Documentation technique – Empêcher la mise en veille d’un PC d’affichage

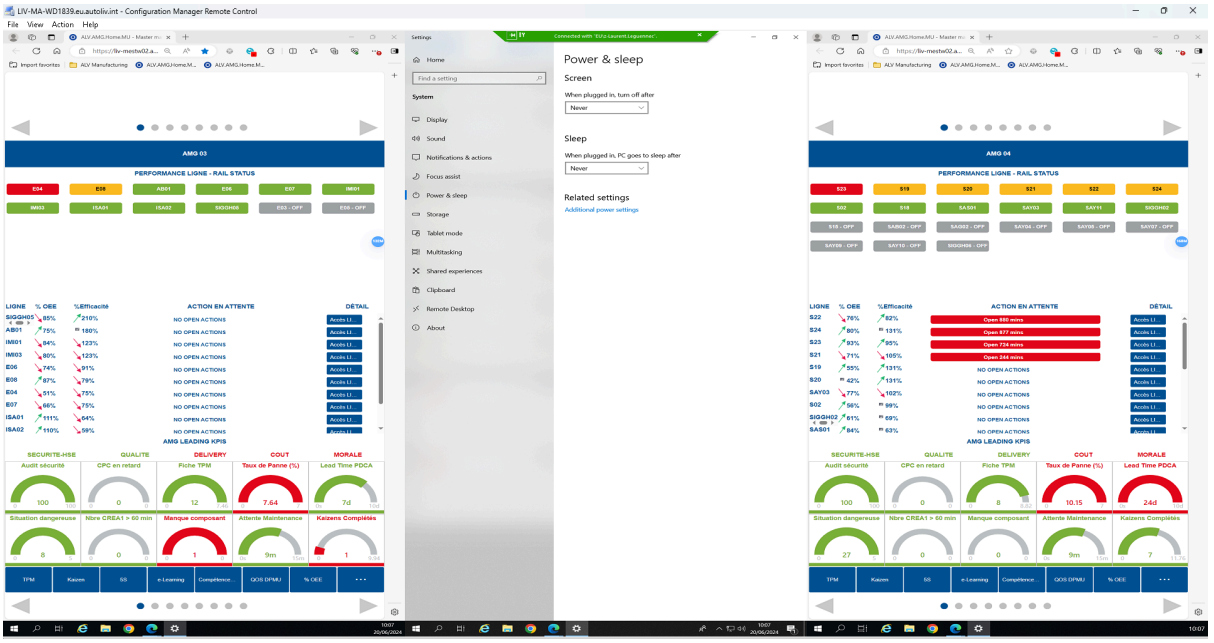
Contexte

Le poste **LIV-MA-WD1839**, utilisé pour de l'affichage, se met en veille automatiquement après **4h ou 8h**, malgré les paramètres système indiquant qu'il **ne doit jamais se mettre en veille**.

Étapes de diagnostic

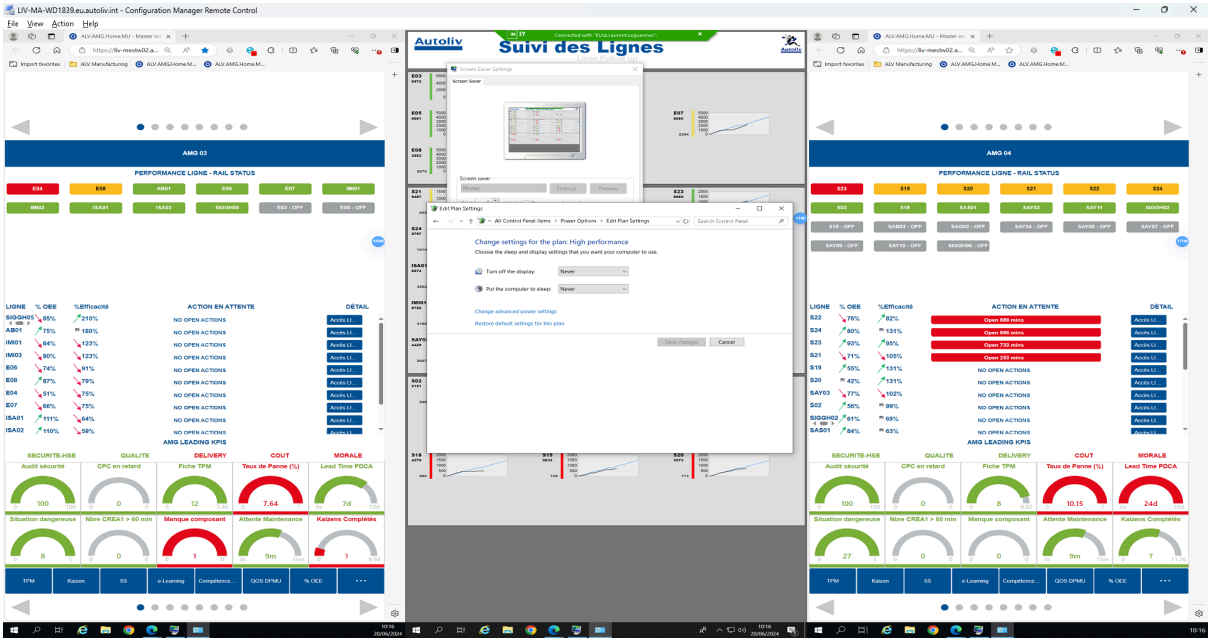
1. Vérification des paramètres système de mise en veille

- Dans **Paramètres > Système > Alimentation et mise en veille** :
  → La mise en veille est **désactivée** pour l'écran et l'ordinateur.



2. Vérification de l'écran de veille

- Dans **Panneau de configuration > Apparence > Écran de veille** :
  → Aucun écran de veille n'est activé.



3. Vérification des GPO (stratégies de groupe)

- Tentative via gpedit.msc pour rechercher des stratégies actives pouvant forcer la mise en veille.

- **Échec d'accès/modification des GPO** dû à un manque de droits administrateur.

Contournement mis en place : Script PowerShell

En l'absence de droits suffisants pour modifier les stratégies de groupe ou utiliser certains outils plus robustes, un **script PowerShell** a été mis en place pour **simuler une activité** sur le PC et empêcher la mise en veille.

🖱️ Version 1 : Simulation de mouvement de souris (toutes les 3h)

Fonctionnement :

Le script simule un **léger déplacement de souris** toutes les 3 heures.

```
# Définir la fonction MoveMouse sans utiliser de classe externe
Add-Type @"
using System;
using System.Runtime.InteropServices;

public class MouseMover {
    // Importation de la fonction user32.dll mouse_event pour simuler les événements de la souris
    [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention = CallingConvention.StdCall)]
    public static extern void mouse_event(long dwFlags, long dx, long dy, long cButtons, long dwExtraInfo);

    // Constante pour indiquer un mouvement de la souris
    private const int MOUSEEVENTF_MOVE = 0x0001;

    // Déplacer la souris
    public static void Move(int x, int y) {
        mouse_event(MOUSEEVENTF_MOVE, x, y, 0, 0);
    }
}
"@

# Fonction pour déplacer la souris de manière aléatoire
function Move-MouseRandomly {
    $x = Get-Random -Minimum -20 -Maximum 20
    $y = Get-Random -Minimum -20 -Maximum 20
    [MouseMover]::Move($x, $y)
}

# Boucle infinie pour déplacer la souris toutes les 3 heures
while ($true) {
    # Déplacer la souris
    Move-MouseRandomly
    # Attendre 3 heures avant de déplacer la souris à nouveau
    Start-Sleep -Seconds 10800  # 3 heures
}
```

❌ Inconvénient :

- Si un utilisateur utilise ponctuellement le PC, ces mouvements de souris peuvent être gênants.

⌨️ Version 2 : Appui simulé sur la touche Verr Num (toutes les 2h)

Fonctionnement :

Le script simule l'appui sur la touche **Num Lock** pendant 5 secondes toutes les 2 heures, sans impact visuel.

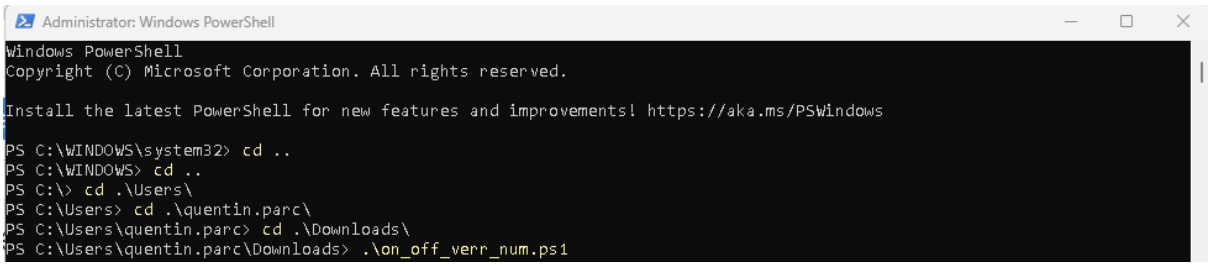
```
# Définition de la classe KeyboardHelper pour gérer les événements de clavier
Add-Type -TypeDefinition @"
using System;
using System.Runtime.InteropServices;

public class KeyboardHelper {
    [DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
    public static extern void keybd_event(byte bVk, byte bScan, uint dwFlags, IntPtr
dwExtraInfo);

    private const int KEYEVENTF_EXTENDEDKEY = 0x1;
    private const int KEYEVENTF_KEYUP = 0x2;
    private const int VK_NUMLOCK = 0x90;  // Touche Num Lock

    public static void PressNumLockKey() {
        keybd_event(VK_NUMLOCK, 0x45, KEYEVENTF_EXTENDEDKEY | 0, IntPtr.Zero);
# Press
        keybd_event(VK_NUMLOCK, 0x45, KEYEVENTF_EXTENDEDKEY |
KEYEVENTF_KEYUP, IntPtr.Zero); # Relâche
    }
}
"@

# Fonction pour simuler l'appui sur la touche Num Lock pendant 5 secondes
function Press-NumLockKeyRepeatedly {
    $endTime = (Get-Date).AddSeconds(5)
    while ((Get-Date) -lt $endTime) {
        [KeyboardHelper]::PressNumLockKey()
        Start-Sleep -Seconds 1  # Pause d'une seconde entre chaque appui
    }
}

# Boucle infinie pour simuler l'appui sur la touche Num Lock toutes les 2 heures
while ($true) {
    Press-NumLockKeyRepeatedly
    Start-Sleep -Seconds 7200  # Attente de 2 heures (2 * 3600 secondes)
}
```

✅ Avantage :

- Invisible pour l'utilisateur.
- Pas de gêne si une personne travaille ponctuellement sur le poste.

Exécution automatique

- Le script est lancé **manuellement** via **Windows PowerShell** (pas ISE).

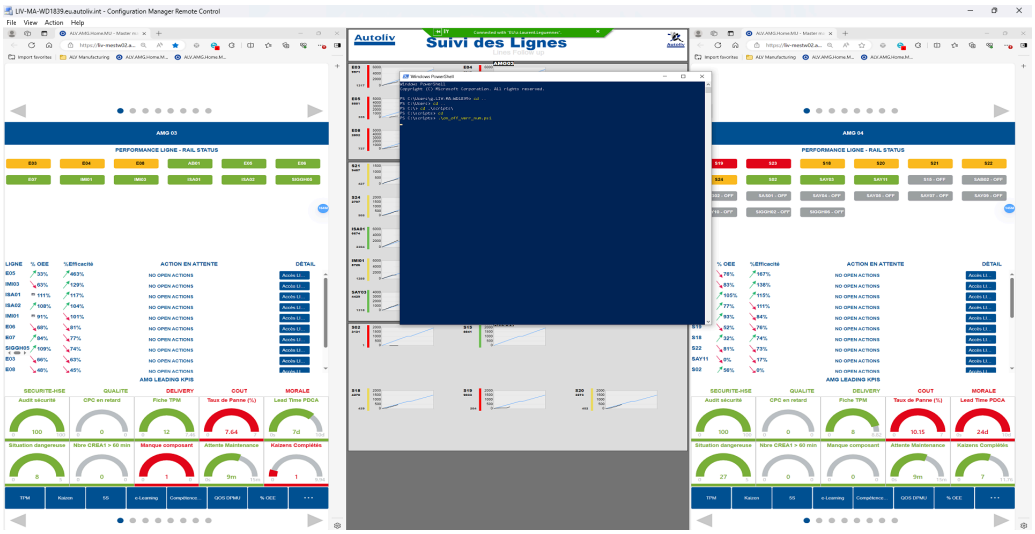


- Pour autoriser l'exécution de scripts :

```
Set-ExecutionPolicy Unrestricted
```

Puis valider par Yes to All.

- Pour arrêter : CTRL + C dans la console.



Améliorations possibles (non réalisées par manque de droits)

Avec des **droits administrateurs**, d'autres méthodes plus propres et durables auraient pu être utilisées :

🔧 **Modifier ou créer une GPO :**

- Désactiver la mise en veille via stratégie de groupe (gpedit.msc ou via GPO Active Directory).

🔧 **Utiliser powercfg :**

- Script avec powercfg /change standby-timeout-ac 0 pour désactiver totalement la mise en veille via ligne de commande.

🔧 **Déployer une tâche planifiée :**

- Planifier automatiquement le lancement du script au démarrage du PC.

🧰 **Utiliser des outils tiers :**

- Ex. : **Caffeine**, **Mouse Jiggler**, ou autres petits utilitaires conçus pour empêcher la mise en veille sans perturber l'utilisateur.

Conclusion

Le problème de mise en veille a été contourné efficacement via un **script PowerShell**, avec une méthode **discrète et non intrusive** (Num Lock).
La solution est **fonctionnelle et stable**, mais pourrait être améliorée si les droits administrateurs étaient disponibles.