

Notifications et présence

Notifications envoyées depuis Unity

Le Game envoie les notifications au backend Web. `NotificationManager.SaveNotification()` lance la coroutine `BackEndBase.SaveNotification()`, qui effectue `POST /saveNotification` avec `userId`, `notificationType`, `isImportant` et `extraData`.

`userId` est le profil jeu du destinataire (`products_code.game_oid`); `extraData.simUid` identifie la simulation. `NotificationController`, protégé par `AuthGuardUser`, délègue à `GameNotificationService.notifyGameProfile()`.

Le serveur vérifie le format, exclut les types réservés au serveur et contrôle que l'expéditeur et le destinataire appartiennent à la simulation BrainCloud. Le profil jeu destinataire est ensuite converti en compte Web pour `NotificationService.notify()`.

La notification est enregistrée dans l'inbox Redis avant son annonce aux connexions du compte. Le push est tenté ensuite; un échec du push n'annule pas l'entrée enregistrée. L'inbox possède une limite de taille et une durée de rétention configurées dans `NOTIFICATION_CONFIG`.

Le Web charge `GET /notifications`, marque une entrée comme lue via `POST /notifications/:notificationUid/read` et la supprime via `DELETE /notifications/:notificationUid`. La réception live utilise `notification.created` sur le socket du compte `/account`.

Voir [Notifications et Push](#) pour le système Web, et [Panel Web — interface superposée au jeu et intégration Unity](#) pour son affichage au-dessus du jeu.

Trois mécanismes de présence

Mécanisme	Donnée et usage	Code
Présence du compte Web	Statut online/away et jeu actif du compte; utilisé par les interfaces Web.	<code>AccountSocketService</code> , socket <code>/account</code> , <code>updateStatus</code> et <code>updateGamePresence</code> .

Mécanisme	Donnée et usage	Code
Heartbeat du Game	Session du jeu et derniers heartbeats utilisés par l'UI Unity pour les utilisateurs de la simulation.	<code>User.DoHeartbeat()</code> , endpoint <code>/heartBeat</code> et <code>UsersLastHeartbeats</code> .
Présence évaluée By Year	Participation enregistrée par étudiant, activité et année, utilisée pour calculer un score.	<code>_command/set_presence.js</code> et <code>evaluation.presence[actUId][yearId]</code> .

Le nouveau statut Web ne remplace pas automatiquement les deux usages Game : le code local conserve des appels et des consommateurs distincts.

Présence Web du compte

`AccountSocketService` suit l'activité du navigateur, le focus et l'inactivité pour émettre `online` ou `away`. `PageWebglComponent` appelle `setGamePresence(pid, gameId)` après validation de l'accès au jeu, puis `clearGamePresence()` à sa destruction.

À la reconnexion du socket, le service réémet le jeu actif conservé en mémoire. Cette information désigne le jeu ouvert; elle ne crée pas d'entrée de présence pédagogique dans BrainCloud.

Heartbeat utilisé par Unity

`LoadUserState` démarre le heartbeat. `User.DoHeartbeat()` utilise `HEARTBEAT_RATE_SECONDS = 5` : l'intervalle d'attente est de 5 secondes entre les appels. La réponse peut mettre à jour les derniers heartbeats dans le controller de simulation.

`SimulationBaseController.IsUserOnline()` et `ActivityBaseController.IsUserOnline()` considèrent un utilisateur online si le dernier heartbeat date de moins de $5 \times 12 = 60$ secondes.

`ProfileImageControl` consomme cet état pour l'affichage. Le heartbeat a donc encore un consommateur Unity.

Présence évaluée By Year

`BackEndBrainCloud.SetPresence()` appelle `_command/set_presence`. Le script traite seulement les utilisateurs avec `isTeacher == 0`, charge leur simulation et l'activité courante, puis enregistre la présence uniquement si `currentYearState == 1`.

`setYearPresence()` écrit `true` dans `evaluation.presence[actUId][yearId]` si l'entrée n'existe pas déjà, sauvegarde le document et envoie `user_new_presence_eval` à l'étudiant et au professeur.

`getPresenceScore()` compte les années enregistrées entre 0 et `yearId`, puis divise le cumul de 100 points par le nombre d'années. `StudentState` et `ActivityByYearController` appellent encore `SetPresence()` dans leurs transitions.

`startYear()` envoie le stub de score à chaque étudiant; il ne marque pas tous les étudiants présents. L'enregistrement effectif relève de `set_presence`.

Réouverture annuelle

`utils_reopen_year.js` appelle `deleteYearPresence(student, actData.currentYear + 1, actData)` après ajustement de l'année courante. Ce flux appartient au cycle des évaluations By Year.

Usages conservés dans le Game

Le heartbeat alimente l'UI Unity, et la présence By Year alimente les évaluations BrainCloud. La présence du compte Web décrit le compte connecté et son jeu actif; ces deux mécanismes Game conservent leurs consommateurs propres.

Sources vérifiées

Game : `Assets/Src/Game/Notifications/NotificationManager.cs`, `Game/BackEnd/BackEndBase.cs`, `BackEndBrainCloud.cs`, `Game/User.cs`, `Game/GameStates/StudentState.cs`, `LoadUserState.cs`, les controllers de simulation/activité et `UI/ProfileImageControl.cs`. BrainCloud : `_command/set_presence.js`, `_utils/utils_by_year.js`, `utils_eval.js` et `_utils/command/utils_reopen_year.js`.

Web : `uniksim-back-end/src/app.controller.ts`, `service/game-notification.service.ts`, `notification.service.ts`; frontend : `service/account/account-socket.service.ts` et `Pages/PageWebgl/PageWebgl.component.ts`.

Vérification par lecture du code Game et BrainCloud de la branche `refactor_notifications`, et du code Web de la branche `messaging_fixes`, le 30 septembre 2026. Aucun envoi de notification ou changement de présence réel n'a été exécuté pour cette vérification.