# Rapport de Problème : AC : echec lors de l'export d'un produit de odoo11 vers woo-Commerce

Date d'enregistrement : 2025-07-02 17:14:32

Statut du Problème : OUVERT

Assigné à (ID) : 1

## Description Détaillée du Problème

j'ai développé un module qui permet la connexion entre WooCommerce et odoo maintenant j'ai écris le code qui va me permettre d'exporter un produit de Odoo vers WooCommerce en utilisant "\_logger.exception(....)" pour voir l'erreur proprement dit dans les logs ensuite j'ai eu des erreurs ou chatgpt m'a demandé de Remplacer \_logger.exception(...) par \_logger.error(..., exc\_info=True) mais jusque là il y'a toujours probleme avec la methode

## Cause Racine Identifiée

\*\*  
  
La cause racine est une gestion des erreurs API incomplète ou inefficace dans le module Odoo, où le code ne parvient pas à intercepter, analyser et logger de manière claire les codes de statut HTTP et les messages d'erreur détaillés renvoyés par l'API WooCommerce. Cela masque la véritable nature de l'échec (ex: problème d'authentification, données invalides, ressource manquante) et rend le diagnostic impossible via les logs d'exceptions Python standard.  
  
\*\*Solutions Suggérées :\*\*  
  
\* \*\*Implémenter une gestion robuste des erreurs API HTTP :\*\*  
 \* Encapsulez toutes les requêtes HTTP vers WooCommerce dans des blocs `try-except` pour capturer les erreurs réseau (ex: `requests.exceptions.ConnectionError`, `Timeout`).  
 \* \*\*Crucial :\*\* Après \*chaque\* appel API, vérifiez systématiquement le code de statut HTTP de la réponse. Si le statut n'est pas dans la plage 2xx (succès), levez une exception personnalisée (ex: `WooCommerceAPIError`) incluant le code de statut et le corps complet de la réponse JSON/texte.  
 \* Analysez le corps de la réponse en cas d'erreur pour extraire et logger les messages d'erreur spécifiques de WooCommerce (souvent sous la clé `message` ou `data.details`).  
  
\* \*\*Améliorer la verbosité et la clarté des logs :\*\*  
 \* Lorsque vous loggez une erreur, incluez non seulement le traceback (`exc\_info=True`) mais aussi toutes les informations contextuelles pertinentes : l'URL de la requête, les en-têtes (sans les clés secrètes), le payload envoyé, et surtout, la réponse complète reçue de l'API WooCommerce (code de statut, en-têtes, corps).  
 \* Utilisez différents niveaux de log : `\_logger.debug` pour les requêtes/réponses réussies en mode développement, `\_logger.error` pour les échecs avec détails complets.  
 \* Assurez-vous que le niveau de log de votre serveur Odoo est configuré pour afficher les messages `debug` ou `info` lors du débogage (`--log-level=debug` ou dans le fichier de conf).  
  
\* \*\*Valider les données avant l'envoi :\*\*  
 \* Avant d'envoyer le produit à WooCommerce, effectuez une validation côté Odoo pour s'assurer que toutes les données requises par l'API WooCommerce sont présentes et dans le bon format (ex: `name`, `type`, `regular\_price`, `sku`). Cela peut éviter des erreurs "Bad Request" (400) de l'API.  
  
\* \*\*Tester l'API WooCommerce indépendamment :\*\*  
 \* Utilisez un outil comme Postman, Insomnia ou `curl` pour envoyer manuellement des requêtes POST de création de produit à votre API WooCommerce avec les mêmes clés API et données que votre module Odoo. Cela vous aidera à isoler si le problème vient de votre code Odoo, de la configuration de WooCommerce, ou des clés API elles-mêmes.  
 \* Vérifiez la documentation officielle de l'API REST WooCommerce pour les spécificités des endpoints, des corps de requête et des formats d'erreur.  
  
\* \*\*Utiliser un débogueur Python :\*\*  
 \* Activez le mode débogage dans Odoo (si vous utilisez PyCharm ou un IDE similaire) ou insérez des points d'arrêt avec `pdb` (`import pdb; pdb.set\_trace()`) dans votre code juste avant et après l'appel à l'API WooCommerce. Cela vous permettra d'inspecter les variables, les requêtes construites et les réponses brutes en temps réel.  
  
\* \*\*Vérifier la configuration de la connexion :\*\*  
 \* Confirmez que les clés API WooCommerce (Consumer Key et Consumer Secret) sont correctes et ont les permissions nécessaires (Lecture/Écriture).  
 \* Vérifiez que l'URL de base de votre site WooCommerce est correcte et accessible depuis le serveur Odoo (pas de problème de DNS, pare-feu, ou SSL).

## Solutions Suggérées/Mises en Œuvre

\*\*  
  
\* \*\*Implémenter une gestion robuste des erreurs API HTTP :\*\*  
 \* Encapsulez toutes les requêtes HTTP vers WooCommerce dans des blocs `try-except` pour capturer les erreurs réseau (ex: `requests.exceptions.ConnectionError`, `Timeout`).  
 \* \*\*Crucial :\*\* Après \*chaque\* appel API, vérifiez systématiquement le code de statut HTTP de la réponse. Si le statut n'est pas dans la plage 2xx (succès), levez une exception personnalisée (ex: `WooCommerceAPIError`) incluant le code de statut et le corps complet de la réponse JSON/texte.  
 \* Analysez le corps de la réponse en cas d'erreur pour extraire et logger les messages d'erreur spécifiques de WooCommerce (souvent sous la clé `message` ou `data.details`).  
  
\* \*\*Améliorer la verbosité et la clarté des logs :\*\*  
 \* Lorsque vous loggez une erreur, incluez non seulement le traceback (`exc\_info=True`) mais aussi toutes les informations contextuelles pertinentes : l'URL de la requête, les en-têtes (sans les clés secrètes), le payload envoyé, et surtout, la réponse complète reçue de l'API WooCommerce (code de statut, en-têtes, corps).  
 \* Utilisez différents niveaux de log : `\_logger.debug` pour les requêtes/réponses réussies en mode développement, `\_logger.error` pour les échecs avec détails complets.  
 \* Assurez-vous que le niveau de log de votre serveur Odoo est configuré pour afficher les messages `debug` ou `info` lors du débogage (`--log-level=debug` ou dans le fichier de conf).  
  
\* \*\*Valider les données avant l'envoi :\*\*  
 \* Avant d'envoyer le produit à WooCommerce, effectuez une validation côté Odoo pour s'assurer que toutes les données requises par l'API WooCommerce sont présentes et dans le bon format (ex: `name`, `type`, `regular\_price`, `sku`). Cela peut éviter des erreurs "Bad Request" (400) de l'API.  
  
\* \*\*Tester l'API WooCommerce indépendamment :\*\*  
 \* Utilisez un outil comme Postman, Insomnia ou `curl` pour envoyer manuellement des requêtes POST de création de produit à votre API WooCommerce avec les mêmes clés API et données que votre module Odoo. Cela vous aidera à isoler si le problème vient de votre code Odoo, de la configuration de WooCommerce, ou des clés API elles-mêmes.  
 \* Vérifiez la documentation officielle de l'API REST WooCommerce pour les spécificités des endpoints, des corps de requête et des formats d'erreur.  
  
\* \*\*Utiliser un débogueur Python :\*\*  
 \* Activez le mode débogage dans Odoo (si vous utilisez PyCharm ou un IDE similaire) ou insérez des points d'arrêt avec `pdb` (`import pdb; pdb.set\_trace()`) dans votre code juste avant et après l'appel à l'API WooCommerce. Cela vous permettra d'inspecter les variables, les requêtes construites et les réponses brutes en temps réel.  
  
\* \*\*Vérifier la configuration de la connexion :\*\*  
 \* Confirmez que les clés API WooCommerce (Consumer Key et Consumer Secret) sont correctes et ont les permissions nécessaires (Lecture/Écriture).  
 \* Vérifiez que l'URL de base de votre site WooCommerce est correcte et accessible depuis le serveur Odoo (pas de problème de DNS, pare-feu, ou SSL).