.
- Couvrir un vieux runMerged/onComplete qui reprend après await et ne reset jamais la nouvelle tentative.

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

- Vérifier unicité, stabilité, double start même tick, abort exact, callbacks stale et observer retry.

106) usePromptRunController.hook.ts — modifié

- Observer reçoit runRef+lease+replayMode+Stop policy ; base thinking + streamInstanceId ; exposer outcome/tentative.
- Conserver une initialisation thinking en attente, la lier synchroniquement au streamInstanceId rendu avant les effets transport.
- Exposer `finalizedStreamInstanceId` et reset complet.

107) usePromptRunController.liveSessionObserver.test.tsx — modifié

- Couvrir identité, modes, base thinking, tentative, outcome et accusé de finalisation exact.

Continuation et thinking

108) 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.
- Retourner une structure de préparation permettant au controller de lier la base thinking à la prochaine tentative acceptée sans générer l'identité en avance.

109) preparePromptContinuationStreaming.util.test.ts — créé

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

110) preparePromptStreamingTurn.util.ts — modifié

- Reset explicite base thinking/séparateur/initialisation en attente pour replace.

111) preparePromptStreamingTurn.util.test.ts — modifié

- Vérifier resets.

112) usePromptThinkingLifecycle.hook.ts — modifié

- Recevoir base thinking et streamInstanceId ; initialiser avant effets transport ; séparateur unique ; préserver historique ; reset replace.
- `usePromptFirstArrivalCleanup` ne retire la bulle que si la tentative courante n'a ni thinking reçu ni thinking de base.

113) usePromptThinkingLifecycle.continuation.test.tsx — créé

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

114) usePromptToolUse.hook.ts — modifié

- Supprimer bulle vide uniquement si baseThinking vide.
- Ne jamais armer deux séparateurs simultanés lorsque continuation et TOOL_USE précèdent le même prochain paquet thinking.

115) usePromptRunController.finalization.hook.ts — modifié

- Garder toutes les mutations par streamInstanceId.
- Reset base thinking/séparateur après terminal, ordre inchangé.
- Publier `finalizedStreamInstanceId` uniquement après meta/notice/message/ref finalisés pour la tentative courante.

116) usePromptRunCommands.hook.ts — modifié

- Remplacer bootstrap continue inline par helper ; préserver autres commandes.
- Transmettre/reset l'état de base thinking requis par les helpers replace/continue.

117) usePromptRunCommands.compactContextNow.test.tsx — modifié

- Adapter le harness au nouveau contrat de préparation thinking et vérifier que compact-context reste `replaceTurnAssistants` sans base héritée.

118) usePromptRunCommands.retryQuery.attachmentOnly.test.tsx — modifié

- Adapter le harness au nouveau ref thinking et préserver les invariants attachment-only/retry.

119) usePromptRunCommands.retryQuery.compactContextNow.test.tsx — modifié

- Adapter le harness au nouveau ref thinking et préserver le mapping retry compact-context.

120) usePromptRunCommands.continueResponse.test.tsx — créé

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

Intégration Prompt et documentation

121) Prompt.tsx — modifié

- Capturer source/adaptateur dans query figée, dispatcher, passer outcome/runRef/tentative/finalizedStreamInstanceId.
- Déduire séparément : visibilité/édition du compositeur, verrou Send, running/Stop compositeur, interactions historique et ownership cleanup.
- Autoriser draft/interactions de compositeur normaux pendant replay normalPrompt, y compris persistance debounced du brouillon lorsque le mode n'est pas encore `normal` mais que la policy autorise la surface interactive.

- Utiliser les flags effectifs pour global hotkeys et cancel controller, afin qu'un replayFailed ne puisse pas déclencher un nouveau Send.
- Cleanup ownership différé jusqu'au handoff direct ; hydration classique off pour tout live bootstrap.

122) Prompt.liveSessionBootstrap.test.tsx — modifié

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

123) PromptInput.tsx — modifié

- Étendre la projection liveSession pour permettre trois surfaces sans dupliquer le compositeur :
- composer normal seul ;
- viewer banner remplaçant le composer pour EasyAgents ;
- composer normal + bandeau de statut/retry sans Stop pour normalPrompt.
- Le bandeau normalPrompt ne remplace jamais textarea/attachments et n'ajoute jamais un second Stop.
- Conserver `running`/`stopEnabled` sur le `VscTextarea` comme unique Stop normal.

124) PromptInput.module.css — modifié

- Ajouter uniquement le wrapper/espacement nécessaire au bandeau normalPrompt au-dessus du compositeur.
- Utiliser les variables VS Code existantes via le composant de bandeau ; aucune couleur hardcodée et aucune CSS inline.

125) PromptInput.batchLiveViewerMode.test.tsx — modifié

- Préserver viewer EasyAgents.
- Ajouter normalPrompt interactif avec textarea/attachments, Stop normal unique, Send verrouillé selon policy, bandeau Retry sans Stop et aucun remplacement du compositeur.

126) EasyAgentsBatch.liveWatch.test.tsx — modifié

- Adapter query exacte source/adaptateur.

127) EasyAgentBatchSurface.test.tsx — modifié

- Adapter query exacte sans changement UX.

128) 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, conservation composer sur retry et barrière post-forwarding/finalizedStreamInstanceId.
- Documenter que l'erreur source du premier header est transformée en chunk de données exact par le handler extension avant le frontend.

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

- Documenter placeholder vide, bulle agrégée, base thinking de continuation, séparateur unique, guards premier text/tool 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/**test**/EasyAgentsBatchRunExecution.commit-and-recovery-failures.service.test.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/**test**/EasyAgentsBatchRunExecution.post-commit-persistence.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/**test**/usePromptRunCommands.compactContextNow.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/**test**/usePromptRunCommands.retryQuery.attachmentOnly.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/**test**/usePromptRunCommands.retryQuery.compactContextNow.test.tsx
- 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/PromptInput.tsx
- AICode/src/frontend/src/app/prompt/PromptInput/PromptInput.module.css

- 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, chunks déjà sortis de Queue A, finalisation React par tentative, 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 avec reconnect même source, 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. Enfin, si le compositeur interactif, le runner de queues ou la finalisation React ne sont pas corrélés à la tentative exacte, un ancien retry peut reset la nouvelle source ou détruire le thinking de continuation. La reconstruction extension dans la FIFO unique, l'éviction exacte des runs non-handoffables, les identités exactes, réservations, époques, guards de tentative 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 unique servant de clé transport.
- 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.
- Conserver l'item Activité Idle lorsqu'un panneau est encore ouvert, même après retrait exact du record ; supprimer uniquement lorsque le panneau est fermé.
- Capturer les messages de continuation sans figer l'enveloppe.
- Réutiliser assistant/thinking sans duplication et préserver le thinking de base face aux premiers text/tool.
- Garder callbacks, queues, finalisation Prompt et son par identités exactes de run/tentative.
- 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, avec attempt interne pour reconnect du même run.
- Conserver le compositeur visible pendant le replay normalPrompt, avec Send verrouillé, Stop normal unique et bandeau Retry sans Stop.
- Préserver brouillon, attachments et nextTurnId lors des retries de replay.
- Utiliser le registre comme autorité deletion/Retry.
- Préserver EasyAgents, queues, locks et sanitizers.
- Tester stream continu, replay, hydratation, erreurs source encodées, transport réel, guards stale 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_025528

Copy
Refine
Code
Verify

Gravité	Modification
🟥 majeure	**Projection Activité corrigée après finalisation** : tout résultat non-`stale` produit désormais **Idle l** lorsque le panneau est fermé, y compris pour `retiredMissingBaseline` et `retiredMissingI`
🟥 majeure	**Surface normalPrompt reconnectée complétée** : ajout de PromptInput.tsx et PromptInput.modul reste verrouillé, le Stop normal reste unique et un bandeau Retry sans Stop peut coexister avec le c
🟥 majeure	**Protection `streamInstanceId` étendue au-delà des callbacks transport** : le runner déjà sort finalisation React sont désormais gardés. `streamInstanceId` remplace l'actuel `streamKey`.
🟥 majeure	**Accusé explicite de finalisation Prompt ajouté** : `finalizedStreamInstanceId` est publié seu le nettoyage des refs. Le handoff direct exige son égalité avec la tentative courante.
🟥 majeure	**Retry de replay rendu non destructif pour le compositeur** : la nouvelle baseline remplace les m attachments et `nextTurnId` déjà modifiés localement. `not_found` normalPrompt conduit à `live`
🟥 majeure	**Corrélation sonore renforcée pour les reconnects du même run** : `onAiStreamStart` retourne tandis que la déduplication terminale reste fondée sur le runRef exact.
🟧 mineure	**Readiness de continuation explicitée** : `continueAssistant` publie immédiatement sa baseline exclusivement le chunk `fallback_persisted`. Les nettoyages premier text/tool conservent la bu
🟧 mineure	**Contrat des queries rendu totalement déterministe** : noms exacts `liveSessionSourceId` et `l` considérée explicite dès qu'un paramètre liveSession est présent, rejet des descripteurs partiels ou c
🟧 mineure	**Inventaire des tests complété** avec EasyAgentsBatchRunExecution.commit-and-recovery-failures persistence.service.test.ts et les trois suites existantes de `usePromptRunCommands`.
🟧 mineure	**Référence Git actualisée** vers le HEAD courant `b16c5bdd`, avec clarification de sa relation de br

</ASSISTANT>
SHOW ALLPage 1 / 1