Documentation du script PowerShell

Ce script a pour but de **collecter les informations système** d'un ordinateur Windows et de **les envoyer** via une requête **POST** à une **API Web**.

Lien de Téléchargement via GitHub

1. Fonctionnement général

Le script :

- Récupère différentes **informations système** locales (nom du PC, OS, RAM, CPU, adresse IP, etc.).
- Formate ces informations en **JSON**.
- Envoie ce JSON à une **API HTTP/HTTPS**.
- Gère la **vérification SSL** pour accepter tout certificat (auto-signé, etc.).
- Gère les erreurs lors de l'envoi de la requête.

2. Paramètres de configuration

Variable	Description	Exemple
`$ocsServer`	Adresse du serveur OCS Inventory	`http://10.29.126.31/`
`$apiEndpoint`	URL complète vers l'**endpoint** API recevant les données	`https://10.29.126.31/track-it/index.php/Pull`
`$apiKey`	**Clé d'API** pour authentifier la requête si nécessaire	`"CLE_API"` ou vide

3. Informations collectées

Champ JSON	Source (commande)	Description
`name`	`$env:COMPUTERNAME`	Nom de l'ordinateur
`os`	`Get-WmiObject Win32_OperatingSystem`	Nom du système d'exploitation
`os_version`	`Get-WmiObject Win32_OperatingSystem`	Version du système d'exploitation
`architecture`	`Get-WmiObject Win32_OperatingSystem`	Architecture (32-bit ou 64-bit)
`user`	`$env:USERNAME`	Nom de l'utilisateur connecté
`ram`	`Get-WmiObject Win32_ComputerSystem`	Quantité de RAM installée (en Go)
`cpu`	`Get-WmiObject Win32_Processor`	Modèle du processeur
`serial`	`Get-WmiObject Win32_BIOS`	Numéro de série du BIOS
`mac`	`Get-WmiObject Win32_NetworkAdapterConfiguration`	Adresse MAC principale
`ip`	`Get-WmiObject Win32_NetworkAdapterConfiguration`	Adresse IP principale
`domaine`	`Get-WmiObject Win32_ComputerSystem`	Domaine ou groupe de travail
`windows_key`	`Get-WmiObject SoftwareLicensingService`	Clé produit Windows (si existante)
`license_status`	`Get-WmiObject SoftwareLicensingProduct`	Statut de la licence Windows

4. Gestion de la sécurité SSL

Pour éviter les erreurs liées aux certificats non valides (auto-signés, expirés, etc.), la ligne suivante force l'acceptation de **tout certificat** :

```
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = { $true }
```

Cela désactive toute vérification de sécurité SSL.

5. Structure de la requête envoyée

- Méthode HTTP : POST
- Format du corps : JSON
- Headers :
    - Content-Type: application/json
    - Accept: application/json
    - Authorization: Bearer <API_KEY> (si $apiKey défini)

6. Gestion des erreurs

Le script utilise un bloc try/catch :

- Si l'envoi est réussi : message "Données envoyées avec succès !"
- Si une erreur survient : message "Erreur lors de l'envoi des données" avec les détails.

7. Remarques

- Le champ $ocsServer est défini mais **non utilisé** dans ce script (préparé pour de futures évolutions).

- Les informations peuvent être **sensibles** (clé Windows, adresse MAC, etc.)
  → vérifier la politique de confidentialité avant envoi.

- Le script est prévu pour **Windows** uniquement (utilisation des cmdlets Get-WmiObject).

- Penser à vérifier que **PowerShell** a les droits nécessaires pour envoyer une requête HTTP/HTTPS sur le réseau.

8. Améliorations possibles

- Utiliser Get-CimInstance à la place de Get-WmiObject (plus moderne).

- Ajouter des logs dans un fichier texte en local.

- Permettre la configuration par fichier (.json, .ps1) au lieu de variables statiques.

- Sécuriser davantage la gestion SSL (en vérifiant le certificat du serveur).

- Ajouter une validation avant l'envoi (ex : vérifier que l'adresse IP et la MAC sont bien valides).

Lancement du script

Aucun paramètre n'est nécessaire : **le script se lance directement** et collecte automatiquement toutes les informations avant de les envoyer.