Permalink URI:
vscode://joelabenhaim.aicode-ja/chat?sessionId=f19df6b55a7941bc85151f9aae71f938-RS-Affiner-la-sessio
<USER>
[Refine]SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_171019
</USER>
<ASSISTANT>
I'm thinking...
</ASSISTANT>
<ASSISTANT>
🔧 ✅ *scanning git*: ᛘ *"Référence Git courante"*, ᛘ *"État Git courant"*
🔧 ✅ *reading project*: 🟨 g *"Docs architecture ciblées"*, 🟨 g *"Docs frontend Prompt"*, 🟨 g *"Docs frontend bus"*, 🟨 g *"Arbre liveSession"*, 🟨 g *"Frontend liveSession"*, 📄 *liveSession.programming-doc-readme.md*, 📄 *easyAgents.programming-doc-readme.md*, 📄 *isomorphism-invariant-rule-between-streaming-and-rehydration-of-chat-sessions.programming-doc-readme.md*, 📄 *thinking-bubble-and-hourglass-streaming-rules-invariants.programming-doc-readme.md*, 📄 *prompt-stream-dispatch-lock.programming-doc-readme.md*, 📄 *fallback-persist-and-title.programming-doc-readme.md*,
🔎 r *"LiveSessionRunRef"*,
🔎 r *"LiveSessionRegistry"*,
🔎 r *"useAiStream"*,
🔎 r *"applyHydratedPromptSessionState"*, 🔎 r *"usePromptLiveSession"*, 🔎 r *"openPanel"*, 🔎 e *"Paramètre live turn"*, 🔎 e *"Type activité fenêtre"*, 🔎 e *"Démarrage son Prompt"*, 🔎 e *"Start observer frontend"*, 🔎 e *"Autorité abort source"*, 🔎 e *"Patch assistant terminal"*, 🔎 e *"Éviction locator live"*, 🔎 h *"Source Prompt normale"*, 👉 *"Contrats live actuels"*,
📄 *LiveSessionRun.type.ts*,
📄 *LiveSessionOpenResult.type.ts*,
📄 *LiveSessionRecord.type.ts*,
📄 *LiveSessionRegistry.service.ts*,
📄 *LiveSessionOpen.service.ts*,
📄 *LiveSessionReplay.service.ts*,
📄 *EasyAgentsBatchLiveSessionAdapter.service.ts*,
📄 *usePromptSessionLocator.hook.ts*,
📄 *useQueryString.hook.ts*,
📄 *AICodeEnv.type.ts*,
📄 *useAicodeEnv.hook.ts*,
📄 *EasyAgentsBatchLiveSession.adapter.ts*,
📄 *PromptLiveSession.types.ts*,
📄 *AiPromptUiMessages.ts*,
📄 *MessageRegistry.ts*,
📄 *openPanel.util.ts*,
📄 *closePromptPanel.util.ts*,
📄 *OpenedPanelsRegistry.util.ts*,
📄 *PromptThreadCleanup.manager.ts*,
📄 *SidebarQueryRunningBridge.ts*,
📄 *SidebarStore.queryWindow.mutations.manager.ts*,
📄 *SidebarStore.core.manager.ts*, 📄 *SidebarStore.handlers.manager.ts*, 🔎 e *"Enregistrement cleanup Prompt"*,
📄 *AiStreamStartArgs.type.ts*,
📄 *useAiStream.hook.ts*,
📄 *useAiStreamQueue.hook.ts*,
📄 *usePromptStreamDispatchGateQueue.hook.ts*,
📄 *useStreamListener.hook.ts*, 📄 *PromptStreamLockStore.ts*, 🔎 e *"Commande Send Prompt"*, 🔎 e *"Commande SPECIF Prompt"*, 🔎 e *"Commande Retry Prompt"*, 🔎 e *"Commande Continue Prompt"*, 🔎 e *"Commande Edit Prompt"*, 🔎 e *"Start query controller"*, 🔎 e *"Start observer controller"*, 🔎 e *"Utilisation handleSend"*, 🔎 e *"Barrière clear Prompt"*, 🔎 e *"Transaction submit Prompt"*, 👉 *"Commandes Prompt actuels"*, 🔎 e *"Définition gateAction"*, 🔎 e *"Appels gateAction"*, 🔎 e *"Callback action index"*, 🔎 e *"Type action index"*, 📄 *usePromptIndexPending.hook.tsx*, 🔎 e *"Hook index Prompt"*, 👉 *"Gate index Prompt"*, 🔎 e *"Flip UI avant*

submit", 👉 "Pré-submit Prompt", Prompt.tsx [L1-L360], FrontChatSession.type.ts, aiStopSuffix.util.ts, AiStreamedChunk.type.ts, AiStopErrorReason.type.ts, OrchMessagesMemoryAssistantPatch.ts, OrchAskStreamErrors.ts, stream.endpoint.ts, OrchAsk.ts, EasyAgentsBatchRunChildExecution.service.ts, EasyAgentsBatchRunExecution.service.ts, EasyAgentsBatchRunRetry.service.ts, 🔎 e "Méthode askStream", 🔎 e "Mémoire stream Orch", 🔎 e "Snapshot messages thread", 🔎 e "Attente stop Orch", 👉 "API Orch stream", 🔎 e "Client Slimbk stream", 🔎 e "Callback erreur stream", 🔎 e "Type SlimbkCallError", 🔎 e "Erreur header générée", 👉 "Sémantique client Slimbk", 🔎 e "Appels openCentralPanel", 🔎 e "Ouvertures Prompt directes", 🔎 e "Imports openPanel util", 🔎 e "Fermetures Prompt directes", SidebarActivityQuery.tsx [L100-L260], VscButton.tsx, VscButton.module.css, SidebarUiMessages.ts, 🔎 e "Erreur Slimbk errorCode", 🔎 e "Usages type runRef live", 🔎 e "Schéma runRef live", 🔎 e "Construction runRef live", 🔎 e "Run refs frontend live", EasyAgentsBatchLastViewerStop.service.ts, 🔎 e "Usages param close live", 🔎 e "Usages param open live", 🔎 e "Usages ligne EasyAgents", 🔎 e "Usages liveSessionStream", 🔎 e "Mocks locator Prompt", 🔎 e "Commande compact contexte", 🔎 e "Bus streamed extension", 🔎 e "Émission stream bus", 🔎 e "Implémentation bus typé", 🔎 h "Flux webview dispose", 👉 "Routage stream extension", extMessageBus.util.ts [L300-L430], extMessageBus.util.ts [L1-L220], 🔎 e "Construction child executor", usePromptRunController.hook.ts [L1-L180], usePromptRunController.hook.ts [L180-L440], 🔎 e "Usages queryState windows", 🔎 e "Fixtures windows activité", PromptQueryDoneSound.manager.ts [L100-L260], PromptQueryDoneSound.manager.ts [L240-L300], stream.endpoint.ts, ChatSessionFile.type.ts [L1-L140], 🔎 e "Schéma session chat", 👉 "Validation session chat", SidebarStore.chatTitles.manager.ts, 🔎 e "Mise à jour titre activité", SidebarUI.port.ts, 🔎 e "Réhydratation fenêtres query", usePromptRunController.toolUseRouter.hook.ts [L1-L220], frontSessionToPromptMessages.util.ts [L1-L160], 🔎 e "Meta session hydratation", usePromptLiveSession.hook.ts, 🔎 e "Persistance brouillon Prompt", 👉 "Brouillon pendant live", usePromptRunController.uiState.hook.ts, usePromptRunUiDerived.hook.ts, 🔎 e "Verrou PromptInput", Prompt.tsx [L1000-L1115], 🔎 e "Usage isLocked input", PromptInput.tsx [L1-L260], 🔎 e "VscTextarea Prompt", 👉 "Composer PromptInput", 🔎 e "Ownership cleanup frontend", 👉 "Registration cleanup", LiveSessionService.ts [L100-L220], 🔎 e "Tests payload aiStream", 🔎 e "Mocks run controller", usePromptCancelController.hook.ts

🔧 ✅ *scanning git*: ᛘ "Base commune b42 HEAD", ᛘ "Diff fondation tranche 1", ᛘ "Écarts b42 vers HEAD", ᛘ "Écarts base vers HEAD"

🔧 ✅ *reading project*: 🔎 e "Usages registerRun live", 🔎 e "Usages markSourceSettled", 🔎 e "Usages markHandoffReady", 🔎 e "Usages markBaselineReady", 🔎 e "Usages releaseViewerLease", 🔎 e "Usages view model live", 🔎 e "Usages mode live Prompt", 🔎 e "Usages OpenResult live", AiQuery.type.ts [L1-L220]

🔧 ✅ *specifying* *"SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_175152"*

SPECIF_FINALISER_LIVESESSION_POUR_LES_SESSIONS_PROMPT_NORMALES_20260712_175152

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 propre courant au commit `c3d66083b46b178e24d4ce497cc3cd5d914a4170` (`c3d66083`). Les commits `b42c734c4178d0725ce98933c085a763288eb697`, `613acafc9ac510e432986f56456d63f185d77347`, `b16c5bdde3b4a27cec2ee3f44c9a471b44465ac6` et `0b9f52cc402ce6d0ae19372fa62a432851c248b0` ne sont pas les ancêtres directs du HEAD courant : leur base commune avec le HEAD courant est `ed3bd94687cc34f6603730689bfc775818881bae`. Le diff complet entre `f65adc8e5172cc1e4f9781bf54ffa6dd089d3d33` et `ed3bd94687cc34f6603730689bfc775818881bae` est vide. Les écarts entre `b42c734c` et le HEAD courant concernent uniquement l'outillage de génération/raffinement des

spécifications ; les écarts supplémentaires depuis `ed3bd946` concernent le même outillage et une documentation personnelle sans rapport avec cette tranche. Aucun arbre source ciblé ici n'a changé. La base fonctionnelle `liveSession` reste donc exactement celle de la tranche 1, mais la référence correcte au dépôt courant est désormais `c3d66083`.

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 ;
- côté extension, un descripteur live explicite invalide est rejeté avant l'appel au low-level opener ; la validation frontend reste une défense en profondeur pour les webviews injectées directement, corrompues ou testées hors du coordinateur ;
- la détection frontend des paramètres liveSession s'effectue à partir d'un snapshot brut unique de la query injectée, avec `URLSearchParams.getAll(...)` pour chaque paramètre critique ; elle ne doit jamais reposer sur `useQueryString(...)`, qui ne permet pas de détecter les doublons ;
- `Prompt.tsx` est l'unique propriétaire du snapshot brut figé utilisé pour la classification live. Ce snapshot est obtenu par un hook dédié, stable pour toute la durée de vie du panneau ; le dispatcher de bootstrap reste une fonction pure recevant ce snapshot et ne relit jamais la globale router ;
- `Prompt.tsx` sépare une frontière bootstrap externe d'un runtime Prompt résolu interne : une query live invalide dont le locator n'est pas univoque affiche/ferme l'erreur sans instancier le contrôleur de run, sans fabriquer de locator sentinelle et sans violer les règles React des hooks ;
- l'ensemble privé exact comprend `openMode`, `liveSessionTurnId`, `liveSessionViewerLeaseId`, `liveSessionClosePolicy`, `liveSessionSourceId`, `liveSessionAdapter` et `easyAgentsBatchLineNumber`; la constante `easyAgentsBatchLineNumber` et la liste des clés privées connues sont définies dans le contrat shared, toute

clé inconnue préfixée `liveSession` est invalide, et tout doublon d'un paramètre critique, y compris `sessionId` ou `storageScopeRelPath` dans une query live explicite, est rejeté ;

- toute query live explicite, valide ou invalide, désactive également tout auto-run frontend ; le nettoyage des paramètres `autoRun*` lors de l'injection normalPrompt reste une défense supplémentaire et non l'unique barrière ;
- `sourceId` identifie la source backend exacte, tandis qu'un `streamInstanceId` frontend-only distingue chaque tentative de stream acceptée ; un retry qui obtient une branche `live` et démarre effectivement un nouveau transport reçoit un nouveau `streamInstanceId`, tandis qu'un retry dont l'Open retourne directement `terminal` ne fabrique aucun identifiant de stream inexistant ;
- chaque séquence initiale ou manuelle d'Open liveSession possède en plus un `replayOpenAttemptId` frontend-only monotone, distinct de `streamInstanceId`, qui garde les réponses Open, les délais `not_ready`, les retries et l'unmount. Cet identifiant ne traverse aucun contrat backend ;
- `streamInstanceId` remplace l'actuel identifiant de listener `streamKey` : il n'existe pas deux identités frontend concurrentes pour la même tentative de stream ;
- l'acceptation d'un start frontend est une transaction synchrone : `useAiStream.start()` retourne une union stricte acceptée/refusée ; pour une source Prompt, `sourceId` et `streamInstanceId` ne sont générés que sur la branche acceptée ; pour un observer, le `sourceId` exact est fourni par le runRef et seul le `streamInstanceId` est généré après acceptation ; aucun scaffold Prompt, `uiPending`, reset de queue, placeholder, base thinking ni autre mutation visible ne doit être conservé lorsqu'un second start du même tick est refusé ;
- les handlers de commande Prompt propagent synchroniquement ce résultat d'acceptation jusqu'à `Prompt.tsx`; les barrières locales `pendingClearAfterFallbackAckRef`, `submitTxnPending` et la restauration de focus ne sont armées qu'après une acceptation, tandis qu'un éventuel flip optimiste du gate UI est annulé dans la même pile sur refus ;
- chaque chemin de commande qui peut démarrer un stream — Send, SPECIF, Retry, Edit, Compact context, Reply et Continue — est couvert par un test transactionnel commun prouvant qu'un refus ne laisse aucune mutation visible et qu'une acceptation committe exactement un scaffold ;
- 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` ;
- `outcome='completed'` ou `outcome='transportError'` n'est publié, et le verrou synchrone d'acceptation d'une nouvelle tentative n'est libéré, qu'après drainage de Queue A, drainage de Queue B et dernière barrière de commit ; un retry ne peut donc jamais accepter une nouvelle tentative pendant que Queue B conserve des chunks différés de la précédente ;
- si un vrai terminal de données a déjà été consommé pour l'instance courante puis qu'une erreur de transport ou une erreur auxiliaire agrégée arrive, `useAiStream` conserve ce terminal exact, publie `outcome='transportError'` et n'injecte jamais un second `meta.error` SYSTEM_ERROR ni un second suffixe. Le SYSTEM_ERROR frontend canonique n'est créé par `onError` que lorsqu'aucun vrai terminal n'a déjà été observé pour cette instance ;
- le handoff direct `normalPrompt` 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` ; lorsque ces conditions sont réunies, le hook passe directement à `normal` sans rappeler `/api/ai/orch/liveSession/open` et sans réhydrater une seconde fois le compositeur ;
- 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 header privé ne s'applique que lorsque le tout premier chunk produit par la source est un `meta.error`, car le header Slimbk est immuable après l'émission du premier résultat ; tout `meta.error` produit après un premier chunk non-erreur reste un chunk de body ordinaire sous un header `200_OK` et suit exclusivement la pipeline `onChunk` ;
- le protocole privé du premier header d'erreur utilise exactement `status='500_SERVER_ERROR'`, `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 ;
- lorsqu'un objet structurel expose un champ `status`, le décodeur privé exige `500_SERVER_ERROR`; lorsqu'il expose sa propre propriété `errorCode`, même avec la valeur `undefined`, celle-ci doit être exactement `AICODE_LIVE_SESSION_SOURCE_ERROR_V1`. Seuls une chaîne ou un `Error` réellement dépourvus de ces champs peuvent utiliser la voie de secours par préfixe seul ;

- une erreur de transport non encodée ou une complétion sans vrai terminal déclenche la récupération et ne simule jamais un passage à `normal` ;
- tout terminal synthétisé par le wrapper normalPrompt après un throw ou un EOF ne l'est que si aucun vrai terminal n'a déjà été observé ; il doit être persisté sur l'assistant exact, avec son suffixe canonique, avant d'être appendu au registre ou exposé ; si cette persistance exacte échoue ou si aucun assistant exact ne peut être patché, le wrapper ne fabrique pas de terminal handoffable, laisse le transport échouer et la finalisation retire le record comme `retiredMissingTerminal` ;
- 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 ;
- une réservation de source revendique immédiatement le pointeur actif du locator et avance son époque avant tout `await`; sa promotion par `registerRun` conserve cette même époque, tandis que son annulation, un handoff ou une retraite retirent le pointeur puis mémorisent l'issue avec l'époque post-retrait. Une répétition immédiate voit donc l'issue courante, mais toute source ultérieure rend l'ancienne finalisation `stale` ;
- une seconde réservation du même runRef exact est refusée et ne devient jamais une réussite idempotente : un seul handler possède la réservation, et seul ce handler peut promouvoir ou finaliser la source qu'il a effectivement réservée ;
- les retraites structurées utilisées pour le fallback terminal et l'idempotence de finalisation restent distinctes des tombstones exactes anti-résurrection ; une éviction destructive peut retirer les retraites replayables, mais elle ne retire jamais les tombstones des run keys déjà observées ;
- lorsqu'un record manque simultanément baseline et terminal, `retiredMissingBaseline` est l'issue déterministe prioritaire ;
- cette finalisation post-forwarding est réservée aux sources `ownership='source'` ; le settlement/handoff EasyAgents reste piloté par son adaptateur métier historique ;
- le registre expose un événement process-local profondément immuable « run observer devenu inactif » ; il est émis seulement après la mise à jour complète des index, leases, pointeur actif et époque, et sert uniquement de wake-up au coordinateur pour terminer un cleanup différé après `markHandoffReady` ou éviction EasyAgents, sans introduire de dépendance du sous-système EasyAgents vers les panneaux ;
- le registre expose également un événement profondément immuable « claim destructif de locator libéré ». Il est émis uniquement après retrait du claim, avancement de sa génération destructive et stabilisation des index ; il réveille le coordinateur pour reprendre un cleanup de panneau qui avait été différé pendant un delete/Retry, sans transporter de décision métier ;
- le résultat d'une finalisation exacte est idempotent tant que l'époque de retraite reste courante ; si une source plus récente a depuis avancé l'époque du locator, `stale` prend toujours priorité sur l'ancien résultat mémorisé afin qu'aucune répétition tardive ne puisse muter l'Activité ou les cleanups de la nouvelle source ;
- `continueResponse` réutilise la bulle thinking agrégée existante et son texte de base ; cette initialisation est corrélée au `streamInstanceId`, y compris lors d'un retry de 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 présents dans le panneau reconnecté ; cette application sélective vaut aussi lorsque le retry reçoit directement une branche `terminal` ;
- l'application sélective de l'hydratation est un contrat typé explicite et discriminé, et non un détournement par setters no-op : la branche de préservation ne lit, ne normalise, n'enregistre ni n'applique les champs de compositeur du snapshot (`promptDraft`, `promptDraftAttachments`, `meta.nextTurnId`) ; elle applique seulement les messages, les stores d'hydratation, le thread, le reset de tentative et le scroll ;
- le parseur frontend de la réponse Open est lui aussi policy-aware : le schéma canonique complet conserve le contrat de validation actuel de `ZChatSessionFile` et la corrélation stricte des branches, sans resserrer globalement ni rendre deep-strict les messages opaques ou les métadonnées historiques ; la branche `preserveLocal` commence par un schéma d'enveloppe strict qui valide l'identité et laisse `session` en `unknown`, puis confie uniquement les champs consommés au validateur sélectif ; aucun parse complet antérieur ne doit réintroduire la validation des champs de draft ignorés ;
- le helper d'application d'un snapshot retourne `void` et construit un snapshot frontend assaini à partir des seuls champs validés/consommés ; la branche `preserveLocal` ne caste jamais une enveloppe contenant des champs de draft non validés vers `FrontChatSession` ;

- 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 ;
- le callback Slimbk `onError` du handler normal ne dépend jamais d'un throw interne pour piloter le résultat : il mémorise de façon idempotente soit l'erreur transport normalisée, soit le terminal source décodé et enfilé ; la décision finale est prise après le retour du client et le drainage de la FIFO, ce qui évite les doubles callbacks ou doubles agrégations du client généré ;
- une erreur auxiliaire sur un chunk de cette FIFO est mémorisée dans l'ordre mais ne court-circuite ni les étapes restantes du même chunk ni les chunks suivants : l'Activité, le son, le forwarding frontend et le terminal ultérieur doivent tous être tentés avant l'agrégation finale ;
- l'allocation du `activityRunId`, l'installation du timer et l'identité du runtime Activité sont synchrones. La publication/broadcast Activité qui suit est une tâche auxiliaire attendue et agrégée, mais son échec ne doit ni perdre le `activityRunId` exact ni empêcher la source backend de démarrer ;
- pour une source Prompt normale, `sourceSettled` et `handoffReady` ne sont pas publiés par l'endpoint backend avant que le handler extension ait drainé cette pipeline ;
- la finalisation normale du registre est atomique et exécutée par le handler extension après le drainage ; elle retourne un résultat strict distinguant `reservationCancelled`, `handoffReady`, `retiredMissingBaseline`, `retiredMissingTerminal` et `stale`, afin que l'Activité, le coordinateur et les tests ne déduisent jamais l'issue à partir d'un état partiel ;
- le coordinateur de panneau est l'unique propriétaire de la transition Activité terminale d'une source Prompt normale : le handler lui transmet le `runId` d'Activité lorsqu'il a été alloué et le résultat strict après avoir arrêté la tentative sonore ; le coordinateur sérialise alors `sourceSettled` avec les événements open/close du même locator, puis exécute les cleanups. Le handler ne publie pas une seconde transition Idle/suppression en parallèle ;
- 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 leurs préchecks UI et registre ; le claim destructif reste cependant l'autorité atomique finale et refuse toute réservation ou tout record actif, quel que soit son ownership ;
- le démarrage extension d'une nouvelle source Prompt normale vérifie synchroniquement que le panneau exact est encore enregistré avant la réservation ; ce read-only guard précède la première mutation et empêche un message de start resté en file après le dispose de ressusciter une source sans panneau ;
- les suppressions durables utilisent un claim destructif process-local du locator : le claim est acquis synchroniquement avant la fermeture du panneau, échoue si une réservation ou un record actif existe, quel que soit son ownership, bloque toute nouvelle réservation ou registration observer pendant l'opération, puis est libéré dans tous les cas ;
- la génération destructive du locator est monotone et distincte de l'époque de run. Elle avance à l'acquisition et à la libération du claim, est capturée par chaque Open/permit et invalide tout Open en vol qui aurait traversé un cycle acquire/release ; elle ne rend pas obsolète une retraite terminale uniquement parce qu'un delete a échoué ;
- toute éviction publique destructive d'un locator exige le handle opaque exact du claim courant. Le registre dérive le locator depuis ce handle, refuse un handle stale ou étranger, n'auto-libère jamais le claim et conserve les tombstones ;
- lorsqu'un panneau est disposé pendant un claim destructif, le coordinateur retire le panneau mais diffère impérativement `Abort/Await/ClearMemory`, l'orphan cleanup, le cleanup du son et le cleanup trivial. Il conserve le mapping détaché ; l'événement de libération du claim réveille ensuite la queue du locator, qui revalide panneau et nouvelle source avant tout nettoyage. Cette barrière empêche le dispose de courir en parallèle avec le delete/Retry disque ;

- la suppression durable n’évince les états retenus qu’après succès disque, tandis que le Retry EasyAgents conserve son éviction destructive pré-delete historique, dans les deux cas avec le handle exact du même claim ;
- 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 ; après une suppression de chat réussie, tous les records/retraites/leases retenus du locator sont évincés afin qu’aucun replay ou fallback terminal process-local ne survive à la destruction durable ;
- le sous-arbre React de l’action Stop détachée est remonté par `sourceId` : une source plus récente du même locator ne doit jamais hériter du `forcedDisabled` interne laissé par `VscButton.disableAfterClick` sur l’ancienne source ;
- la disponibilité du Stop transport est distincte de `isStreaming` : un replay frontend peut être terminé en erreur tandis que la source backend reste encore abortable ; le cancel core reçoit donc à la fois l’état réel du transport et l’éligibilité exacte du Stop, et ne transforme jamais ce cas en simple annulation UI ;
- le modèle de présentation liveSession distingue explicitement la surface composer et le modèle de bannière : une bannière Retry normalPrompt peut coexister avec le compositeur, tandis que la bannière EasyAgents continue de le remplacer. Le composant de bannière ne dépend plus d’un type sémantiquement limité au viewer read-only.
  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 `c3d66083`; les arbres source ciblés sont inchangés par rapport à `b42c734c`, `613acafc`, `b16c5bdd`, `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`.
- `easyAgentsBatchLineNumber` et la liste canonique des paramètres privés liveSession résident dans le contrat shared afin que l'extension et le frontend appliquent les mêmes règles.
- `Prompt.tsx` obtient une seule fois la query brute au moyen de `usePromptRouterQueryStringRawSnapshot()`, puis transmet ce snapshot au dispatcher pur ; le dispatcher strict utilise `URLSearchParams.getAll(...)`, retourne aussi le locator/classification nécessaires au Prompt et ne relit jamais la globale router.
- Une frontière React externe traite les classifications invalides avant le montage du runtime Prompt résolu ; aucun locator factice ni hook conditionnel n'est autorisé.
- Les hooks scalaires `useQueryString(...)` restent réservés aux paramètres non-live et ne participent ni à la validation liveSession, ni au choix classic/live, ni à la validation du locator d'une query live explicite.
- `streamInstanceId` reste frontend-only, change à chaque tentative de stream acceptée, sert directement de clé `useStreamListener` et ne traverse aucun contrat backend.
- `replayOpenAttemptId` reste frontend-only, change à chaque séquence Open initiale ou Retry, garde les réponses/timers Open et ne représente jamais un stream qui n'a pas été démarré.
- `useAiStream.start()` retourne une union stricte `{ accepted: false } | { accepted: true; runRef; streamInstanceId }`. Les callers n'installent le scaffold Prompt et les refs de continuation qu'après la branche acceptée ; une branche refusée annule tout état optimiste déjà armé par le gate UI.
- Les commandes Prompt qui lancent un stream retournent ce même résultat synchroniquement au caller. `Prompt.tsx` n'arme les barrières locales de clear/focus qu'après `accepted: true`, avant que React ne puisse exécuter l'effet transport.
- `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 résultat d'une tentative n'est terminal côté frontend qu'après `streamQueue.whenIdle()`, `dispatchGateQueue.whenIdle()` et la barrière de commit ; le ref d'acceptation reste occupé jusque-là.
- Un `onError` reçu après un vrai terminal de la même tentative ne crée pas de terminal frontend supplémentaire ; il change seulement l'outcome en `transportError`.
- 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, revendique le pointeur actif et avance l'époque du locator.
- Avant cette première mutation, le handler vérifie que le panneau Prompt exact est toujours enregistré ; une demande arrivée après le dispose est rejetée sans réservation, sans Activité et sans source backend.
- Une réservation exacte existante n'est jamais réacquise avec succès par un second handler ; la duplication est rejetée afin qu'une même identité source ne puisse lancer deux appels endpoint.
- Une réservation n'est pas replayable et ne viole pas « seul `registerRun` crée un record » : pour `ownership='source'`, `registerRun` promeut atomiquement la réservation exacte et retourne l'état d'abort précoce à appliquer.
- 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 et, séparément, des tombstones exactes process-lifetime. Seule une retraite terminale dont l'époque post-retrait reste courante autorise un fallback absent-record ; les tombstones interdisent la résurrection du même run exact même lorsqu'une retraite replayable a été supprimée.

- Une finalisation répétée rend le même résultat terminal uniquement tant que la retraite reste dans l'époque courante ; dès qu'un nouveau run est devenu courant, le résultat est `stale` même si une retraite plus ancienne mémorise une issue non-stale.
- `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. Ce lifecycle est injecté dans le child executor EasyAgents et dans le service source normalPrompt ; l'adaptateur EasyAgents liveSession reste un adaptateur de policy/registre et ne dépend pas du lifecycle de consommation du Readable.
- 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 au démarrage. Il alloue synchroniquement son `activityRunId` et son runtime timer avant d'attendre la publication Sidebar ; une panne de cette publication reste auxiliaire et n'empêche pas le provider de démarrer. Son forwarding ordinaire et le terminal source décodé depuis un header 500 encodé passent par la même pipeline FIFO. La transition Activité terminale est ensuite sérialisée par le coordinateur avec les événements de panneau.
- Chaque chunk normal est appendu au registre avant exposition.
- Si le premier chunk produit est un `meta.error`, il est appendu tel quel au buffer, puis l'endpoint place le statut 500, le code privé V1 et l'encodage préfixé V1 dans le header. Si un chunk non-erreur a déjà provoqué le header 200, tout `meta.error` ultérieur est envoyé dans le body comme donnée ordinaire et ne repasse jamais par l'encodage de header.
- Le handler extension reconnaît strictement l'encodage privé 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 wrapper synthétiques ne sont appendues qu'en l'absence de vrai terminal déjà observé et après patch durable de l'assistant exact. `patchLastAssistantMeta` garantit désormais le suffixe canonique pour `SYSTEM_ERROR` comme pour `INTERRUPTED`. Si cette persistance ne peut pas être prouvée, le wrapper ne produit pas de terminal de données.
- Les erreurs de transport ordinaires non encodées restent canonisées côté frontend en `SYSTEM_ERROR` uniquement si aucun terminal de données n'a déjà été reçu et donnent toujours `transportError`. La finalisation retire exactement tout record qui ne possède pas de vrai terminal après drainage, afin qu'aucun replay ne puisse attendre indéfiniment un `handoffReady` impossible.
- Les throws source après baseline et sans terminal deviennent un `meta.error` canonique `SYSTEM_ERROR` seulement après sa persistance exacte ; un EOF sans vrai terminal devient un `meta.error` canonique `INTERRUPTED` selon la même règle ; les deux sont ensuite 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 est rejetée et 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, run actif et claim destructif immédiatement avant chaque étape destructive et après chaque `await` de cleanup.

- Le mapping cleanup Prompt est indexé par locator structurel complet.
- Le coordinateur s'abonne au signal profondément immuable d'inactivité des records observer ; lorsqu'un Prompt EasyAgents fermé avait différé son cleanup, `markHandoffReady` ou l'éviction réveille la queue du locator, qui revalide panneau et nouveaux runs avant tout nettoyage.
- Le coordinateur s'abonne aussi à la libération des claims destructifs. Un dispose survenu sous claim ne nettoie jamais mémoire, attachments, son ou session disque en parallèle du delete/Retry ; la libération du claim réveille la queue, qui revalide avant de reprendre le cleanup différé.
- 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, tous deux gardés par `replayOpenAttemptId`. 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.
- Le handoff direct normalPrompt ne rappelle aucun endpoint Open : il conserve le snapshot de baseline déjà installé, le replay déjà appliqué et le compositeur local, puis déverrouille seulement les politiques `normal` après l'accusé de finalisation de la tentative exacte.
- Un retry de replay recharge la baseline des messages et les stores d'hydratation, tout en préservant les champs du compositeur déjà présents localement, y compris si Open répond directement `terminal`. Cette branche terminale est gardée par `replayOpenAttemptId` et ne crée pas de `streamInstanceId`.
- Le parse de ce retry utilise l'enveloppe Open sélective avant toute validation complète de session, puis la branche typée `preserveLocal` de `applyHydratedPromptSessionState`; aucun setter no-op n'est utilisé et aucun champ de draft ignoré n'est lu ou casté comme validé.
- Le schéma Open canonique complet conserve la profondeur de validation actuelle du projet ; cette tranche ne rend pas `ZChatSessionFile` ou ses messages opaques globalement plus stricts et n'introduit pas de régression sur les métadonnées historiques.
- 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 frontend distingue `isStreaming` de `transportStopAvailable` : en `liveSessionReplayFailed`, un transport terminé en erreur conserve la disponibilité du Stop exact tant qu'aucune preuve terminale/not-found ni aucun premier Stop ne l'a consommée ; une complétion replay normale, un Open terminal/not_found, un reset ou le premier Stop la suppriment.
- 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. Une suppression ou un Retry acquiert un claim destructif atomique avant de fermer le panneau ; ce claim ferme la course entre le dernier check et les opérations disque.
- Le claim destructif est aussi la barrière de cleanup : le dispose du panneau est mémorisé mais ses cleanups attendent la libération du claim, ce qui évite toute concurrence avec les suppressions disque.
- `continueResponse` réutilise une bulle assistant et une bulle thinking agrégées 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.
- La clé React du sous-composant contenant le Stop détaché inclut `sourceId`, afin que le verrou interne `disableAfterClick` soit recréé pour chaque source exacte.

- Le composant `PromptLiveSessionBanner` consomme un modèle de bannière dédié, indépendant de la décision `showComposer`, afin de supporter proprement viewer EasyAgents et composer normalPrompt + bannière.
- 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. préparation pure de la commande et du turn, sans mutation visible ;
2. acceptation synchrone du start frontend ; la branche refusée s'arrête avant toute mutation Prompt durable et restaure dans la même pile les éventuels flags optimistes ;
3. génération de `sourceId` et `streamInstanceId`, retour de l'identité acceptée au caller, puis installation synchrone du scaffold Prompt et des refs corrélées avant les effets transport ;
4. armement, uniquement après acceptation, de la barrière locale de clear après `fallback_persisted`, du masque transactionnel et de la restauration de focus ;
5. message bus vers l'extension ;
6. vérification synchrone read-only que le panneau exact est encore ouvert ; si le panneau a déjà été disposé, le start est rejeté avant toute mutation extension ;
7. réservation exacte `ownership='source'` comme première mutation synchrone ; cette réservation revendique le pointeur actif et avance l'époque du locator ; une réservation exacte déjà possédée par un autre handler est un conflit ;
8. allocation synchrone du runtime Activité, de son `activityRunId` et de son timer ; publication Activité Running/source comme tâche auxiliaire dont l'échec est mémorisé sans empêcher la source ;
9. armement du son exact et de son identifiant interne de tentative ;
10. persistance éventuelle de la préférence SPECIF Code ;
11. chargement catalogue ;
12. capture synchrone et défensive des messages de continuation dans la section critique de démarrage, si nécessaire ;
13. retour réussi de `askStream(...)` ;
14. priming du `Readable` ;
15. promotion réservation → record par l'unique opération `registerRun`, sans nouvelle avancée d'époque, et application d'un abort précoce ; pour `continueAssistant`, publication immédiate de la baseline messages ;
16. append de chaque chunk avant exposition ; pour `replaceTurnAssistants`, le chunk `fallback_persisted` rend la baseline prête ;
17. replay possible ;
18. vrai terminal Orch déjà durable, ou terminal wrapper synthétique seulement s'il n'existe encore aucun vrai terminal et après persistance sur l'assistant exact ;
19. fin de consommation backend sans retrait du run actif ;
20. retour Slimbk et drainage de la pipeline FIFO extension ; si le tout 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 ; un `meta.error` reçu après un premier chunk normal reste dans la voie body/onChunk ;
21. finalisation atomique post-forwarding : handoff si baseline + vrai terminal durable, sinon retraite/éviction exacte ; la retraite mémorise l'époque post-retrait et une répétition tardive après changement d'époque retourne `stale` ;
22. arrêt de la tentative sonore exacte ;
23. délégation au coordinateur de panneau, qui sérialise la transition Activité terminale et le cleanup avec les événements open/close du locator ; le `activityRunId` reste disponible même si la publication initiale de l'Activité a échoué ;
24. finalisation frontend de la tentative courante, après drainage Queue A/Queue B, avec publication de `finalizedStreamInstanceId` ;
25. pour normalPrompt connecté, passage direct à `normal` sans second Open ; pour EasyAgents ou un bootstrap qui reçoit déjà `terminal`, hydratation du snapshot terminal selon la policy produit ;
26. Activité Idle si panneau ouvert, suppression si panneau fermé ; un résultat stale ne touche jamais l'activité d'une source plus récente ;
27. cleanup mémoire/attachments/son/trivial uniquement sans panneau ni nouvelle source et hors claim destructif.

Replay, erreurs, baselines et application sélective
`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 conditionné par la position du chunk :

- si le premier chunk produit par la source est `meta.error`, le service source append le chunk original au buffer ;

- l’endpoint, qui n’a encore envoyé aucun header,
  envoie `500_SERVER_ERROR`, `errorCode='AICODE_LIVE_SESSION_SOURCE_ERROR_V1'` et `errorMessage='AICODE_LIVE_SESSION_SOURCE_ERROR_V1:' + JSON.stringify({ version: 1, value })` ;
- si un chunk non-erreur a déjà déclenché le header `200_OK`, un `meta.error` ultérieur reste un body chunk ordinaire et n’utilise jamais le protocole header ;
- le client généré transmet le header non-200 au callback `onError` sous forme de `SlimbkCallError` et n’attend pas les callbacks async ;
- le handler extension tente le décodage strict avant toute normalisation générique ; si l’objet expose `status` ou sa propre propriété `errorCode`, ces champs doivent correspondre exactement au protocole ;
- 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 ;
- si le payload n’est pas valide, le callback mémorise une erreur transport normalisée sans dépendre d’un throw du client généré ;
- l’appel Slimbk est ensuite considéré selon l’état mémorisé après drainage, ce qui permet au frontend d’observer le vrai terminal via `onData` puis une complétion transport normale pour le protocole privé valide ;
- 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 seulement en l’absence d’un terminal data déjà reçu ;
- un EOF source sans terminal tente d’abord de persister un `INTERRUPTED` exact sur le dernier assistant du turn ; seulement après cette barrière il est appendu et suit le chemin de données terminales normal ; si la barrière échoue, aucun terminal n’est appendu et le chemin devient `transportError + retiredMissingTerminal` ;
- un throw après un vrai terminal déjà observé ne synthétise jamais un second terminal : l’erreur reste transport, le record peut être handoffable côté backend, mais le frontend n’effectue pas de handoff direct sans complétion transport normale et ne remplace pas le terminal déjà reçu par un SYSTEM_ERROR local.
  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.
  Pour normalPrompt, le replay déjà appliqué constitue l’état autoritatif du panneau reconnecté. Le passage à `normal` n’effectue donc pas de deuxième Open lorsqu’un terminal data exact, `outcome='completed'`, le drainage des deux queues et `finalizedStreamInstanceId === streamInstanceId` sont tous prouvés. Un Open terminal reste utilisé pour un bootstrap qui arrive après convergence et pour les policies produit qui exigent le snapshot terminal, notamment EasyAgents.
  Le contrat Open expose deux niveaux de validation complémentaires :
- `ZLiveSessionOpenResult` et `ZLiveSessionOpenResultTransport` restent les schémas canoniques complets utilisés par le backend et les hydratations complètes. Leur corrélation de branche/propriétés est stricte, mais la tranche conserve volontairement le contrat actuel de `ZChatSessionFile` et des messages opaques ; elle n’introduit pas de resserrement deep-strict global ;
- `ZLiveSessionOpenResultEnvelope` valide strictement le discriminant, les propriétés présentes, `threadId`, `turnId`, `sourceId` et `replayMode`, mais garde `session` en `unknown` ; il est réservé au chemin frontend policy-aware ;
- une hydratation complète reparcourt ensuite le schéma canonique complet puis le validateur frontend historique ;
- une application `preserveLocal` transmet le `session` brut au validateur sélectif sans parse complet préalable.
  Le helper d’application de snapshot expose une union discriminée :
- `composerStatePolicy='hydrateFromSnapshot'` applique les messages, stores, `promptDraft`, `promptDraftAttachments`, `meta.nextTurnId`, thread, reset et scroll ;
- `composerStatePolicy='preserveLocal'` applique les messages, stores, thread, reset et scroll, mais ne reçoit pas les setters de compositeur, ne lit ni ne valide les champs de draft hors périmètre, ne normalise pas les attachments de draft du snapshot et ne les enregistre pas dans les stores de noms de documents ;
- les deux branches construisent un snapshot frontend assaini à partir des champs réellement validés ; la fonction retourne `void`, et la branche sélective ne produit jamais un cast mensonger vers un `FrontChatSession` complet contenant des champs ignorés.
  Chaque séquence Open initiale ou Retry incrémente `replayOpenAttemptId`. Les réponses, timers `not_ready` et continuations ne mutent l’état que si cet identifiant reste courant. Si la réponse est `live`, l’acceptation de `startLiveObserver` produit ensuite un nouveau `streamInstanceId`. Si la réponse est directement `terminal`, le snapshot est appliqué selon la policy sans créer de stream fictif.

Cette distinction évite qu'un retry normalPrompt échoue ou écrase l'état local à cause d'un champ d'enveloppe obsolète ou malformé qui n'appartient pas au périmètre de l'application sélective.

Claims destructifs, génération destructive et anti-résurrection

Le registre maintient quatre concepts distincts :

- une réservation/source active, qui revendique le pointeur du locator ;
- une retraite structurée, qui porte l'issue terminale et l'époque post-retrait nécessaires au fallback/idempotence ;
- une tombstone exacte de run key, process-lifetime, qui interdit la résurrection du même `{ locator, turnId, sourceId }` même si la retraite structurée a été supprimée ;
- un claim destructif de locator, accompagné d'une génération destructive monotone indépendante de l'époque de run.

Un claim destructif de locator est un handle opaque process-local :

1. son acquisition est synchrone et échoue si le locator possède une réservation ou un record actif, quel que soit l'ownership, ou un autre claim ;
2. l'acquisition avance la génération destructive afin d'invalider les Open déjà en vol ;
3. pendant le claim, `reserveSource`, les registrations observer et les Open/fallback du locator sont refusés ;
4. le propriétaire peut fermer le panneau et exécuter ses opérations disque sans course avec une nouvelle source ;
5. le dispose du panneau constate le claim, conserve son mapping détaché et diffère tous les cleanups susceptibles de toucher mémoire, attachments, son ou disque ;
6. toute éviction publique destructive reçoit le handle exact, en dérive le locator et refuse les handles stale/étrangers ;
7. le claim est toujours libéré dans un `finally` ; la libération retire le handle, avance de nouveau la génération destructive puis émet le wake-up immuable ;
8. le wake-up fait reprendre le coordinateur, qui revalide panneau, source et claim avant de nettoyer ;
9. une suppression de chat n'appelle l'éviction sous claim qu'après succès durable du delete ;
10. le Retry EasyAgents conserve son éviction pré-delete historique, mais uniquement sous le même claim exact ;
11. `evictLocator` retire réservations, records, leases et retraites replayables, réveille les subscriptions et conserve/ajoute les tombstones exactes des runs concernés ;
12. la génération destructive ne participe pas à l'éligibilité d'une retraite terminale après un delete avorté : elle ne sert qu'à invalider les opérations Open concurrentes.

Cycle de vie du panneau

Le wiring low-level compose le coordinateur, qui reçoit opener, panneau registry, liveSession resolver, abonnement aux inactivations observer, abonnement aux libérations de claims destructifs, releases adapter-specific, runtime 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 → rejet avant low-level open ; si une webview invalide existe malgré tout, aucun fallback ni cleanup destructif d'un run actif.

Lorsqu'un panneau EasyAgents fermé a différé son cleanup parce que son record observer est encore actif, le coordinateur conserve l'état détaché. Le registre émet ensuite un événement immuable lorsque ce run observer quitte l'index actif par handoff ou éviction. Cet événement ne transporte aucune décision de cleanup : il réveille seulement la queue du locator, qui revalide panneau, source plus récente et mapping avant de nettoyer.

Lorsqu'un panneau est fermé par une suppression ou un Retry sous claim destructif, le coordinateur conserve également l'état détaché mais n'exécute aucun cleanup concurrent. La libération du claim produit un second type de wake-up générique ; la queue du locator reprend alors le cleanup historique ou terminal uniquement si aucun panneau ni nouvelle source n'existe.

Pour une source Prompt normale, le handler ne publie pas directement l'état terminal de l'Activité. Après finalisation du registre et arrêt du son exact, il appelle `onSourceSettled` du coordinateur avec `runRef`, `activityRunId` lorsqu'il a été alloué et résultat strict. Le coordinateur sérialise cette transition avec `openPromptPanel` et `onPanelDisposed`, revalide l'état réel du panneau, demande au runtime Activité de convertir vers Idle ou de supprimer, puis décide le cleanup.

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

La fonction `Prompt` externe lit une fois le snapshot brut et le classe. Si le locator est absent ou non univoque dans une query live explicite invalide, elle rend un petit garde d'erreur/fermeture qui n'instancie aucun hook métier du runtime Prompt. Seule une classification classique ou live valide monte le sous-composant runtime interne avec un locator strict. 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 incrémente `replayOpenAttemptId`. S'il obtient une branche `live`, le nouveau transport accepté possède un nouveau `streamInstanceId`; s'il obtient directement `terminal`, aucun stream n'est créé. Les callbacks antérieurs, les continuations async, les réponses Open stale et les chunks déjà sortis de Queue A ne peuvent plus muter le runtime. La tentative courante ne publie son outcome et ne libère le single-flight qu'après Queue A, Queue B et la dernière barrière React. 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. Il passe alors à `normal` sans nouvel appel Open et sans remplacer le brouillon, les attachments ou `nextTurnId`. 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 tant que `transportStopAvailable` est vrai 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 au moyen de `composerStatePolicy='preserveLocal'` et conserve le brouillon, les attachments et `nextTurnId` locaux, y compris lorsque la nouvelle réponse Open est déjà terminale.

Toute query live explicite court-circuite l'effet auto-run, même si elle contient encore des paramètres `autoRun*`. L'injection normalPrompt les retire en amont, mais le frontend ne dépend jamais de ce nettoyage pour sa sûreté.

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 sous-composant ou le bouton détaché porte une clé React contenant `sourceId`, de sorte que le verrou interne de l'ancienne source ne survive jamais à une source successive du même 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 runtime Activité fournit une opération de début en deux temps : allocation synchrone `{ runId, runtime }`, puis promesse de publication. Cela permet au handler de conserver l'identité exacte, de continuer la source si le broadcast Sidebar échoue et de terminer le timer/runtime exact lors de la coordination finale.

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.

La pipeline source continue après une erreur auxiliaire d'un chunk : elle mémorise cette erreur, tente individuellement les autres étapes du chunk, puis les chunks suivants, attend la totalité, finalise le registre, arrête le son, délègue au coordinateur la transition Activité/cleanup puis agrège les erreurs dans l'ordre. Si cette agrégation transforme finalement le transport en erreur après qu'un vrai terminal a déjà atteint le frontend, l'outcome devient `transportError` sans création d'un second terminal frontend.

Liste exacte des fichiers avec détails d'implémentation

Contrats shared

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

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

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

- Ajouter exactement `LIVE_SESSION_SOURCE_ID_PARAM='liveSessionSourceId'` et `LIVE_SESSION_ADAPTER_PARAM='liveSessionAdapter'`.

- Déplacer/centraliser également `EASY_AGENTS_BATCH_LINE_NUMBER_PARAM='easyAgentsBatchLineNumber'` dans ce contrat shared ; supprimer sa déclaration locale frontend.
- Ajouter le schéma strict adaptateur `easyAgentsBatch | normalPrompt`.
- Réutiliser le schéma `sourceId`; rendre `continueOnPanelClose` effectif pour normalPrompt.
- Documenter les couples valides : EasyAgents + `stopOnLastViewerClose`; normalPrompt + `continueOnPanelClose`.
- Centraliser la liste des paramètres privés connus afin que les parseurs extension/frontend appliquent exactement la même classification explicite.

4) `AICode/src/sharedlib/src/types/liveSessionTypes/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`.
- Conserver `ZLiveSessionOpenResult` et `ZLiveSessionOpenResultTransport` comme schémas canoniques complets pour les champs déclarés et la corrélation stricte des branches.
- Ne pas rendre globalement `ZChatSessionFile`, ses messages opaques ou ses métadonnées historiques deep-strict dans cette tranche ; la validation frontend complète existante reste complémentaire.
- Ajouter `ZLiveSessionOpenResultEnvelope` / `LiveSessionOpenResultEnvelope` : même corrélation stricte des branches et propriétés, mais `session` reste `unknown` afin que le frontend puisse appliquer une validation policy-aware sans toucher les champs draft hors périmètre.
- Interdire l'utilisation du schéma d'enveloppe comme `respSchema` backend : il est réservé à la lecture frontend sélective.

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

- Couvrir clés, collisions, noms exacts des query params, liste privée, constante ligne EasyAgents, adaptateurs, couples close-policy, replay modes, transports et sourceId obligatoire.
- Couvrir le schéma Open complet versus le schéma d'enveloppe sélectif : identité toujours stricte, draft malformé rejeté par le complet mais laissé opaque par l'enveloppe.
- Vérifier que la tranche ne resserre pas accidentellement la profondeur historique/opaque de `ZChatSessionFile` hors des champs explicitement consommés.

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

- Ajouter les fabriques canoniques des `meta.error` transport/source.
- Définir le protocole privé exact V1 : statut `500_SERVER_ERROR`, 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.
- Lorsqu'un objet possède `status`, exiger `500_SERVER_ERROR`; lorsqu'il possède sa propre propriété `errorCode`, même `undefined`, exiger le code V1. La voie préfixe-seul est réservée aux valeurs réellement dépourvues de ces champs.
- Valider 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) `AICode/src/sharedlib/src/types/aiTypes/__test__/aiStopSuffix.util.test.ts` — modifié

- Tester Error/string/status/errorCode, statut erroné, propriété errorCode présente mais absente/incorrecte, réversibilité, stacks/prefixes, détails multilignes, propriétés supplémentaires invalides, SYSTEM_ERROR, INTERRUPTED et identité du chunk.

Persistance terminale Orch

8) `AICode/src/extension/src/aiTools/orch/context/memory/core/msg/OrchMessagesMemoryAssistantPatch.ts` — modifié

- Rendre l'ajout de suffixe canonique symétrique pour `AiStopErrorReason.SYSTEM_ERROR` et `AiStopErrorReason.INTERRUPTED` lors d'un `patchLastAssistantMeta`.
- Conserver l'idempotence : ne jamais dupliquer un suffixe déjà présent dans le dernier bloc texte exact.
- Préserver tous les blocs, métadonnées, usages et hydratations existants ; seule l'erreur terminale exacte est ajoutée/patchée.
- Cette garantie est utilisée par le wrapper normalPrompt avant d'appendre un terminal synthétique au registre.

9) `AICode/src/extension/src/aiTools/orch/context/memory/core/msg/__test__/OrchMessagesMemoryAssistantPatch.errorSuffix.test.ts` — créé

- Couvrir SYSTEM_ERROR, INTERRUPTED, assistant sans texte, assistant avec texte, suffixe déjà présent et persistance sink.

- Vérifier que l’erreur patchée reconstruit le même texte et le même `meta.error` après hydratation.

Contrats Sidebar

10) `AICode/src/sharedlib/src/types/busTypes/ui/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`.
- Ne pas profiter de cette modification pour resserrer les schémas de session/chat historiques sans rapport avec l’Activité.

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

- Ajouter les RPC exacts `sidebar->ext:openQueryActivity(locator)` et `sidebar->ext:stopDetachedQuery(runRef) -> {accepted}`.

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

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

Backend Orch `liveSession`

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

- Documenter identité, réservations, revendication du pointeur actif, époque de réservation, readiness, replay, continuation, protocole header V1 limité au premier chunk, ownership, lifecycle, Activity, son, retraites, époque post-retrait, retrait exact des runs sans terminal, barrière post-forwarding, priorité stale après changement d’époque et distinction runRef/soundAttemptId/streamInstanceId/replayOpenAttemptId.
- Documenter la barrière de persistance précédant tout terminal synthétique, l’interdiction de synthétiser après un vrai terminal et le wake-up observer utilisé par le cleanup différé des panneaux EasyAgents fermés.
- Documenter la distinction retraite structurée/tombstone exacte, le refus des réservations dupliquées et le claim destructif de locator utilisé par delete/Retry.
- Documenter la génération destructive distincte de l’époque de run, l’invalidation ABA des Open, l’éviction publique exigeant le handle exact et le wake-up de libération de claim qui reprend les cleanups différés.

14) `AICode/src/extension/src/aiTools/orch/liveSession/registry/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`.
- Définir un handle opaque `LiveSessionLocatorDestructionClaim` utilisé pour sérialiser une suppression/Retry destructif sans exposer les structures mutables du registre.
- Ajouter aux permits/snapshots Open la génération destructive capturée, distincte de l’époque du run.
- Définir un événement profondément immuable de run observer devenu inactif, contenant le runRef exact et l’ownership, utilisé uniquement comme signal de réévaluation.
- Définir un événement profondément immuable de libération de claim destructif, contenant le locator exact et la génération destructive post-libération, utilisé uniquement comme wake-up du coordinateur.
- Étendre permits/subscriptions à sourceId et interdire les baselines incohérentes.
- Mémoriser l’issue terminale et l’époque post-retrait séparément de l’époque courante du locator.
- Définir les snapshots immuables de réservations/sources actives nécessaires à Activité, au coordinateur, à deletion et au Stop.
- Distinguer explicitement les retraites replayables des tombstones exactes anti-résurrection process-lifetime.

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

- Réservations/cancel exacts, transformation atomique vers record par `registerRun`, listings actifs génériques/source, existence et claims atomiques.
- La réservation source est la mutation qui revendique le pointeur actif et avance l’époque ; la promotion source par `registerRun` ne réavance pas l’époque et retourne `abortRequested`.
- Refuser une seconde réservation du même runRef exact ; aucune idempotence ne doit autoriser deux owners à poursuivre vers l’endpoint.
- Supprimer `canAbortSource` et exposer des claims exacts distinguant viewer observer et source normale.
- Ajouter `claimLocatorDestruction(locator)`, `releaseLocatorDestructionClaim(claim)` et `isLocatorDestructionClaimed(locator)` : acquisition synchrone, handle opaque exact, rejet si réservation ou record actif de

n’importe quel ownership, blocage des nouvelles réservations/registrations/Open pendant le claim, libération idempotente uniquement par le bon handle.

- Maintenir une génération destructive monotone séparée de l’époque de run ; l’avancer à l’acquisition et à la libération, l’exposer aux fences Open et ne jamais l’utiliser pour invalider une retraite terminale après un delete avorté.
- Faire exiger le handle courant par l’éviction publique : `evictLocator` dérive le locator du claim, refuse tout handle stale/étranger, n’auto-libère jamais le claim et conserve les tombstones.
- 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.
- Si baseline et terminal manquent ensemble, retourner déterministiquement `retiredMissingBaseline`.
- Après retrait du pointeur et bump d’époque, mémoriser l’issue avec cette époque post-retrait ; tant qu’elle reste courante, une répétition retourne l’issue mémorisée, puis retourne `stale` dès qu’un nouveau run avance l’époque.
- Retraites structurées et époques empêchant la résurrection d’un ancien terminal.
- Conserver en plus les tombstones exactes process-lifetime de chaque run key observée ; l’éviction destructive supprime les retraites replayables mais jamais ces tombstones.
- Exposer une souscription process-local aux transitions exactes où un run observer quitte l’index actif ; construire un snapshot profondément immuable, terminer toutes les mutations internes avant l’émission, isoler les erreurs des listeners et ne jamais exposer le record mutable.
- Exposer une souscription équivalente aux libérations réelles de claims destructifs ; émettre après retrait du claim et bump de génération, jamais sur une seconde release idempotente, et isoler les listeners.
- Émettre le signal observer sur handoff/éviction observer, mais pas à nouveau lors de la simple expiration TTL d’un record déjà inactif.
- L’éviction sous claim doit retirer réservations, records, leases et retraites retenues du locator, réveiller les subscriptions et avancer l’époque, sans relâcher le claim détenu par le caller et sans retirer les tombstones exactes.
- Préserver leases, retention et replay tranche 1.
- Conserver l’elapsed source depuis `sourceStartedAtMs`; ne pas imposer cet ancrage aux observers EasyAgents.

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

- Couvrir sources successives, réservations, pointeur/époque dès la réservation, promotion sans second bump, 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 le rejet d’une réservation exacte dupliquée et l’absence de mutation/finalisation par le handler non-owner.
- Vérifier l’idempotence des cinq résultats dans l’époque courante, l’époque post-retrait, la priorité `stale` après démarrage d’une source plus récente et le refus de cette finalisation sur un record observer.
- Vérifier l’ordre et l’unicité du signal d’inactivité observer, son émission après mutation complète, l’immuabilité profonde du payload ainsi que l’isolation d’un listener qui throw.
- Vérifier le claim destructif : acquisition/refus pour tout record actif, blocage reserve/register/Open, release exacte, génération destructive acquire/release, événement de libération après mutation complète et isolation des listeners.
- Vérifier qu’un Open fence capturé avant un cycle claim acquire/release devient stale sans modifier l’époque terminale du run.
- Vérifier que l’éviction publique échoue sans le handle exact, réussit sous le bon claim, ne libère pas le claim et conserve les tombstones.
- Vérifier l’éviction destructive complète d’un locator et le refus de réenregistrement d’un ancien run exact après suppression des retraites structurées.

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

- Exposer toutes les nouvelles opérations du registre, supprimer `canAbortSource`, conserver Open/Replay/release/sweep.
- Exposer le résultat strict de finalisation post-forwarding, les snapshots actifs/réservés, les claims exacts, le claim destructif de locator et les souscriptions aux inactivations observer et aux libérations de claims.
- Exposer l’éviction destructive uniquement à partir du handle opaque courant.

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

- Couvrir parcours normal, réservations, réouverture, sources successives, claims, finalisation, retraite stale, finalisation sans terminal, époque de réservation et forwarding du signal observer inactif.
- Couvrir forwarding du claim destructif, génération destructive, signal de release, éviction exigeant le handle, blocage des sources et conservation des tombstones après éviction.

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

- `not_ready` pour réservation exacte.
- Refuser immédiatement Open/fallback lorsque le locator possède un claim destructif.
- Capturer et revalider à la fois l'époque de run et la génération destructive autour de toute lecture persistante, y compris le fallback absent-record, afin qu'un cycle claim acquire/release en cours de lecture rende uniquement cette tentative Open `not_found` sans invalider une retraite terminale future.
- `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 type de retraite + époque post-retrait.
- Un run retiré faute de terminal/baseline ou évincé destructivement ne doit jamais être classé comme live ou handoff-ready.

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

- Couvrir réservation, baselineOnly, continuation/enveloppe récente, mismatch, sources mêmes turn, retraites, époque post-retrait, éviction destructive, retrait faute de terminal et état live avant finalisation post-forwarding.
- Couvrir Open refusé pendant un claim destructif puis de nouveau autorisé après release si le snapshot existe encore.
- Couvrir un Open différé commencé avant acquire/release : la tentative en vol devient `not_found`, puis un nouvel Open post-release retrouve correctement le snapshot/retraitement autorisé.

21) `AICode/src/extension/src/aiTools/orch/liveSession/replay/__test__/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.
- Adapter toute éviction publique de test à l'acquisition et au passage du claim destructif exact.

22) `AICode/src/extension/src/aiTools/orch/liveSession/source/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.

23) `AICode/src/extension/src/aiTools/orch/liveSession/source/__test__/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.

24) `AICode/src/extension/src/aiTools/orch/liveSession/source/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 via la promotion `registerRun`, 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.
- Si le tout premier chunk produit est `meta.error`, l'appendre inchangé puis le laisser à l'endpoint pour l'encodage du premier header V1 ; si un chunk précédent existe, le `meta.error` reste un chunk body normal.
- Suivre explicitement si un vrai terminal a déjà été observé. Pour un throw/EOF wrapper après baseline et sans terminal, construire SYSTEM_ERROR/INTERRUPTED, patcher d'abord durablement `errorReason`, `errorMessage`, `errorDetails` et le suffixe canonique sur le dernier assistant exact via OrchMemory, puis seulement append/yield.
- Si un vrai terminal existe déjà, ne jamais en synthétiser un second ; laisser l'éventuelle erreur ultérieure conserver sa nature transport.
- Si l'assistant exact est absent ou si la persistance échoue, ne pas append de terminal synthétique ; propager une erreur transport non encodée afin que la finalisation produise `retiredMissingTerminal`.
- Ne pas finaliser le registre ; garantir qu'un échec registration/promotion ne touche aucun tiers.

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

- Couvrir tous les query types, continuation, readiness immédiate/ack, append order, abort, premier error/header, error après premier chunk, throw, EOF, throw après vrai terminal, absence de settlement/handoff avant finalisation et isolation registration.
- Vérifier qu'un terminal synthétique est persisté avant append, que le suffixe est isomorphe après hydratation, qu'un vrai terminal n'est jamais doublé et qu'un échec de persistance n'ajoute aucun terminal au registre.

Adaptation EasyAgents

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

- `replayMode='replaceTurnAssistants'`, `sourceId=turnId`, timestamp.
- Rester un adaptateur mince de policy/registre ; ne pas injecter ni posséder le lifecycle Readable partagé.
- Supprimer seam `canAbortSource` au profit du claim viewer exact.
- Exposer les opérations nécessaires au Retry : présence source active/réservée, acquisition/release du claim destructif et éviction du locator exigeant le handle exact.
- Préserver settlement/handoff métier EasyAgents.
- Ne jamais importer le coordinateur de panneau : les wake-ups de cleanup sont fournis par les signaux génériques du registre lors du handoff/éviction observer et de la libération du claim.

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

- Vérifier identité, replayMode, timestamp, nouveau claim viewer, forwarding du claim destructif/autorité source, ordre historique et que les opérations de handoff/éviction restent déléguées au registre générique.
- Vérifier que l'éviction Retry transmet le handle opaque exact et que la release déclenche le wake-up générique sans dépendance vers les panneaux.

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

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

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

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

30) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/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.

31) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunExecution.post-commit-persistence.service.test.ts` **— modifié**

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

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

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

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

- Conserver intégration Batch et déplacer tests lifecycle purs.

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

- Instancier/injecter le lifecycle partagé dans le child executor.
- Migrer `items` et consulter `hasActiveOrReservedSourceForLocator` en complément de la projection Activité pour le blocage précoce de Retry.
- Passer au Retry l'adaptateur liveSession enrichi des opérations de claim destructif et de l'éviction exigeant le handle exact.

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

- Injecter lifecycle dans le child executor et runRefs sourceId.

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

- Adapter construction au lifecycle partagé.

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

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

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

- Documenter sourceId=turnId, lifecycle partagé dans le child executor, Activity observer/open, elapsed viewer historique, zéro viewer=hard stop, claim destructif du Retry et réveil générique du cleanup différé après inactivité observer.
- Documenter que le dispose sous claim ne nettoie rien en parallèle du Retry disque et que la libération du claim réveille le cleanup différé.

Endpoints et message bus

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

- Exiger sourceId et transporter sourceId/replayMode.

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

- Couvrir champs, branches et mismatch.

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

- Exiger sourceId et relayer runRef complet.

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

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

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

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

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

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

45) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/liveSession/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.
- La route générée consommée par le frontend est exactement `/api/ai/orch/liveSession/abortSource`.

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

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

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

- Ajouter sourceId, préserver ordre SPECIF/catalogue, déléguer au service source.
- Si et seulement si le tout premier chunk est `meta.error`, envoyer le header `500_SERVER_ERROR` avec `errorCode` et `errorMessage` du protocole privé V1 ; ne jamais bufferiser une autre représentation.
- Après émission d'un header `200_OK`, transférer tout `meta.error` ultérieur comme body chunk normal et ne jamais tenter de modifier le header.
- Ne pas finaliser le registre ; terminer après consommation/canonisation.

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

- Tester délégation, ordre, premier chunk, statut/code/header encodé exact, absence double forwarding, persistance préalable des erreurs synthétiques, absence de synthèse après vrai terminal et absence de finalisation prématurée.
- Tester explicitement `fallback_persisted` ou token puis `meta.error` : header 200, erreur dans le body, aucune utilisation du protocole privé de header.

49) `AICode/src/extension/src/messageBus/handlers/ia/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.
- Utiliser l'allocation synchrone du runtime Activité observer et traiter la publication comme auxiliaire, afin qu'un dispose/reconnect puisse toujours arrêter le bon timer par runId.

50) `AICode/src/extension/src/messageBus/handlers/ia/__test__/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.

- Couvrir l’échec de publication Activité observer sans perte du runId ni fuite de timer.

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

- Recevoir sourceId et construire runRef exact.
- Vérifier synchroniquement que le panneau Prompt exact est encore présent avant la première mutation ; une requête arrivée après dispose est rejetée sans réservation.
- Réserver synchroniquement avant tout await.
- Allouer synchroniquement `activityRunId` et timer, puis attendre/capturer séparément l’éventuelle erreur de publication Activity ; cette erreur de présentation ne doit pas empêcher l’appel endpoint.
- Armer le son exact avec `soundAttemptId` sans perdre l’identité si une autre étape auxiliaire échoue.
- Une réservation exacte dupliquée échoue sans finaliser ni annuler la réservation appartenant au premier handler.
- 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 avant normalisation : 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 mémorise l’erreur transport normalisée sans dépendre d’un rejet du client généré.
- Le décodeur `onError` ne concerne que le header privé. Un `meta.error` tardif reçu dans `onChunk` reste un chunk de données ordinaire et ne doit pas être encodé/décodé une seconde fois.
- Rendre `onError` idempotent afin qu’un éventuel double appel du client généré ne puisse ni doubler le terminal décodé ni doubler l’erreur transport.
- Chaque tâche FIFO tente séparément Activity, son et emit, capture les erreurs dans l’ordre, puis laisse la chaîne continuer afin que les étapes restantes et tous les chunks suivants, notamment le terminal, soient encore tentés ; agréger les erreurs après drainage.
- Après retour Slimbk, attendre Queue FIFO complète. Une erreur source encodée traitée n’est pas un `transportError`; toute autre erreur l’est.
- Finaliser atomiquement le registre après FIFO, y compris lorsque l’échec est antérieur à l’appel endpoint, uniquement si ce handler possède réellement la réservation ; retirer exactement les records sans vrai terminal et obtenir le résultat strict.
- Appeler `onAiStreamStop` avec le `soundAttemptId`, puis déléguer au coordinateur `onSourceSettled({ runRef, activityRunId?, finalizationResult })` ; ne pas publier directement une seconde transition Activité terminale.
- Exécuter tous les cleanups, agréger les erreurs dans l’ordre et préserver la finalisation/coordination même si une étape auxiliaire a échoué.

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

- Vérifier runRef/sourceId/soundAttemptId, allocation synchrone du activityRunId, publication Activity auxiliaire, FIFO avant finalisation/son/coordination, 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 meta.error après un premier chunk normal : passage exclusif par `onChunk`, aucune tentative de décodage header, complétion normale.
- Ajouter terminal backend produit mais callback encore en transit et run sans terminal retiré après drainage.
- Vérifier qu’une erreur auxiliaire sur une étape d’un chunk n’empêche ni les autres étapes du même chunk ni le terminal suivant d’être envoyés au son/Activity/frontend avant l’agrégation.
- Vérifier l’idempotence si le callback `onError` est invoqué plus d’une fois pour la même erreur.
- Vérifier que la transition Activité terminale est demandée une seule fois via le coordinateur, même si la publication Activity initiale a échoué.
- Vérifier qu’un start reçu après suppression du panneau est rejeté sans réservation et qu’un conflit de réservation ne finalise jamais la source du premier handler.

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

- Vérifier réservation, allocation Activity synchrone, publication start, FIFO, finalisation, coordination, 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.

54) `AICode/src/extension/src/extension/panels/prompt/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.
- Ne pas supprimer l'état d'un run/source encore actif lors du dispose du panneau ; le cleanup du son est piloté par le coordinateur uniquement après revalidation de l'absence de panneau, de source et de claim destructif.

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

- Couvrir mêmes turns/sources différentes, deux attempts du même source, stale chunk/stop, dédup terminal, absence de création tardive et conservation de la tentative source pendant une fermeture de panneau.

Activité runtime extension

56) `AICode/src/extension/src/messageBus/handlers/ia/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.
- Remplacer le début async monolithique par une allocation synchrone retournant au minimum `runId` et une promesse de publication : le runtime/timer existe avant le premier await et reste terminable même si le broadcast échoue.
- Exposer un snapshot read-only des runtimes courants pour la réhydratation Activité, sans donner au SidebarStore la propriété des timers.
- Exposer une transition terminale source appelée uniquement par le coordinateur avec `runRef`, `runId` optionnel, panelState relu et résultat de finalisation ; elle arrête toujours le timer exact mais ne mute pas la projection pour `stale` ou pour un runtime remplacé.

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

- Couvrir sources successives, stale, observer fermé, reconnect observer même source, panel source fermé/rouvert et transition terminale source coordonnée unique.
- Couvrir allocation runId/timer synchrone, publication Sidebar rejetée et cleanup exact du runtime malgré cet échec.

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

- Usage exact et transition Idle/suppression, y compris retrait sans terminal : panneau ouvert → Idle ; panneau fermé → suppression.
- Vérifier que l'échec de publication initiale ne bloque ni usage ultérieur ni arrêt du timer exact.

Coordination panneau Prompt

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

- Parser locator et descripteur complet, construire query normalPrompt, distinguer classique/explicite valide/invalide, retirer autoRun lors injection.
- Importer la liste et la constante `easyAgentsBatchLineNumber` depuis le contrat shared ; aucune seconde liste littérale locale.
- Classer comme explicite toute query contenant au moins un paramètre privé connu, `easyAgentsBatchLineNumber`, ou une clé inconnue préfixée `liveSession`, même sans `openMode`.
- Lire toutes les occurrences dans le tableau de query original ; rejeter les doublons/incohérences de `sessionId`, `storageScopeRelPath` et des paramètres live critiques au lieu de choisir implicitement une occurrence.
- Lors de l'injection normalPrompt, générer une lease viewer fraîche, inclure sourceId/adaptateur/policy exacts et retirer `autoRunType` et tous les paramètres `autoRun*` afin qu'une réouverture ne puisse jamais relancer la source.

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

- Couvrir identité, noms exacts des paramètres, adapter, lease fraîche, ordre, policies, partiels, clés `liveSession*` inconnues, doublons locator/live, invalide, rejet avant low-level open et suppression complète des paramètres autoRun.

61) `AICode/src/extension/src/extension/panels/prompt/liveSession/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.
- Rejeter une query live explicite invalide avant l'opener ; préserver une query live explicite valide ; injecter normalPrompt seulement sans descripteur et avec source Prompt exacte active/réservée.

- S’abonner via des ports injectés aux événements exacts où un run observer devient inactif et où un claim destructif est libéré ; ces événements ne décident rien, ils réveillent seulement le locator concerné.
- Dispatch release EasyAgents/normalPrompt, detach source sans abort, ignorer settlement stale.
- Être l’unique propriétaire de la transition Activité terminale source : relire le panneau dans la queue, appeler le runtime Activité avec le `activityRunId` exact lorsqu’il existe, puis seulement décider cleanup.
- 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.
- Pour un cleanup différé EasyAgents, conserver le mapping détaché jusqu’au signal observer inactif ; à chaque wake-up, revalider un éventuel panneau rouvert ou une source plus récente avant tout effet.
- Si `isLocatorDestructionClaimed(locator)` est vrai lors du dispose, conserver le mapping détaché et différer tous les cleanups thread/mémoire/attachments/son/trivial ; reprendre uniquement sur le wake-up de release du claim, après revalidation complète.

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

- Couvrir priorités, rejet des query invalides avant opener, releases, races open/cleanup, pré-baseline, observer, source suivante, drainage post-terminal, retrait sans terminal, Idle panneau ouvert, transition Activité unique et protection mémoire/son.
- Couvrir panneau EasyAgents fermé avec autre viewer Batch encore présent, cleanup différé, signal handoff/éviction observer, réouverture avant signal et nouvelle source avant signal.
- Couvrir dispose sous claim delete/Retry : aucun cleanup avant release, wake-up après release, no-op si panneau rouvert ou nouvelle source, et cleanup unique si le locator reste fermé/inactif.

63) `AICode/src/extension/src/extension/panels/prompt/liveSession/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.

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

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

65) `AICode/src/extension/src/extension/panels/prompt/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.
- Conserver un mapping détaché lorsqu’un cleanup est différé ; le supprimer seulement après cleanup terminal réussi ou réenregistrement explicite du panneau.
- Ne jamais exécuter un cleanup qui touche mémoire/attachments pendant un claim destructif ; laisser le coordinateur reprendre après le wake-up de release.
- Revalider l’absence de source/panneau/claim après chaque await du cleanup historique.

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

- Couvrir scopes, detach, idle, terminal, mapping absent, nouvelle source pendant cleanup et sérialisation attachments.
- Couvrir cleanup observer différé puis réveillé après handoff/éviction et no-op lorsque le panneau a rouvert.
- Couvrir mapping détaché sous claim : aucun cleanup avant release, puis cleanup revalidé et idempotent.

67) `AICode/src/extension/src/extension/panels/core/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.
- Injecter au coordinateur les souscriptions observer inactive et claim released du LiveSessionService, le runtime Activité et l’autorité de claim.
- S’assurer que le dispose sous claim ne lance aucun cleanup concurrent avant le wake-up de release.

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

- Wiring sans cycle, identité/adaptateur, reveal, dispose unique, unregister avant coordinateur, ports release, abonnements observer/claim, notification sourceSettled/Activity, dispose sous claim et pages non-Prompt/fixtures `items`.

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

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

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

- Mocker façade Prompt et préserver URL.

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

- Await ouverture coordonnée.

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

- Vérifier async et propagation erreur.

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

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

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

- Couvrir root/nested, façade coordonnée et guards.

SidebarStore et Activité

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

- Transitions atomiques `panelOpened`, `sourceStarted`, `sourceSettled`; lecture panelState et source active sans await intermédiaire.
- `sourceStarted` reçoit l'identité/runtime déjà allouée synchroniquement et retourne une promesse de publication séparée ; l'échec de broadcast ne doit pas perdre le runId.
- `sourceSettled` reçoit le résultat strict de finalisation et le runRef exact, et n'est appelé pour une source normale que par le coordinateur.
- Mapping obligatoire : résultat non-stale + panneau ouvert → Idle ; résultat non-stale + panneau fermé → suppression ; stale ou sourceId différent de l'item courant → no-op.

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

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

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

- Initialiser items et appeler `rehydrateQueryActivities`.

78) `AICode/src/extension/src/extension/panels/sidebar/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.

79) `AICode/src/extension/src/extension/panels/sidebar/__test__/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.

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

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

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

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

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

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

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

- Vérifier payloads, accepted et délégation.

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

- Lire items ; bloquer si projection Running ou source/observer active/réservée selon l'autorité du registre.

- Acquérir synchroniquement un `LiveSessionLocatorDestructionClaim` avant de fermer le panneau ; l'acquisition remplace la revalidation racy « check puis await », refuse tout record/réservation actif et bloque tout nouveau start/Open pendant la suppression.
- Fermer le panneau et retirer la projection tout en conservant le claim ; le dispose coordonné doit différer ses cleanups jusqu'à la libération du claim.
- Supprimer les données durables sans cleanup panneau concurrent.
- Après suppression durable réussie, appeler l'éviction avec le handle exact du claim, retirer records, retraites replayables et leases, puis libérer le claim ; les tombstones exactes restent conservées.
- Si une étape précédant le succès disque échoue, libérer le claim sans évincer préventivement les états retenus ; le wake-up de release laisse alors le coordinateur reprendre le cleanup normal du panneau fermé.
- Préserver la propagation d'erreur disque et ne pas perdre l'autorité liveSession si la suppression durable échoue.
  85) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/delete.endpoint.liveSession.test.ts` **— créé**
- Couvrir open/closed/réservation/store stale/absent, terminal bufferisé avant drainage, claim indisponible, source tentant de démarrer pendant le claim, panneau Idle après retrait, run retiré sans terminal et éviction des états retenus uniquement après suppression durable réussie.
- Vérifier release du claim sur succès et sur erreur, passage du handle exact à l'éviction et conservation des tombstones après éviction.
- Vérifier qu'aucun cleanup Prompt ne court pendant le claim et qu'il reprend seulement après le wake-up de release.
  86) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/list.endpoint.windowsLock.eperm.test.ts` **— modifié**
- Adapter fixtures items.
  87) `AICode/src/extension/src/slimbk-local/endpoints/ai/orch/chat/__test__/diskAliases.lastWins.endpoints.test.ts` **— modifié**
- Adapter items et préserver delete disque.
  88) `AICode/src/extension/src/aiTools/orch/context/persistence/__test__/ChatPersistenceWriter.windowsFallback.test.ts` **— modifié**
- Adapter fixture items.
  89) `AICode/src/extension/src/aiTools/orch/context/memory/core/mem/__test__/OrchMemory.naive_title.user_prompt_specif_multiline.e2e.repro.test.ts` **— modifié**
- Adapter fixture items.
  90) `AICode/src/frontend/src/app/sidebar/sidebar-activity/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.
- Donner au sous-composant ou au bouton Stop une clé React dérivée de `sourceId`, distincte de la clé locator de l'item, pour réinitialiser l'état interne `disableAfterClick` à chaque source exacte.
  91) `AICode/src/frontend/src/app/sidebar/sidebar-activity/__test__/SidebarActivityQuery.test.tsx` **— modifié**
- Couvrir locators, observer/source, Stop, disparition, payload, désactivation et testIds stables sans labels fragiles.
- Couvrir A Stop désactivé puis remplacement par source B sur le même locator : le bouton B doit être réactivé grâce à sa clé `sourceId`.

  Frontend `liveSession` et Prompt
  92) `AICode/src/frontend/src/app/prompt/liveSession/adapters/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 sans second Open, 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.
- Déclarer explicitement que les retries normalPrompt utilisent `composerStatePolicy='preserveLocal'`, alors que l'installation initiale de baseline utilise l'hydratation complète.
  93) `AICode/src/frontend/src/app/prompt/liveSession/adapters/__test__/NormalPromptLiveSession.adapter.test.ts` **— créé**
- Couvrir parsing, paramètres partiels, policies, view models composer/bandeau, handoff direct, policy d'application sélective et absence de Stop banner.
  94) `AICode/src/frontend/src/app/prompt/liveSession/adapters/PromptLiveSessionBootstrap.adapter.ts` **— créé**
- Dispatcher pur strict EasyAgents/normalPrompt, aucun fallback.

- Recevoir le snapshot brut figé par `Prompt.tsx`; ne jamais lire lui-même la globale router ni utiliser `useQueryString(...)`.
- Importer la liste privée et `EASY_AGENTS_BATCH_LINE_NUMBER_PARAM` depuis le contrat shared.
- Analyser les occurrences avec `URLSearchParams.getAll(...)`, retourner la classification classic/explicite valide/explicite invalide et le locator strict correspondant lorsqu'il est univoque.
- Toute présence partielle de paramètre liveSession, de `easyAgentsBatchLineNumber`, de clé `liveSession*` inconnue ou tout doublon critique doit produire `invalid`, même si `openMode` manque.
- Une branche `invalid` peut légitimement ne pas fournir de locator lorsqu'aucune identité unique ne peut être dérivée ; le caller doit alors utiliser la frontière bootstrap externe et ne jamais fabriquer de locator.
- Exposer un booléen de classification explicite utilisé par Prompt pour désactiver hydratation classique et auto-run.

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

- Couvrir dispatch, partiels, doublons réels dans la query brute, locator dupliqué identique/différent, branche invalide sans locator, clés `liveSession*` inconnues, adaptateurs inconnus et résultat locator/classification.
- Vérifier que le parser est pur et ne relit jamais la globale router.

96) `AICode/src/frontend/src/app/prompt/liveSession/core/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, stratégie de handoff, policy d'application du snapshot de retry et bannière optionnelle.
- Définir un modèle de bannière dédié, indépendant de la surface composer, afin qu'une bannière Retry sans Stop puisse coexister avec un compositeur normal visible.
- Représenter explicitement `directAfterReplay` pour normalPrompt et `terminalSnapshot` pour EasyAgents/bootstrap terminal.
- Documenter que `replayOpenAttemptId` garde les requêtes Open alors que `streamInstanceId` n'existe que pour un stream effectivement accepté.

97) `AICode/src/frontend/src/app/prompt/liveSession/components/PromptLiveSessionBanner.tsx` — modifié

- Consommer le modèle de bannière dédié au lieu d'un type limité au viewer read-only.
- Conserver le rendu générique status/details/Stop/Retry, sans synthétiser de wording ni de second Stop.
- Préserver les callbacks `useCallback`, les testIds et les variantes existantes.

98) `AICode/src/frontend/src/app/prompt/liveSession/components/__test__/PromptLiveSessionBanner.test.tsx` — modifié

- Adapter les fixtures au modèle de bannière dédié.
- Conserver les tests actions visibles/cachées/désactivées et ajouter le cas bannière Retry normalPrompt sans Stop.

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

- Transporter sourceId/replayMode, retry not_ready annulable, replay selon mode.
- Maintenir un `replayOpenAttemptId` monotone pour chaque bootstrap/retry ; vérifier cet identifiant après chaque `await`, dans les timers et avant toute mutation, et invalider l'attempt à l'unmount ou au démarrage d'un retry suivant.
- Parser d'abord toute réponse avec `ZLiveSessionOpenResultEnvelope` pour valider la branche et l'identité exacte.
- Pour une application complète, reparcourir `ZLiveSessionOpenResult` puis le validateur frontend historique ; pour un retry normalPrompt `preserveLocal`, transmettre le `session` opaque au helper sélectif sans parse complet préalable.
- Handoff direct normal uniquement vrai terminal data + Queue A idle + Queue B idle + outcome completed + finalisation Prompt du même streamInstanceId.
- Lorsque le handoff direct normal est validé, passer à `normal` sans appeler de nouveau `/api/ai/orch/liveSession/open`, sans `resetPromptRunState` global et sans réappliquer l'enveloppe au compositeur.
- `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 la baseline ; une réponse `live` acceptée crée un nouveau `streamInstanceId`, tandis qu'une réponse directement `terminal` reste une tentative Open sans streamInstanceId.

- Sur retry, appeler `applyHydratedPromptSessionState` avec sa branche discriminée `composerStatePolicy='preserveLocal'` : remplacer messages et stores d'hydratation, mais ne pas fournir ni appeler les setters du brouillon, des attachments et de `nextTurnId`, y compris sur une réponse `terminal`.
- Traiter explicitement un refus synchrone de `startLiveObserver` sans installer de placeholder et sans laisser le mode Streaming ; rester/revenir en replayFailed avec un message déterministe appartenant à la policy normalPrompt.
- Gérer explicitement `transportStopAvailable` : conserver sur `transportError` sans terminal, supprimer sur vrai terminal, complétion replay normale, Open terminal/not_found, reset ou premier Stop consommé.

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

- Ajouter identité/replayMode, comportements not_ready/not_found par adaptateur et préserver EasyAgents.
- Vérifier la policy d'application : bootstrap initial hydrate le compositeur ; retry normalPrompt utilise `preserveLocal` et ne touche pas aux champs de draft.
- Vérifier qu'un draft volontairement malformé est rejeté par l'hydratation complète mais n'empêche pas un retry `preserveLocal` dont les messages consommés sont valides.
- Vérifier qu'une réponse/timer d'un ancien `replayOpenAttemptId` est ignoré après retry ou unmount.
- Ne pas tester ici la lecture de la globale/query brute : cette responsabilité appartient au dispatcher pur et à l'intégration `Prompt.tsx`.

101) **`AICode/src/frontend/src/app/prompt/liveSession/core/__test__/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.
- Vérifier explicitement que le handoff direct normal n'émet aucun second appel Open et ne rappelle pas l'hydrateur terminal.
- Vérifier que le handoff n'est pas évalué avant Queue B idle.
- Vérifier qu'un vrai terminal suivi d'un transportError ne reçoit aucun second meta.error local et ne handoff pas directement.

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

- Couvrir not_ready, not_found, surface interactive, draft/attachments/nextTurnId préservés, Stop, replay failure/retry, direct handoff sans Open, terminal, éviction sans terminal et close/reopen.
- Couvrir le retry dont Open répond directement terminal : seuls messages/stores sont remplacés, le compositeur local reste intact, aucun `startLiveObserver` n'est appelé et aucun `streamInstanceId` fictif n'est créé.
- Couvrir un retry dont l'enveloppe contient des champs draft hors contrat mais des messages valides : le schéma d'enveloppe et `preserveLocal` doivent réussir sans lire ces champs.
- Couvrir replayFailed avec transport terminé mais source encore abortable, puis consommation locale du premier Stop exact.
- Couvrir réponses Open stale gardées par `replayOpenAttemptId`.

103) **`AICode/src/frontend/src/app/prompt/liveSession/core/__test__/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.
- Vérifier que SYSTEM_ERROR/INTERRUPTED synthétiques ne deviennent handoffables qu'après persistance exacte et suffixe hydraté identique.
- Vérifier qu'un terminal data suivi d'un transportError conserve le terminal unique et ne crée aucun suffixe/error supplémentaire.

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

- Importer `EASY_AGENTS_BATCH_LINE_NUMBER_PARAM` depuis le contrat shared ; supprimer la constante locale.
- Ajouter sourceId=turnId/adaptateur, préserver UX, elapsed viewer et bandeau read-only historique.
- Déclarer l'application complète des snapshots pour bootstrap/handoff/retry EasyAgents, qui n'a pas de compositeur local interactif à préserver.

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

- Adapter query, sourceId, dispatcher, constante shared et policy d'application complète.

106) **`AICode/src/frontend/src/messageBus/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.
- Définir et exporter l’union de retour synchrone du start : refus sans identité, ou acceptation avec runRef exact et streamInstanceId.
- Documenter que `replayOpenAttemptId` n’appartient pas à ce contrat car il garde l’Open RPC et non le stream.

107) `AICode/src/frontend/src/messageBus/useAiStream.hook.ts` — modifié

- Générer identités après acceptation synchrone, avec ref single-flight immédiatement muté pour bloquer tous les doubles starts du même tick, même si le turnId diffère.
- Faire retourner l’union d’acceptation ; la branche refusée ne doit muter aucun buffer, lock, queue, activeStream ni identité.
- Utiliser streamInstanceId comme clé listener à la place de streamKey.
- Exposer runRef, streamInstanceId, outcome, `transportStopAvailable` et l’état local de demande Stop.
- Stop exact selon source/observer ; `/api/ai/orch/liveSession/abortSource` pour une source Prompt normale et l’endpoint lease exact pour un observer.
- Consommer localement `transportStopAvailable` avant le premier await du Stop et retourner `{ accepted }`, afin qu’un double clic ou Escape tardif ne puisse pas lancer un second abort.
- 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 : il utilise la fabrique canonique SYSTEM_ERROR seulement si aucun vrai terminal data n’a déjà été consommé pour la même instance ; sinon il conserve le terminal existant et publie uniquement `transportError`.
- 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.
- Avant de publier `completed`/`transportError` et de libérer le ref single-flight,
  attendre `streamQueue.whenIdle()`, `dispatchGateQueue.whenIdle()` puis la barrière de commit.
- Invalider l’identité courante avant tout reset de queue/lock.
- Conserver `transportStopAvailable` sur transportError sans terminal tant qu’aucune preuve terminale/not-found ou action Stop ne l’a consommée ; la supprimer sur terminal data, complétion normale/reset.
- Ne modifier aucun algorithme Queue/lock/sanitizer/raw-live.

108) `AICode/src/frontend/src/messageBus/__test__/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 sans terminal donne transportError.
- Vérifier qu’un onError après un terminal data conserve ce terminal, n’enfile aucun SYSTEM_ERROR supplémentaire et donne transportError.
- Couvrir un vieux runMerged/onComplete qui reprend après await et ne reset jamais la nouvelle tentative.
- Vérifier que l’outcome et la nouvelle acceptation restent bloqués tant que Queue B n’est pas idle.
- Vérifier les transitions de `transportStopAvailable` sur completed, terminal data, transportError sans terminal, Stop et reset.

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

- Vérifier unicité, stabilité, double start même tick, branche d’acceptation/refus sans mutation, abort exact, callbacks stale et observer retry.
- Vérifier le résultat Stop `{ accepted }` et la consommation synchrone de `transportStopAvailable`.
- Vérifier que sourceId n’est généré que pour une source Prompt acceptée et que l’observer conserve le sourceId fourni.

110) `AICode/src/frontend/src/app/prompt/Prompt/usePromptRunController.hook.ts` — modifié

- Observer reçoit runRef+lease+replayMode+Stop policy ; base thinking + streamInstanceId ; exposer outcome/tentative/runRef/transportStopAvailable.
- Conserver une initialisation thinking en attente, la lier synchroniquement au streamInstanceId accepté avant les effets transport.
- `startLiveObserver` appelle d’abord `start`, retourne son union stricte, puis ne prépare les placeholders qu’après une acceptation réussie ; un refus ne doit pas dupliquer les bulles.
- Exposer `finalizedStreamInstanceId`, l’opération explicite de retrait de `transportStopAvailable` utilisée par Open terminal/not_found, et reset complet.
- Passer au cancel core `isStreaming` réel et `transportStopAvailable` séparément, afin qu’un replayFailed puisse encore appeler le Stop backend exact sans être traité comme un simple cancel UI.

111) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunController.liveSessionObserver.test.tsx` **— modifié**

- Couvrir identité, modes, base thinking, tentative, outcome, acceptation refusée sans placeholder, Stop disponible après transport replay échoué, retrait explicite après Open terminal/not_found et accusé de finalisation exact.

112) `AICode/src/frontend/src/app/prompt/Prompt/usePromptCancelController.hook.ts` **— modifié**

- Séparer dans `usePromptCancelCore` l'état réel `isStreaming` de `transportStopAvailable`.
- UI-only cancel uniquement lorsque les deux sont faux.
- Lorsqu'un replay transport est terminé mais que la source exacte reste abortable, appeler `stop()` une seule fois, consommer la disponibilité exacte et ne jamais effacer la source par une simple mutation UI.
- Conserver le handshake jusqu'à la fin du transport pour un stream réellement actif ; pour replayFailed déjà hors transport, le handshake peut se fermer après la réponse Stop tandis que `transportStopAvailable=false` maintient le bouton désactivé.

113) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptCancelController.stopCore.test.tsx` **— modifié**

- Préserver les tests cancel UI et stream actif.
- Ajouter le cas `isStreaming=false + transportStopAvailable=true`, vérifier appel Stop exact unique, double clic ignoré, puis retour au mode non-abortable sans mutation de source.

Continuation et thinking

114) `AICode/src/frontend/src/app/prompt/Prompt/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 pure permettant au controller de lier la base thinking à la prochaine tentative acceptée sans générer l'identité en avance ni muter le Prompt avant l'acceptation.

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

- Couvrir assistant, bases, thinking, séparateur, manquant, sources successives, aucune duplication et absence de mutation pendant la préparation pure.

116) `AICode/src/frontend/src/app/prompt/Prompt/preparePromptStreamingTurn.util.ts` **— modifié**

- Reset explicite base thinking/séparateur/initialisation en attente pour replace.
- Séparer préparation pure et commit du scaffold afin que le caller appelle `start()` avant toute mutation visible et ne committe qu'après `accepted: true`.

117) `AICode/src/frontend/src/app/prompt/Prompt/__test__/preparePromptStreamingTurn.util.test.ts` **— modifié**

- Vérifier resets et absence de mutation lorsque la préparation n'est pas commitée après un start refusé.

118) `AICode/src/frontend/src/app/prompt/Prompt/usePromptThinkingLifecycle.hook.ts` **— modifié**

- Recevoir base thinking et streamInstanceId ; initialiser la tentative avant le traitement des premiers deltas 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.

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

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

120) `AICode/src/frontend/src/app/prompt/Prompt/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.

121) `AICode/src/frontend/src/app/prompt/Prompt/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.

122) `AICode/src/frontend/src/app/prompt/Prompt/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.
- Pour Send, SPECIF, Retry, Edit, Compact, Reply et Continue, construire d'abord un plan pur, appeler `start()`, propager synchroniquement son union au caller, puis seulement committer scaffold, `uiPending`, placeholders, refs et scroll après acceptation.

- Retirer les marqueurs `async` artificiels des handlers qui n'attendent rien afin que le résultat d'acceptation remonte réellement dans la même pile.
- Sur refus, ne conserver aucune mutation locale ; remettre `uiPending=false` si un gate l'avait armé optimistement.
- Les handlers publics de démarrage retournent le résultat strict afin que `Prompt.tsx` puisse armer ou non ses barrières locales dans la même pile.

123) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.compactContextNow.test.tsx` **— modifié**

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

124) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.retryQuery.attachmentOnly.test.tsx` **— modifié**

- Adapter le harness au nouveau contrat d'acceptation et au nouveau ref thinking ; préserver les invariants attachment-only/retry.

125) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.retryQuery.compactContextNow.test.tsx` **— modifié**

- Adapter le harness au nouveau contrat d'acceptation et au nouveau ref thinking ; préserver le mapping retry compact-context.

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

- Vérifier helper, même turn, nouvelle source/tentative, thinking unique, assistant exact.
- Vérifier qu'un start refusé dans le même tick ne modifie ni bulles, ni base thinking, ni `uiPending` final.

Intégration Prompt et documentation

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

- Transformer le fichier en frontière bootstrap externe + runtime Prompt interne résolu, sans déplacer le comportement métier : la frontière externe appelle `usePromptRouterQueryStringRawSnapshot()` une seule fois, transmet ce snapshot au dispatcher pur et ne monte le runtime interne qu'avec un locator strict.
- Une classification live explicite invalide sans locator univoque doit afficher/fermer l'erreur depuis la frontière externe ; ne pas instancier `usePromptRunController`, ne pas déclencher hydration/autoRun et ne pas fabriquer de locator sentinelle.
- Ne pas utiliser les hooks scalaires `useQueryString` pour décider si un bootstrap live est explicite, valide, dupliqué ou pour choisir son locator.
- 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 et qu'un Stop backend encore disponible reste appelable.
- Cleanup ownership différé jusqu'au handoff direct ; hydration classique off pour toute query live explicite, valide ou invalide.
- Court-circuiter explicitement l'effet auto-run pour toute query live explicite, même si des paramètres autoRun sont présents.
- Lors d'un Send/Edit accepté, armer `pendingClearAfterFallbackAckRef`, `submitTxnPending` et la restauration de focus seulement après le résultat `accepted: true` retourné synchroniquement ; sur refus, annuler le seul éventuel flip optimiste `uiPending` dans la même pile.

128) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.liveSessionBootstrap.test.tsx` **— modifié**

- Couvrir dispatch brut, doublons, identité, surface, draft, query figée, hydration off, replay failure, bandeau Retry, verrou Send, éviction sans terminal et ownership cleanup.
- Vérifier qu'une query live explicite avec paramètres autoRun ne déclenche jamais de Send/SPECIF automatique.
- Vérifier qu'une query invalide avec locator ambigu ne monte pas le runtime Prompt et ne déclenche aucun hook/backend métier.
- Vérifier refus start même tick sans barrières clear/submit/focus persistantes.
- Remplacer les mocks scalaires live par le snapshot brut canonique et faire exporter le hook de snapshot par le mock du module locator.

129) `AICode/src/frontend/src/app/prompt/PromptInput/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.
- Consommer le modèle de bannière dédié séparément de `showComposer` et conserver le verrou Send distinct de l'édition du textarea/attachments.

130) `AICode/src/frontend/src/app/prompt/PromptInput/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.

131) `AICode/src/frontend/src/app/prompt/PromptInput/__test__/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.

132) `AICode/src/frontend/src/app/easy-agents/easy-agents-batch/core/__test__/EasyAgentsBatch.liveWatch.test.tsx` **— modifié**

- Adapter query exacte source/adaptateur et constante ligne shared.

133) `AICode/src/frontend/src/app/easy-agents/easy-agents-item/core/batch/__test__/EasyAgentBatchSurface.test.tsx` **— modifié**

- Adapter query exacte sans changement UX.

134) `AICode/src/frontend/src/app/prompt/hydration/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, et qu'un meta.error postérieur au header 200 reste un body chunk normal.
- Documenter que tout terminal synthétique wrapper est persisté avec suffixe exact avant append, n'est créé qu'en l'absence de terminal et qu'un échec de cette persistance interdit le handoff.
- Documenter qu'un terminal data suivi d'un transportError reste l'unique terminal visible et qu'aucun SYSTEM_ERROR frontend supplémentaire n'est ajouté.
- Documenter que le handoff normalPrompt réussi est une transition directe sur l'état replayé et ne relit pas un second snapshot terminal.
- Documenter la différence explicite entre hydratation complète et application sélective préservant le compositeur, y compris le parse d'enveloppe Open qui laisse les champs draft opaques.
- Documenter la séparation `replayOpenAttemptId`/`streamInstanceId`, notamment la branche Open terminal sans stream.

135) `AICode/src/frontend/src/app/prompt/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.

Contrats transverses révélés par la réinspection finale

136) `AICode/src/frontend/src/app/prompt/Prompt/applyHydratedPromptSessionState.util.ts` **— modifié**

- Transformer les arguments en union discriminée stricte par `composerStatePolicy`.
- Branche `hydrateFromSnapshot` : conserver exactement le comportement actuel du compositeur, avec setters obligatoires et traitement des draft attachments.
- Branche `preserveLocal` : ne pas accepter les setters de `promptValue`, attachments ou `nextTurnId`; ne pas lire/valider/normaliser/enregistrer `promptDraft`, `promptDraftAttachments` ni `meta.nextTurnId` ; appliquer uniquement les messages, stores d'hydratation, thread, reset et scroll.
- Adapter la validation du snapshot à la policy afin qu'un champ draft malformé hors périmètre ne fasse pas échouer `preserveLocal`, tout en conservant la validation stricte des métadonnées/messages réellement consommés.
- Construire dans chaque branche un objet assaini ne contenant que des champs validés ; la branche `preserveLocal` retire explicitement les champs draft/nextTurnId bruts au lieu de caster l'objet source en `FrontChatSession` complet.
- Faire retourner `void` au helper, aucun caller actuel ne consommant son ancien retour ; ne jamais exposer un type complet non prouvé.
- Continuer à enregistrer les doc refs présents dans les messages hydratés dans les deux branches.
- Documenter que ce contrat évite les faux setters no-op et les erreurs provenant d'une enveloppe de draft hors périmètre.

137) `AICode/src/frontend/src/app/prompt/Prompt/__test__/applyHydratedPromptSessionState.util.test.ts` — modifié

- Adapter les appels existants à `hydrateFromSnapshot`.
- Ajouter `preserveLocal` et vérifier que messages/stores/thread/reset/scroll sont appliqués, alors que les setters de compositeur ne font pas partie de la branche.
- Vérifier que la branche de préservation n'appelle pas le normaliseur d'attachments de draft, ne lit/valide pas `meta.nextTurnId` et n'échoue pas sur des champs de draft volontairement hors contrat, tout en continuant à traiter les attachments des messages.
- Vérifier que le helper retourne `void` et qu'aucun objet non validé n'est exposé au caller.

138) `AICode/src/frontend/src/app/prompt/Prompt/usePromptSessionHydration.hook.ts` — modifié

- Appeler le helper avec `composerStatePolicy='hydrateFromSnapshot'` pour l'ouverture classique.
- Généraliser les commentaires : le hook classique est désactivé pour toute query live explicite, pas uniquement pour EasyAgents.

139) `AICode/src/frontend/src/app/prompt/hydration/frontSessionHydration.type.ts` — modifié

- Mettre à jour le contrat documentaire de `allowOrphanToolResults=true` : il couvre toute baseline live-safe qui retire les assistants actifs du tour, EasyAgents comme normalPrompt.
- Ne modifier aucun algorithme de pairing/hydratation.

140) `AICode/src/frontend/src/app/prompt/lib/usePromptSessionLocator.hook.ts` — modifié

- Ajouter `usePromptRouterQueryStringRawSnapshot()`, qui appelle le lecteur bas niveau une seule fois via `useMemo(..., [])` et retourne le snapshot brut figé.
- Conserver les parseurs locator existants pour les consommateurs non-live et les liens Markdown.
- Corriger les commentaires afin que le module n'affirme plus que `usePromptSessionLocator()` est l'unique voie autorisée pour lire la query brute ; le nouveau hook est la voie unique de bootstrap Prompt.

141) `AICode/src/frontend/src/app/prompt/lib/__test__/usePromptSessionLocator.hook.test.tsx` — modifié

- Conserver les tests locator existants.
- Ajouter un harness du snapshot brut, modifier la globale après le premier rendu puis rerender, et vérifier que le snapshot reste strictement inchangé.

142) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.autoRunPrompt.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator et fournir une query classique canonique.
- Adapter `handleSend` et `handleSendSpecifAction` à l'union synchrone acceptée.
- Préserver les assertions auto-run classiques et vérifier qu'aucune régression n'est introduite par la classification live séparée.

143) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.editLastPrompt.noThumbTurnIdDrift.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator.
- Faire retourner une acceptation synchrone par `handleSubmitEdit` afin que la barrière locale ne soit armée qu'après acceptation.
- Préserver l'invariant de turnId des thumbnails jusqu'au fallback ack.

144) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.editLastPrompt.restoreAttachments.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator avec une query classique canonique.
- Préserver les assertions de restauration d'attachments ; aucun changement fonctionnel de l'edit n'est attendu dans ce test.

145) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.ensureNextTurnId.stability.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator avec une query classique canonique.
- Aligner les mocks de démarrage sur l'union d'acceptation sans modifier les assertions de stabilité du nextTurnId.

146) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator.
- Faire retourner `accepted: true` par le mock de Send/Edit avant d'attendre le fallback ack.
- Préserver la preuve que les attachments ne sont effacés qu'après `fallback_persisted`.

147) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx` — modifié

- Faire exporter le hook de snapshot brut par le mock locator.
- Faire retourner `accepted: true` par le mock de Send/Edit et vérifier que le masque transactionnel n'est armé qu'après cette acceptation.

- Préserver la preuve de non-perte du texte avant fallback ack.
  148) `AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.promptDraft.hydration.test.tsx` **— modifié**
- Faire exporter le hook de snapshot brut par le mock locator avec une query classique canonique.
- Aligner les mocks de démarrage sur l'union d'acceptation ; préserver les assertions de debounce/hydratation du brouillon.
  149) `AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.startAcceptance.test.tsx` **— créé**
- Construire un harness commun strict pour Send, SPECIF, Retry, Edit, Compact context, Reply et Continue.
- Pour chaque commande, faire retourner `{ accepted: false }` par `start()` et vérifier : aucun message/scaffold ajouté ou retiré, aucune ref cible/base thinking mutée, aucun reset de buffer/notice conservé, `uiPending` revenu à false et résultat refusé propagé synchroniquement.
- Pour chaque commande, vérifier la branche acceptée : un seul commit de scaffold, identités issues du résultat accepté, `uiPending`/scroll conformes et résultat strict propagé au caller.
  150) `AICode/src/frontend/src/messageBus/prompt-stream-dispatch-lock.programming-doc-readme.md` **— modifié**
- Conserver les algorithmes Queue A/Queue B inchangés.
- Documenter que `streamInstanceId` garde les callbacks et `runMerged`, que Queue A et Queue B doivent être toutes deux idle avant outcome/reset/libération single-flight, et qu'un reset stale ne peut jamais vider la tentative suivante.
- Documenter la dernière barrière de commit précédant `completed|transportError`.
- Documenter qu'un transportError après terminal data ne crée pas un second terminal dans la queue.
  151) `AICode/src/extension/src/aiTools/orch/ask/stream/core/fallback-persist-and-title.programming-doc-readme.md` **— modifié**
- Documenter que la barrière frontend de clear/mask/focus n'est armée qu'après acceptation synchrone du start.
- Documenter qu'un start refusé laisse le brouillon, les attachments et le nextTurnId inchangés et annule le seul flip UI optimiste.
- Ne modifier aucune règle backend de fallback persist ou de titre.
  Atomicité destructive du Retry EasyAgents révélée par la réinspection
  152) `AICode/src/extension/src/aiTools/easyAgents/batch/run/EasyAgentsBatchRunRetry.service.ts` **— modifié**
- Conserver tous les guards Git/préparation/viewer historiques.
- Après les validations non destructives et immédiatement avant la fermeture du child Prompt, acquérir synchroniquement un claim destructif exact sur `targetRow.childSessionLocator`.
- Si le claim est indisponible, retourner/lever le blocker child-running existant sans fermer le panneau, sans éviction et sans delete.
- Maintenir le claim pendant fermeture, éviction liveSession, retrait Activité et suppression durable du child.
- Passer le handle opaque exact à l'éviction pré-delete ; aucune éviction publique sans claim n'est autorisée.
- Le dispose du child Prompt doit constater le claim et différer tous ses cleanups ; le signal de release reprendra le cleanup après la séquence destructive.
- Conserver l'ordre historique d'éviction pré-delete du Retry ; le claim garantit qu'aucune nouvelle source/réservation ne peut apparaître entre le dernier check et le delete.
- Libérer le claim dans un `finally`, y compris lorsque l'éviction, le retrait UI, le delete ou la reconstruction cold échoue.
- Ne pas modifier les règles Git, de reconstruction, de viewer ou de redémarrage de la ligne.
  153) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.testHarness.ts` **— modifié**
- Ajouter les mocks typés `claimLocatorDestruction`, `releaseLocatorDestructionClaim`, `hasActiveOrReservedSourceForLocator` et `evictLocator` exigeant le claim ; exposer le wake-up de release si nécessaire au harness coordinateur.
- Fournir par défaut un claim opaque valide et exposer les appels au harness sans casts inutiles.
  154) `AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.guards.service.test.ts` **— modifié**
- Vérifier qu'un claim indisponible bloque avant fermeture/éviction/delete.
- Vérifier que le claim reste détenu pendant la séquence destructive, que l'éviction reçoit exactement ce handle, puis qu'il est libéré sur succès et sur chaque erreur.
- Vérifier qu'une source qui tente de se réserver entre le précheck et le delete est refusée par le claim et qu'aucune source concurrente n'est évincée par erreur.
- Vérifier qu'aucun cleanup Prompt ne court pendant le claim et que la libération réveille ensuite le cleanup différé.
- Préserver les tests de guards Git, préparation et child-running existants.

Récapitulatif exact des chemins de fichiers
Créés

- AICode/src/sharedlib/src/types/liveSessionTypes/LiveSessionReplay.type.ts
- AICode/src/extension/src/aiTools/orch/context/memory/core/msg/__test__/OrchMessagesMemoryAssistantPatch.errorSuffix.test.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
- AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptRunCommands.startAcceptance.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/extension/src/aiTools/orch/context/memory/core/msg/OrchMessagesMemoryAssistantPatch.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/batch/run/EasyAgentsBatchRunRetry.service.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.testHarness.ts
- AICode/src/extension/src/aiTools/easyAgents/batch/run/__test__/EasyAgentsBatchRunRetry.guards.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/components/PromptLiveSessionBanner.tsx
- AICode/src/frontend/src/app/prompt/liveSession/components/__test__/PromptLiveSessionBanner.test.tsx
- 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/usePromptCancelController.hook.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/usePromptCancelController.stopCore.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
- AICode/src/frontend/src/app/prompt/Prompt/applyHydratedPromptSessionState.util.ts
- AICode/src/frontend/src/app/prompt/Prompt/__test__/applyHydratedPromptSessionState.util.test.ts
- AICode/src/frontend/src/app/prompt/Prompt/usePromptSessionHydration.hook.ts
- AICode/src/frontend/src/app/prompt/hydration/frontSessionHydration.type.ts
- AICode/src/frontend/src/app/prompt/lib/usePromptSessionLocator.hook.ts
- AICode/src/frontend/src/app/prompt/lib/__test__/usePromptSessionLocator.hook.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.autoRunPrompt.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.editLastPrompt.noThumbTurnIdDrift.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.editLastPrompt.restoreAttachments.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.ensureNextTurnId.stability.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearAttachments.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.fallbackPersistAck.clearInput.test.tsx
- AICode/src/frontend/src/app/prompt/Prompt/__test__/Prompt.promptDraft.hydration.test.tsx
- AICode/src/frontend/src/messageBus/prompt-stream-dispatch-lock.programming-doc-readme.md
- AICode/src/extension/src/aiTools/orch/ask/stream/core/fallback-persist-and-title.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, séparation replayOpenAttemptId/streamInstanceId, réservations et époque dès la réservation, conflit de réservation exacte, acceptation synchrone avant scaffold Prompt, propagation synchrone accept/refuse jusqu'aux barrières locales de Prompt, couverture transactionnelle de toutes les commandes, callbacks stale, chunks déjà sortis de Queue A, chunks différés Queue B, finalisation React par tentative, terminal data suivi d'un transportError sans double terminal, lifecycle lazy, pipeline non attendue, double callback `onError`, continuation de FIFO après erreur auxiliaire, allocation synchrone du runtime Activité malgré erreur de broadcast, protocole strict statut/code limité au premier chunk source-error, meta.error tardif sous header 200, persistance obligatoire des terminaux synthétiques seulement en l'absence de terminal, finalisation post-forwarding, priorité stale après changement d'époque, séparation retraite/tombstone, retrait des records sans terminal, EOF, continuation, thinking, panneau/cleanup, transition Activité terminale coordonnée unique, wake-up d'inactivité observer EasyAgents, validation brute des query strings, frontière React pour locator ambigu, snapshot brut figé, neutralisation autoRun, parse Open policy-aware sans resserrement global de ZChatSessionFile, application sélective sans lecture des champs draft, Activity, remount du Stop par sourceId, son avec reconnect même source, handoff direct sans second Open, draft/attachments, Stop pré-enregistrement, Stop disponible après transport replay échoué, claim destructif delete/Retry, génération destructive anti-ABA Open, éviction exigeant le handle exact, dispose différé sous claim, wake-up de release, start extension arrivé après dispose et éviction sans terminal.
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. À l'inverse, tenter d'appliquer ce

protocole à un `meta.error` tardif serait impossible après le header 200 et créerait un double chemin incohérent. Le callback généré peut en outre invoquer `onError` sans rejeter sa promesse, voire repasser par le callback si celui-ci throw ; le handler doit donc mémoriser l'issue de façon idempotente et décider après drainage. Un terminal wrapper synthétique doit être persisté avant append et ne doit jamais doubler un vrai terminal : sinon le handoff direct afficherait un état impossible à reconstruire après hydratation. Le frontend doit appliquer la même règle lorsqu'un transportError suit un terminal data, sans injecter un second SYSTEM_ERROR. 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. L'époque doit être revendiquée dès la réservation et l'issue mémorisée après retrait, sinon une source suivante peut ressusciter ou recevoir la finalisation de l'ancienne. Les retraites replayables ne peuvent pas servir seules d'anti-résurrection, car une éviction destructive doit pouvoir les retirer sans autoriser le même run exact à se réenregistrer. Le cleanup EasyAgents fermé doit être réveillé après le handoff observer sans introduire de cycle vers les panneaux. Le handoff normal doit également rester réellement direct : un second Open réintroduirait une course et pourrait écraser le compositeur que l'utilisateur a commencé à préparer. Le retry de replay doit utiliser une application d'hydratation explicitement sélective ; des setters no-op ou un parse canonique complet exécuté avant la policy laisseraient encore le helper parser et enregistrer une enveloppe de draft qui doit être ignorée. Une query live invalide sans locator univoque exige en outre une vraie frontière React, faute de quoi l'implémentation serait tentée de fabriquer une identité ou d'appeler conditionnellement les hooks. La transition Activité terminale doit avoir un propriétaire unique et son runId doit exister avant le premier await afin d'éviter les fuites de timer si le broadcast Sidebar échoue. Enfin, un simple « check puis await » ne ferme pas la course destructive : le claim locator doit protéger deletion et Retry, son handle exact doit autoriser l'éviction, sa génération distincte doit invalider les Open en vol sans casser les retraites terminales, et le dispose doit différer ses cleanups jusqu'au wake-up de release. Sans ces barrières, un cleanup de panneau pourrait courir en parallèle du delete, un source request retardé pourrait recréer une session après sa destruction, un Retry pourrait évincer la mauvaise source, ou un ancien run pourrait renaître après suppression de sa retraite. La reconstruction extension dans la FIFO unique, la persistance terminale préalable, l'éviction exacte des runs non-handoffables, les wake-ups observer/claim, les identités exactes, réservations, époques, tombstones, claims destructifs, guards de tentative, parsing brut des queries, frontière bootstrap, application sélective typée et tests d'isomorphisme sont donc obligatoires.

Résumé

- Étendre `liveSession` aux sources Prompt normales en une implémentation complète.
- Référencer le dépôt courant `c3d66083`, dont les sources ciblées restent identiques à la base tranche 1.
- Ajouter sourceId de bout en bout et streamInstanceId frontend unique servant de clé à chaque stream accepté.
- Distinguer `replayOpenAttemptId` pour les requêtes Open/retry, notamment lorsqu'un Open retourne directement terminal sans créer de stream.
- Rendre l'acceptation du start synchrone et transactionnelle, la propager jusqu'aux commandes/Prompt et n'armer les barrières locales qu'après acceptation.
- Tester transactionnellement tous les chemins de commande qui peuvent démarrer un stream.
- Vérifier que le panneau exact est encore ouvert avant la réservation extension ; rejeter tout start arrivé après dispose.
- Réserver puis enregistrer les sources, revendiquer l'époque dès la réservation, refuser les réservations exactes dupliquées, append-before-forward et partager le lifecycle.
- Allouer synchroniquement l'identité/timer Activité avant son broadcast afin qu'une erreur Sidebar reste auxiliaire et nettoyable.
- Drainer Queue A, Queue B et une FIFO extension unique avant publication des outcomes/finalisation.
- Décoder uniquement le premier header d'erreur source via un protocole V1 strict statut/code, l'enfiler comme chunk terminal exact, l'émettre comme donnée et conserver une complétion transport normale ; laisser tout meta.error tardif dans le body 200 normal.
- Ne jamais dépendre d'un throw du callback Slimbk `onError`; mémoriser et agréger l'issue après drainage.
- Ne jamais ajouter un second SYSTEM_ERROR frontend lorsqu'un vrai terminal data a précédé un transportError.
- Persister tout SYSTEM_ERROR/INTERRUPTED synthétique avec suffixe exact avant append, uniquement lorsqu'aucun vrai terminal n'existe ; en cas d'échec, interdire le faux handoff et retirer le run sans terminal.
- Finaliser atomiquement après drainage : `handoffReady` seulement avec baseline + vrai terminal durable ; sinon retraite/éviction exacte afin qu'aucun replay ne reste bloqué ; après une nouvelle époque, une répétition ancienne devient `stale`.
- Séparer retraites replayables et tombstones exactes anti-résurrection.
- Sérialiser la transition Activité terminale d'une source dans le coordinateur, avec un propriétaire unique commun aux courses open/close/settle.
- Réveiller le cleanup différé d'un panneau EasyAgents fermé lorsque son record observer devient inactif, sans dépendance EasyAgents → panneau.

- 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.
- Valider les descripteurs live depuis un snapshot brut complet et figé, avec détection des doublons et paramètres privés partiels/inconnus.
- Traiter les query live invalides sans locator par une frontière React externe, sans runtime Prompt ni locator factice.
- Désactiver tout autoRun dès qu’une query live explicite est détectée.
- Modéliser Activité avec ownership et panneau open/closed.
- Ajouter uniquement le Stop détaché source/closed et le remounter par sourceId.
- 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 fondé sur un modèle de bannière dédié.
- Garder le Stop backend disponible en replayFailed même lorsque le transport frontend est déjà terminé en erreur, puis consommer cette disponibilité après le premier clic.
- Effectuer le handoff normalPrompt directement sur l’état replayé, sans second Open.
- Préserver brouillon, attachments et nextTurnId lors des retries de replay au moyen d’un parse Open d’enveloppe puis d’une branche d’hydratation typée qui ignore totalement les champs draft du snapshot, y compris si le retry reçoit immédiatement un terminal.
- Faire retourner `void` à l’hydrateur et ne jamais caster des champs ignorés comme validés.
- Conserver le contrat de profondeur actuel de `ZChatSessionFile`; ne pas introduire un resserrement deep-strict sans rapport.
- Utiliser le registre comme autorité deletion/Retry, acquérir un claim destructif avant fermeture/delete et évincer les états retenus selon la policy de chaque opération.
- Exiger le handle exact pour l’éviction destructive, utiliser une génération destructive distincte pour invalider les Open ABA et réveiller le coordinateur à la libération du claim.
- Empêcher tout cleanup de dispose de courir en parallèle du delete/Retry ; reprendre ce cleanup seulement après release et revalidation.
- Préserver EasyAgents, queues, locks et sanitizers.
- Mettre à jour les programming docs liveSession, EasyAgents, isomorphisme, thinking, dispatch Queue B et fallback persist.
- Tester stream continu, replay, hydratation, erreurs source encodées, meta.error tardif en body, terminaux synthétiques persistés sans double terminal, terminal data suivi d’un transportError, transport réel, guards stale, Queue B différée, wake-ups observer/claim, start refusé pour chaque commande, snapshot brut figé, locator ambigu, application sélective, transition Activité unique, publication Activity auxiliaire, claims destructifs, génération Open anti-ABA, delete/Retry des états retenus 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_175152

Copy

Refine

Code

Verify

Niveau	Changement appliqué à la spécification	Justification
🟥 Majeur	Renforcement complet du **claim destructif de locator** : rejet de toute réservation ou de tout record actif quel que soit l’ownership, éviction publique autorisée uniquement avec le handle opaque exact, et conservation explicite du claim jusqu’à sa libération.	Le plan précédent protégeait le démarrage de nouvelles sources, mais n’empêchait pas formellement un caller non propriétaire d’évincer un locator pendant le claim.
🟥 Majeur	Ajout d’une **génération destructive distincte de l’époque de run**, avancée à l’acquisition et à la libération du claim, puis	Un Open commencé avant un cycle rapide `claim`

Niveau	Changement appliqué à la spécification	Justification
	capturée et revalidée par les opérations Open.	`acquire → release` pouvait autrement traverser l'opération destructive sans détecter l'ABA. Une génération séparée évite aussi d'invalider à tort une retraite terminale lorsqu'un delete échoue.
🟥 Majeur	Ajout d'un événement immuable **« claim destructif libéré »** et report obligatoire des cleanups de dispose pendant le claim. Le coordinateur reprend ensuite le cleanup sur ce wake-up, après revalidation du panneau et des sources.	La fermeture du panneau déclenchée par Delete ou Retry pouvait lancer en parallèle `Abort/Await/ClearMemory`, l'orphan cleanup, le cleanup sonore ou le cleanup trivial, créant une course directe avec les opérations disque destructives.
🟥 Majeur	Séparation de `replayOpenAttemptId` et `streamInstanceId` : chaque séquence Open/retry possède une identité d'Open, tandis que `streamInstanceId` n'existe que lorsqu'un stream est effectivement accepté.	Le plan précédent affirmait simultanément que chaque retry créait un `streamInstanceId` et qu'un Open pouvait retourner directement `terminal` sans démarrer de stream. La nouvelle distinction supprime cette contradiction et garde aussi les réponses/timers Open stale.
🟥 Majeur	Précision du traitement **terminal data puis transportError** : conserver le terminal exact, publier `transportError`, mais ne jamais injecter un second `meta.error` SYSTEM_ERROR ni un second suffixe.	Une erreur auxiliaire agrégée ou une erreur de transport après un terminal déjà reçu pouvait produire un deuxième terminal frontend, casser l'isomorphisme et masquer la vraie fin de source.
🟥 Majeur	L'`activityRunId`, le timer et le runtime Activité sont désormais alloués **synchroniquement avant le premier await** ; la publication Sidebar devient une tâche auxiliaire dont l'échec n'empêche pas la source backend de démarrer.	Avec l'API async actuelle, un échec de broadcast après mutation pouvait perdre le runId retourné, arrêter indûment la génération ou laisser un timer impossible à finaliser exactement.
🟧 Mineur	Centralisation explicite de `easyAgentsBatchLineNumber` avec la liste des paramètres liveSession privés dans le contrat shared.	L'extension et le frontend doivent utiliser une seule définition canonique pour éviter toute divergence de classification des queries explicites.

Niveau	Changement appliqué à la spécification	Justification
🟧 Mineur	Clarification de la portée de la validation Open complète : la corrélation des branches reste stricte, mais cette tranche ne rend pas globalement `ZChatSessionFile`, ses messages opaques ou ses métadonnées historiques deep-strict.	Le wording précédent pouvait conduire à un resserrement non demandé du schéma de session, avec un risque de rejet des métadonnées historiques ou EasyAgents sans rapport avec l'application sélective.
🟧 Mineur	Extension des tests et des programming docs aux nouveaux contrats : ABA des claims, wake-up après release, dispose différé, éviction avec handle, Open terminal sans stream, terminal suivi d'une erreur transport et publication Activité auxiliaire.	Ces scénarios constituent les preuves de non-régression nécessaires pour les corrections ajoutées pendant ce raffinement.

</ASSISTANT>