

Commandes serveur, réouverture et transitions By Year

Entrées communes

Le framework annuel est exposé par des commandes communes et par des wrappers game-specific.

Commandes communes

- `change_year_state` : change l'état annuel;
- `student_take_decisions` : marque/enregistre la prise de décisions selon le flux annuel;
- `get_activity_current_year` : année courante;
- `ignore_year_from_average` : exclut une année de certaines moyennes;
- `save_market_size_modifier` : modificateur de marché;
- `set_year_sequence_seen` : état de séquence côté utilisateur.

Wrappers game-specific

Les dossiers `ent1/command`, `hr1/command`, `mgt1/command`, `mk1/command` et `mk2/command` exposent typiquement :

- `<game>_create_team`;
- `<game>_custom_command`;
- `<game>_generate_results`;
- `<game>_start_year`;
- `<game>_start_next_activity`;
- `<game>_reopen_year`;
- `<game>_compute_scenario_score`;

- `<game>_compute_segment_score`.

Certains jeux ajoutent des wrappers propres, par exemple des mises à jour de rankings.

Réouverture d'année

`_utils/command/utils_reopen_year.js` est la logique partagée. Une réouverture ne se limite pas à décrémenter `currentYear` : il faut supprimer/retourner les résultats, nettoyer les évaluations annuelles, logbook/presence associés et remettre l'activité/équipes dans un état cohérent.

Transition activité suivante

`utils_start_next_activity.js` gère la transition entre activités. Les modes sync/async et le GameUid influencent les messages et l'activité chargée côté clients.

Custom commands

`utils_custom_command_by_year.js` est le point commun de dispatch lorsque la commande n'est pas directement implémentée dans le wrapper game-specific. Les règles métier restent dans `utilsByGameUid[gameUid]`.

Debug

```
flowchart LR
    UNITY[BackEndBrainCloud] --> WRAP[gameUid/command]
    WRAP --> COMMON[_utils/command]
    COMMON --> SPEC[game-specific utils]
    SPEC --> DOC[activity / teams / results]
    DOC --> RTT[act_year_* / result messages]
```

Ajouter une Custom Command By Year

Les commandes communes comme market research, decisions finished ou financial help passent par `utils_custom_command_by_year.js` ; les décisions métier propres au jeu doivent rester dans les utils game-specific.

Procédure

1. Définir `commandUid` et la payload.
2. Brancher le wrapper `<gameUid>_custom_command`.
3. Si commande commune, ajouter le cas dans `executeCommandByYear()`.
4. Sinon déléguer vers `utilsByGameUid[gameUid]` / utils custom du jeu.
5. Modifier `teamDecisionsData` / `teamData` / `actData` au bon niveau.
6. Sauvegarder les documents concernés.
7. Envoyer un RTT avec la payload minimale.
8. Ajouter le modèle/handler Unity.

Réouverture d'année — système destructif

`utils_reopen_year.reopenYear()` ne doit pas être traité comme un simple changement de state. Il doit remettre en cohérence :

- activity/current year;
- results;
- team decisions/cash si le jeu les reset;
- evaluationTeam de l'année;
- logbook annuel;
- présence;
- events/produits game-specific lorsque le moteur le prévoit;
- rankings si nécessaire.

Quand une nouvelle donnée annuelle est ajoutée

Auditer immédiatement `reopenYear()`, `GenerateResults<Game>.deleteResults()`, `deleteProducts()` et les helpers de reset du jeu. Si la donnée existe seulement après génération, elle doit généralement être supprimée/restaurée lors d'une réouverture.

Transition activité suivante

`utils_start_next_activity.js` porte la logique commune. Un nouveau jeu ou nouveau type d'activité doit être testé avec :

- activité précédente terminée;
- création/chargement de la suivante;
- première année de la suivante;
- étudiants/teams copiés ou regroupés;
- async vs sync;
- RTT `next_act_started` / `async_next_act_started`.

Revision #2

Created 2026-09-18 19:26:56 UTC by Mathis

Updated 2026-09-18 20:17:51 UTC by Mathis