# Manuel de Référence - Usine à Tests V3.0 **(Version Détaillée)**

*Dernière mise à jour : 22 août 2025*

## Partie 1 : Vision, Principes et Architecture

### 1.1. Introduction et Vision du Projet

L'**Usine à Tests** est un système intégré et automatisé conçu pour gérer l'ensemble du cycle de vie des questionnaires au sein de l'écosystème Google Workspace.

**L'objectif global** est de transformer une simple ligne de configuration en un système de test complet et autonome. Cette approche vise à résoudre plusieurs problématiques clés :

* **Standardisation** : Assurer que tous les tests partagent une structure, une logique de traitement et une qualité de code communes.
* **Rapidité de déploiement** : Réduire drastiquement le temps de déploiement d'un nouveau test de plusieurs jours à quelques minutes.
* **Fiabilité** : Éliminer les erreurs manuelles inhérentes à la copie, la modification et la configuration de multiples scripts et fichiers.
* **Flexibilité** : Offrir une plateforme capable de générer une large gamme de tests avec des logiques de calcul et de communication variées sans modification du code de base.

Le produit final de l'usine est un "Kit" autonome incluant :

* Un **Google Form** multilingue et dynamique, prêt à être partagé.
* Une **Google Sheet** de traitement des réponses ("Kit de Traitement") dotée d'une logique de calcul et de communication embarquée.
* Un système d'envoi d'**e-mails de résultats** personnalisés, multilingues, avec pièces jointes et pouvant être différés dans le temps.

### 1.2. Contexte et Évolution

Le projet a maturé à travers plusieurs phases majeures pour atteindre sa version actuelle, stable et robuste :

* **Version 0 - Le Prototype Artisanal** : Un système monolithique où une seule feuille de calcul gérait un unique test. Bien que fonctionnel, il était impossible à dupliquer ou à faire évoluer sans un travail manuel considérable et risqué.
* **Version 1 - L'Usine Spécialisée** : Introduction de la séparation fondamentale entre l'usine de génération [MOTEUR] et le produit [TEMPLATE]. Cependant, chaque "Kit" généré contenait encore une logique de calcul qui lui était propre (code "en dur"). L'ajout d'un nouveau type de test nécessitait de modifier le code du [TEMPLATE], ce qui créait des branches de maintenance complexes.
* **Version 2 - L'Usine Universelle (Actuelle)** : L'architecture actuelle où **toute la logique métier est externalisée dans les données**. Le code n'est plus qu'un interpréteur intelligent de la configuration. Cette version a été considérablement renforcée par :
  + La mise en place d'un **moteur de calcul universel** (Logique\_Universel.gs) piloté par des structures JSON, rendant le système agnostique au type de test.
  + La transformation du système d'e-mails en un **moteur d'assemblage dynamique** qui construit les communications en piochant des "briques" de contenu dans la base de données (sys\_Composition\_Emails).
  + L'ajout de **fonctionnalités métier avancées** comme l'e-mail de confirmation et l'envoi différé, entièrement pilotées par la configuration.
  + Le développement d'un **framework de débogage** (LoggerLib) et d'interfaces utilisateur (UI) pour la configuration et le retraitement, améliorant l'ergonomie et la maintenabilité.

### 1.3. Principes Directeurs

Le développement du projet respecte trois principes fondamentaux qui garantissent sa robustesse et sa maintenabilité.

1. **Zéro Convention Implicite** : Tout paramètre ou toute logique doit être explicitement défini dans la configuration. Le code ne doit rien deviner.
   * *Exemple concret* : La résolution du bug des e-mails non reçus était due à une clé de traduction (PREFIXE\_COPIE\_EMAIL) manquante dans la BDD. Le système a échoué de manière prévisible car l'information n'était pas explicitement fournie, prouvant la validité de ce principe.
2. **Le JSON comme Source de Vérité** : Toute la logique de calcul d'une question (son mode de fonctionnement, les profils impactés, les scores attribués) est contenue dans le champ Paramètres (JSON) de la base de données. Le code n'est qu'un interpréteur de cette configuration.
   * *Exemple concret* : La logique du mode ECHELLE\_NOTE (bornes inférieure/supérieure, labels textuels) est entièrement définie dans le JSON. Le script lit ces paramètres pour construire la question dans le Google Form sans aucune connaissance préalable de ce que la question signifie.
3. **Le Code plus Intelligent que la Configuration** : Le code doit être conçu pour anticiper et se défendre contre des erreurs de configuration humaines et gérer des cas variés de manière robuste.
   * *Exemple concret* : L'ajout systématique de la fonction .trim() lors de la lecture des ID de pièces jointes a résolu un bug critique causé par un caractère de saut de ligne invisible. Le code nettoie désormais la donnée avant de l'utiliser, le rendant insensible à ce type d'erreur de saisie.

### 1.4. Architecture Globale des Composants

Le système s'articule autour de cinq composants principaux qui interagissent en permanence pour transformer une idée en un test fonctionnel.

* **1. [CONFIG] Usine à Tests (Google Sheet)**
  + **Rôle** : Le panneau de contrôle central, le point de départ de toute création. C'est ici que l'administrateur décrit les tests à générer via une interface de configuration (FormulaireUI.html) ou en modifiant directement l'onglet Paramètres Généraux.
  + **Scripts associés** : 02\_configuration\_Script\_250822.txt
* **2. [MOTEUR] Usine à Tests (Apps Script)**
  + **Rôle** : L'orchestrateur de construction. Ce script est le "bras armé" de l'usine. Il lit une ligne de configuration dans [CONFIG], interprète la demande, et construit l'ensemble des fichiers (le Formulaire et le Kit de traitement) en allant chercher les ressources nécessaires dans la BDD.
  + **Scripts associés** : 01\_Moteur\_Script\_250822.txt
* **3. [BDD] Tests & Profils (Google Sheet)**
  + **Rôle** : La base de données centrale, le "cerveau" de l'usine. Elle contient toutes les données métier nécessaires au fonctionnement des tests : textes des questions, logique de scoring, descriptions des profils, contenu des e-mails, traductions, etc.
  + **Scripts associés** : 03\_BaseDeDonnées\_Script\_250822.txt (contient des utilitaires pour la gestion de la BDD).
* **4. [TEMPLATE] Kit de Traitement V2 (Google Sheet)**
  + **Rôle** : Le modèle de feuille de calcul autonome qui est copié pour chaque nouveau test. Son script interne est le "moteur d'exécution" qui prend en charge une réponse de formulaire, la traite, calcule les scores et gère toute la communication par e-mail.
  + **Scripts associés** : 04\_Templates\_Script\_250822.txt
* **5. LoggerLib (Apps Script Library)**
  + **Rôle** : Une librairie de script partagée et externe dont l'unique rôle est de fournir des fonctions de journalisation (logging) standardisées pour le débogage. Elle permet d'activer un mode "verbeux" pour analyser le comportement d'un Kit sans altérer son code de production.

### 1.5. Dépendances et Prérequis

* **Librairie Externe** : Le [TEMPLATE] dépend de la librairie LoggerLib pour le débogage. Son ID de script doit être ajouté dans les dépendances du projet Apps Script du template pour qu'il soit accessible.
* **Permissions Google Workspace** : Pour fonctionner, les scripts requièrent des autorisations de la part de l'utilisateur qui les exécute pour accéder aux services Google suivants :
  + **Google Drive** (pour créer, copier et gérer les fichiers et dossiers).
  + **Google Sheets** (pour lire la configuration, la BDD, et écrire les résultats).
  + **Google Forms** (pour créer et modifier dynamiquement les questionnaires).
  + **Gmail** (pour envoyer les e-mails de confirmation et de résultats, potentiellement via un alias).

## Partie 2 : Guides d'Utilisation

### 2.1. Guide de Démarrage Rapide : Déployer un Test en 3 Étapes

Le moyen le plus simple et le plus fiable de créer un nouveau test est d'utiliser le processus de déploiement centralisé.

* **Étape 1 : CONFIGURER (Décrire le test)**
  1. Ouvrez la feuille de calcul [CONFIG] Usine à Tests.
  2. Dans le menu, cliquez sur **🚀 Actions Usine -> Configurer un nouveau test...**.
  3. Remplissez le formulaire qui apparaît dans la barre latérale. C'est ici que vous prenez les décisions clés : le titre public, le type de test (qui définit le contenu), les questions standards à inclure (blocs méta), et les options d'envoi d'e-mail (qui reçoit quoi, et quand).
  4. Cliquez sur "Créer la Configuration". Une nouvelle ligne est ajoutée dans l'onglet Paramètres Généraux avec le statut **"En construction"**. Notez le numéro de cette ligne.
* **Étape 2 : DÉPLOYER (Construire le test)**
  1. Ouvrez le projet Apps Script [MOTEUR] Usine à Tests.
  2. Dans le menu, cliquez sur **🏭 Usine à Tests -> 🚀 Déployer un test de A à Z...**.
  3. Une boîte de dialogue vous demandera le numéro de la ligne à déployer. Entrez le numéro noté précédemment. Le moteur va alors lire la configuration, créer les fichiers, les lier entre eux et peupler le formulaire avec les questions de la BDD.
* **Étape 3 : ACTIVER (Mettre le test en service)**
  1. Une fois le processus terminé, une fenêtre de confirmation s'affiche avec deux liens. Cliquez sur le second : **"Ouvrir le Kit pour l'activer"**.
  2. Dans le menu de cette nouvelle feuille de calcul (le "Kit de Traitement"), cliquez une seule fois sur **⚙️ Actions Usine -> Activer le traitement automatique**. Cette action crée le déclencheur onFormSubmit qui écoutera les futures soumissions du formulaire. C'est une étape de sécurité requise par Google.

**Votre test est maintenant 100% opérationnel.** Le lien public du formulaire est disponible dans la fenêtre de confirmation et est également sauvegardé pour référence dans la colonne Lien\_Formulaire\_Public de votre feuille [CONFIG].

### 2.2. Guide des Opérations Courantes

* Retraiter une Réponse  
  Cette fonctionnalité est essentielle pour le support et le débogage.
  + **Cas d'usage** : Un participant a fait une erreur dans son adresse e-mail ; vous souhaitez envoyer une copie des résultats à un manager ; vous avez corrigé un bug dans le script et voulez retraiter une réponse qui avait échoué.
  1. Ouvrez le "Kit de Traitement" concerné.
  2. Allez dans le menu **⚙️ Actions Usine -> Retraiter une réponse...**.
  3. Entrez le **numéro de la ligne** de la réponse à retraiter.
  4. Une **barre latérale** s'ouvrira, vous permettant de forcer une langue pour l'e-mail, de changer le niveau de détail du résultat, de spécifier un alias d'expédition ou de modifier la liste des destinataires avant de lancer le retraitement.
* Ajouter des Pièces Jointes PDF  
  Le système permet de joindre dynamiquement des fichiers stockés sur Google Drive.
  1. Ouvrez la feuille de calcul [BDD] Tests & Profils.
  2. Dans le menu, cliquez sur **⚙️ Utilitaires BDD -> Lister les fichiers d'un dossier Drive**.
  3. Entrez l'ID du dossier Drive contenant vos fichiers PDF. L'utilitaire va scanner le dossier et ajouter le nom et l'ID de chaque fichier dans l'onglet Liste\_Fichiers\_Drive.
  4. Vous pouvez maintenant aller dans l'onglet sys\_Composition\_Emails et créer une nouvelle ligne avec l'élément de type Document, en collant l'ID du fichier désiré dans la colonne Contenu / ID\_Document.

## Partie 3 : Dictionnaire des Données et Configuration

### 3.1. Le Dictionnaire Complet des Données

#### Fichier [CONFIG]V2 Usine à Tests

C'est le panneau de contrôle de l'usine. Sa structure doit être respectée scrupuleusement.

* **Onglet Paramètres Généraux** : Le "carnet de commandes". Chaque ligne est une instruction de fabrication.
  + Type\_Test : Code du test (ex: "Couleurs"). C'est la clé principale qui détermine quelles questions (Questions\_Couleurs\_FR) et quels profils (Profils\_Couleurs\_FR) seront utilisés depuis la BDD.
  + Moteur\_Calcul : Doit être **"Universel"** pour la version actuelle.
  + Blocs\_Meta\_A\_Inclure : Liste des ID des questions standards (ex: META\_EMAIL,META\_NOM\_COMPLET), séparés par des virgules.
  + ID\_Gabarit\_Email\_Confirmation\_[LANGUE] : (Optionnel) Permet de surcharger le gabarit de confirmation par défaut (défini dans sys\_ID\_Fichiers) avec un Google Doc spécifique pour ce test et cette langue.
  + Repondant\_Quand : Définit le délai d'envoi des résultats. Les valeurs (Immediat, 4h, 24h, etc.) proviennent de sys\_Options\_Parametres. Une valeur autre que Immediat active la logique d'envoi différé.
  + Email\_Alias : (Optionnel) Permet de spécifier une adresse d'expédition alternative qui doit être préalablement configurée et vérifiée dans vos paramètres Gmail.
  + ID\_Sheet\_Cible, ID\_Formulaire\_Cible, Lien\_Formulaire\_Public : Ces colonnes sont remplies automatiquement par le [MOTEUR] lors du déploiement et ne doivent pas être modifiées manuellement.
* **Onglet sys\_ID\_Fichiers** : Table de correspondance centrale pour les ID des fichiers critiques.
  + ID\_BDD : L'ID de la feuille de calcul [BDD] Tests & Profils.
  + ID\_TEMPLATE\_TRAITEMENT\_V2 : L'ID de la feuille de calcul [TEMPLATE] Kit de Traitement V2.
  + ID\_GABARIT\_CONFIRMATION\_[LANGUE] : L'ID du Google Doc servant de gabarit de confirmation **par défaut** pour chaque langue.
* **Onglet sys\_Options\_Parametres** : Alimente les menus déroulants de l'interface de configuration pour garantir la cohérence des données saisies.

#### **Fichier [BDD]** Tests & Profils

C'est le cerveau du système. Toute la logique métier y est stockée.

* **Onglet Questions\_[Type]\_[Langue]** : Stocke les questions.
  + ID : Identifiant unique et invariant de la question.
  + Paramètres (JSON) : **Le champ le plus important**. Contient un objet JSON qui décrit entièrement le comportement de la question.
    - "mode" : Le type de traitement (QCU\_CAT, QRM\_CAT, ECHELLE\_NOTE).
    - "options" : Un tableau d'objets pour les questions à choix, où chaque objet contient :
      * "libelle" : Le texte de l'option de réponse, qui doit correspondre **exactement** à celui du formulaire.
      * "profil" : Le code du profil à incrémenter.
      * "valeur" : Le nombre de points à ajouter.
    - "profil" : Pour les modes comme ECHELLE\_NOTE, le profil unique à impacter.
* **Onglet Profils\_[Type]\_[Langue]** : Stocke les textes descriptifs pour chaque profil de résultat.
* **Onglet sys\_Composition\_Emails** : Le pilier du moteur d'assemblage d'e-mails. Chaque ligne est une "brique" de contenu. Le script filtre ces briques en fonction du type de test, de la langue, du niveau de l'e-mail et du profil du répondant pour construire l'e-mail final. Les Element possibles sont :
  + Sujet\_Email : Définit la ligne de sujet.
  + Introduction, Corps\_Texte : Blocs de texte statique.
  + Document : Spécifie l'ID d'un fichier Drive à joindre.
  + Ligne\_Score : Un modèle de ligne qui sera répété pour chaque profil, affichant le nom et le score.
  + Info\_Copie : Un bloc de texte qui n'est ajouté qu'au début des e-mails envoyés en copie (non au destinataire principal).
* **Onglet traductions** : Centralise les chaînes de texte statiques utilisées par les scripts (PREFIXE\_COPIE\_EMAIL, etc.). Une clé manquante pour une langue peut empêcher l'envoi d'un e-mail.
* **Onglets Obsolètes** : Pour le moteur universel, l'onglet sys\_PiecesJointes n'est plus utilisé. La logique des pièces jointes est désormais gérée directement dans sys\_Composition\_Emails avec l'élément de type Document.

### 3.2. Fonctionnalités Clés Pilotées par les Données

* **Les Blocs Méta Dynamiques** : L'interface de configuration (FormulaireUI.html) n'a pas de questions codées en dur. Elle lit dynamiquement l'onglet Questions\_META\_FR de la BDD pour afficher les questions standards disponibles sous forme de cases à cocher. Cela permet d'ajouter de nouvelles questions standards (ex: "Service/Département") sans modifier une seule ligne de code.
* **L'Envoi d'E-mail Différé** : Cette fonctionnalité est entièrement pilotée par la colonne Repondant\_Quand dans Paramètres Généraux. Si la valeur n'est pas "Immediat", le script TraitementReponses.gs appelle programmerEnvoiResultats() du fichier GestionTriggers.gs, qui crée un déclencheur temporel pour exécuter l'envoi plus tard. Le script GestionTriggers.gs est conçu pour s'auto-nettoyer : il exécute la tâche, supprime les données temporaires de la tâche, puis supprime le déclencheur lui-même.

## Partie 4 : Architecture des Scripts, Maintenance et Dépannage

### 4.1. Cartographie des Scripts

* **Scripts du [MOTEUR]** (01\_Moteur\_Script\_250822.txt)
  + MoteurV2.js : Contient la fonction principale lancerDeploiementComplet().
  + UtilsV2.js : Fonctions utilitaires pour lire la configuration (getConfigurationFromRow) et créer les items du formulaire (creerItemFormulaire).
  + InterfaceV2.js : Gère la création du menu utilisateur et les boîtes de dialogue.
* **Scripts du [TEMPLATE]** (04\_Templates\_Script\_250822.txt)
  + TraitementReponses.gs : Le cœur logique du kit. Contient onFormSubmit(), traiterLigne() (le chef d'orchestre du traitement) et assemblerEtEnvoyerEmailUniversel() (le constructeur d'e-mails).
  + Utilities.gs : Boîte à outils pour lire la configuration centrale (getTestConfiguration), les traductions (loadTraductions), etc.
  + Logique\_Universel.gs : Le moteur de calcul qui interprète le Paramètres (JSON) pour déterminer les scores (calculerResultats).
  + GestionTriggers.gs : Gère la création (programmerEnvoiResultats) et l'exécution (envoyerEmailProgramme) des envois d'e-mails différés.
  + Menu.js, RetraitementUI.html, DialogueLigne.html : L'ensemble des fichiers qui gèrent l'interface de retraitement.

### 4.2. Maintenance et Dépannage

* Procédure de Mise à Jour d'un Kit  
  Lorsqu'une modification est apportée aux scripts du [TEMPLATE], elle n'est pas automatiquement propagée aux Kits déjà générés. Pour mettre à jour un kit existant :
  1. Ouvrir le projet Apps Script du [TEMPLATE].
  2. Copier le contenu du ou des fichiers mis à jour.
  3. Ouvrir le projet Apps Script du Kit à mettre à jour.
  4. Coller le nouveau code en remplacement de l'ancien.
* Le Framework de Débogage  
  Pour analyser un problème sur un kit spécifique :
  1. Ouvrez l'éditeur de script du Kit de Traitement à analyser.
  2. Dans le fichier TraitementReponses.gs, passez la constante MODE\_DEBUG\_ACTIF à true.
  3. Relancez le traitement (via le formulaire ou l'interface de retraitement).
  4. Consultez les journaux détaillés dans le menu **Exécutions** de l'éditeur de script.
  5. **Important** : N'oubliez pas de repasser la constante à false une fois le débogage terminé pour ne pas surcharger les logs en production.
* **FAQ des Erreurs Courantes**
  + **Problème : Une pièce jointe n'est pas envoyée.**
    - **Cause la plus probable** : Un caractère invisible (saut de ligne, espace) dans la cellule de l'ID du document dans sys\_Composition\_Emails.
    - **Solution** : Le code a été renforcé avec .trim(), mais il est bon de vérifier la propreté des données. Vérifiez aussi que le fichier sur Drive est bien partageable.
  + **Problème : L'e-mail de copie pour le formateur/patron n'est pas reçu.**
    - **Cause la plus probable** : La clé PREFIXE\_COPIE\_EMAIL est manquante dans l'onglet traductions de la BDD pour la langue concernée.
    - **Solution** : Ajouter la clé et sa traduction pour toutes les langues.
  + **Problème : Les scores sont à zéro ou le profil est incorrect.**
    - **Cause la plus probable** : Incohérence entre le libellé de la réponse dans le formulaire et le libelle correspondant dans le Paramètres (JSON) de la question dans la BDD. Les textes doivent correspondre **exactement**, y compris la casse et les espaces.
    - **Solution** : Copier-coller la valeur de l'option du formulaire dans le JSON pour garantir une correspondance parfaite.
  + **Problème : L'interface de retraitement ne se lance pas ou affiche une erreur.**
    - **Cause la plus probable** : Les scripts du Kit sont obsolètes et ne correspondent plus à la version attendue par le [TEMPLATE].
    - **Solution** : Mettre à jour les fichiers Menu.js, RetraitementUI.html et DialogueLigne.html du Kit avec les dernières versions du [TEMPLATE].