

Tests automatisés des Server Scripts

But

Les tests automatisés valident la logique métier des Cloud Scripts sans passer par l'UI Unity. Ils servent surtout à protéger les moteurs complexes By Module : création des données, bills, answer key, grading, inventaire et factories.

Emplacement

Les tests se trouvent sous `node.js/backend/braincloud/tests/`.

Dossier	Ce qui est testé
<code>unit/by_module_tests/answer_key_tests/</code>	Génération des bonnes réponses, fonctions auxiliaires, petty cash.
<code>unit/by_module_tests/bills_tests/</code>	Création et utilitaires des bills.
<code>unit/by_module_tests/classes_tests/</code>	Classes métier comme l'inventaire.
<code>unit/by_module_tests/create_module_type_data_tests/</code>	Génération initiale du Module Type Data.
<code>unit/by_module_tests/factories_tests/</code>	Factories serveur, notamment <code>BillFactory</code> .
<code>unit/by_module_tests/grade_module_tests/</code>	Correction, états financiers, formulaires fiscaux et modules spécialisés.

Infrastructure de test

- `tests/lib/test_base.js` : base commune;
- `tests/lib/unit_test_core.js` : orchestration des tests;
- `tests/utils/test_data.js` : helpers/données de test;

- `tests/data/` : fixtures (companies, conditions, employees, partners, products, salaries, services);
- `tests/utils/billtypedata.json` et `reference.json` : références utilisées par les suites.

Quand ajouter un test

Ajouter ou modifier un test dès qu'un changement touche une fonction pure ou déterministe du moteur serveur. Les zones à protéger en priorité sont :

- nouveau type de bill;
- nouvelle règle d'answer key;
- nouvelle règle de grading;
- modification de création du Module Type Data;
- modification d'inventaire ou de calcul qui influence plusieurs pages;
- nouvelle factory ou branche de factory.

Stratégie

```
flowchart LR
    CHANGE[Changement métier] --> UNIT[Test ciblé]
    UNIT --> EDGE[Cas limites]
    EDGE --> REG[Régression du module]
    REG --> DEBUG[Debug local si le problème dépend de documents réels]
```

Tests vs debug local

Les tests automatisés ne remplacent pas **Debug local des Server Scripts BrainCloud**. Un test unitaire valide une fonction avec des fixtures contrôlées; le debug local permet d'exécuter un vrai Cloud Script avec des documents issus de MongoDB ou récupérés depuis BrainCloud.

Lancer réellement les tests

Le projet possède un runner Web, pas seulement des fichiers `.test.js`.

Runner Web recommandé

Depuis `node.js/backend/braincloud/` :

```
node tests/web-runner/server.js
```

Le serveur écoute par défaut sur `http://localhost:3000` et ouvre automatiquement le navigateur. L'UI liste les suites sous `tests/unit/by_module_tests`, permet de sélectionner des tests individuels et de lancer en mode normal ou debug.

Mode normal vs debug

- **normal** : les fichiers By Module sont instrumentés avec Istanbul et un rapport HTML est généré sous `tests/test-reports/`;
- **debug** : `global.DEBUG_MODE = true`, l'instrumentation coverage est désactivée et aucun rapport n'est généré.

Dépendance MongoDB locale

`tests/lib/test_base.js` crée `MongoDBClient` et s'authentifie avant d'initialiser les utils. Si les tests ne démarrent même pas, vérifier d'abord le MongoDB local et sa configuration avant de chercher une régression métier.

Point à connaître : game-specific de test

Le bootstrap de test inclut actuellement explicitement `/acc2/utils/acc2_utils`. Si une nouvelle suite dépend réellement d'une différence Acc1/Acc2, ne pas supposer que le runner change automatiquement de game-specific.

Créer un nouveau test

1. Placer le fichier sous la catégorie appropriée dans `tests/unit/by_module_tests/`.
2. Créer un `TestSuite`.
3. Ajouter les cas avec `suite.addTest(name, fn)`.
4. Initialiser la fixture commune avec `suite.init()` si nécessaire.
5. Utiliser les assertions de `unit_test_core.js` : `areEqual`, `isTrue`, `inRange`, etc.
6. Ajouter les fixtures réutilisables dans `tests/data` ou `tests/utils`, pas en duplicata dans chaque test.

7. Lancer le test seul, puis la suite complète du domaine.

Quand un test doit être ajouté

Tout changement Answer Key/Grading/Create Module Type Data/BillFactory/inventory doit avoir au minimum un test qui aurait échoué avant le changement. Un bug corrigé sans test de régression est considéré incomplet lorsque la fonction est testable localement.

Revision #2

Created 2026-09-18 19:12:51 UTC by Mathis

Updated 2026-09-18 20:09:27 UTC by Mathis