BTS SIO — Option SISR

Documentation Technique

Migration RMM NinjaOne

Instance Inersio → Instance Convergence France

Auteur	Evan Pellegrino
Entreprise	Asap Network
Migration	Inersio RMM → NinjaOne Convergence France
Technologie	NinjaOne RMM + PowerShell
Clients migrés	10 organisations — migration en cours
Formation	BTS Services Informatiques aux Organisations — Option SISR
Épreuve	E5 — Situation professionnelle individuelle

Version 1.0

Sommaire

1. Introduction

Cette documentation décrit la migration du parc informatique client depuis l'instance NinjaOne RMM héritée de la société Inersio (rachetée par Asap Network) vers la nouvelle instance NinjaOne gérée par Convergence France, groupe opérateur auquel appartient Asap Network.

Elle détaille le contexte du projet, les problématiques techniques rencontrées, la solution d'automatisation mise en place via PowerShell, ainsi que le protocole de déploiement utilisé dans la console NinjaOne.

2. Contexte et objectifs

2.1 Origine du projet

Suite au rachat de la société Inersio par Asap Network, le parc de machines clients sous contrat de maintenance a été récupéré, ainsi que la gestion de celles-ci via NinjaOne RMM sur l'instance Inersio. Cette instance était utilisée pour la supervision des postes et la gestion de tickets de niveau 1.

En parallèle, le groupe Convergence France — dont fait partie Asap Network — a ouvert un nouveau service de ticketing de niveau 1 centralisé pour l'ensemble des partenaires du groupe. La direction a donc décidé de migrer tous les postes clients vers cette nouvelle instance afin de déléguer le support de niveau 1 à Convergence, et de permettre aux techniciens Asap Network de se concentrer sur le niveau 2.

2.2 Objectifs de la migration

Objectif	Description
Décharger le N1	Déléguer le support de niveau 1 à Convergence France via la nouvelle instance NinjaOne
Unifier le parc	Regrouper toutes les machines clients dans une instance commune au groupe
Automatiser	Déployer la migration sans intervention manuelle poste par poste
Transparence client	Réaliser la migration de façon totalement invisible pour l'utilisateur final

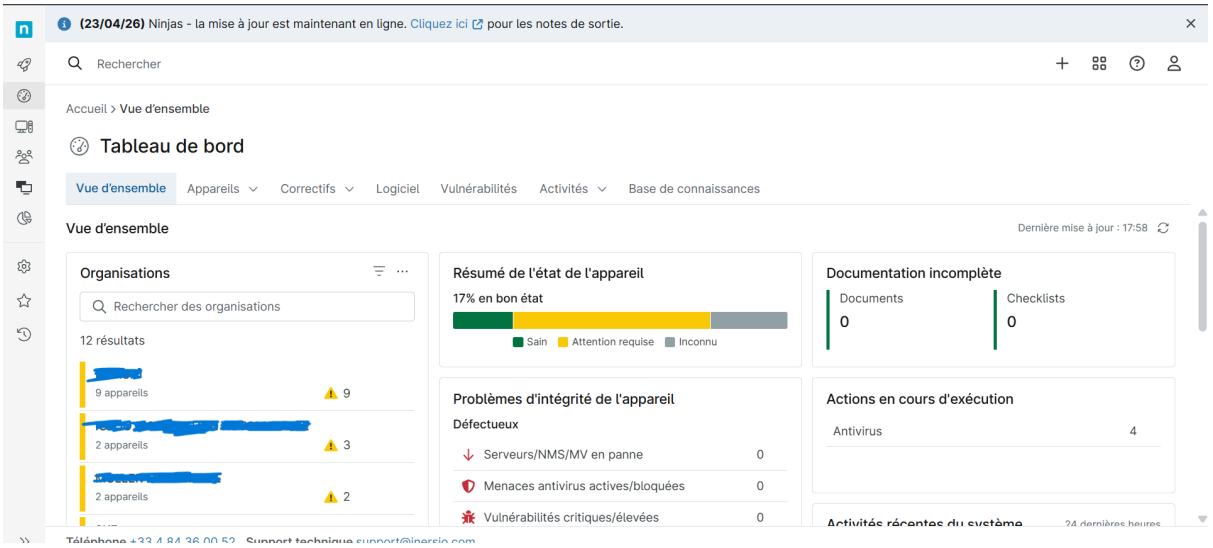
3. Présentation de NinjaOne RMM

NinjaOne est une solution de Remote Monitoring and Management (RMM) utilisée par les prestataires IT pour superviser et administrer à distance des postes et serveurs clients. Elle permet notamment la supervision en temps réel, l'exécution de scripts à distance, la gestion des mises à jour, et la génération d'alertes automatiques.

Fonctionnalité	Utilisation chez Asap Network
Supervision des appareils	Monitoring en temps réel de l'état des postes et serveurs clients
Exécution de scripts	Déploiement à distance de scripts PowerShell (dont le script de migration)
Tâches planifiées	Automatisation d'actions récurrentes (redémarrages, migrations, etc.)
Gestion des organisations	Séparation des parcs clients par organisation dans la console
Intégration ticketing	Remontée d'alertes vers JitBit pour créer automatiquement des tickets

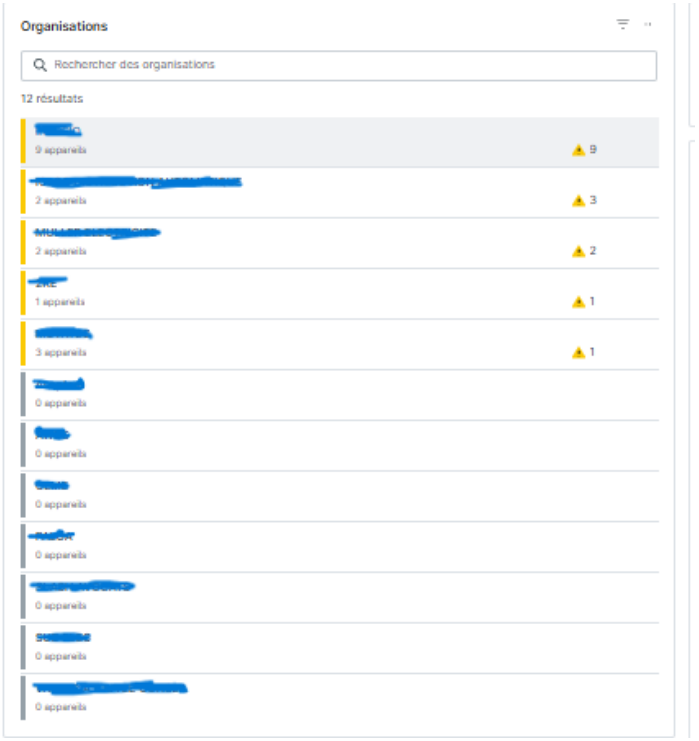
3.1 Tableau de bord — Vue d'ensemble

La capture ci-dessous présente le tableau de bord de la nouvelle instance NinjaOne Convergence. On y voit les 12 organisations migrées ou en cours de migration, avec l'état de santé global des appareils (17% en bon état au moment de la capture, le reste étant en phase de migration/supervision).



3.2 Liste des organisations

La liste des organisations dans NinjaOne correspond aux différents clients sous contrat de maintenance. Chaque organisation regroupe les appareils (postes et serveurs) du client. Les organisations affichant des alertes (icône ⚠️) ont encore des appareils nécessitant attention ou migration.



4. Problématiques techniques rencontrées

4.1 Impossibilité de désinstallation standard

La première idée envisagée était simple : détruire l'ancienne instance, ce qui libèrerait les agents, puis réinstaller le nouvel agent sur chaque poste. En réalité, NinjaOne ne fonctionne pas ainsi. Son agent est conçu pour être résilient et ne peut pas être désinstallé via les méthodes Windows classiques.

Méthode tentée	Résultat
Désinstallation via Panneau de configuration Windows	Désinstallation incomplète ou nulle — l'agent continue de tourner en arrière-plan
Suppression manuelle des fichiers	L'agent se réinstalle automatiquement au redémarrage
Désinstallation via uninstall.exe sans les bons paramètres	Fenêtre graphique visible, interaction utilisateur requise — non compatible déploiement silencieux
Appel au support NinjaOne	Solution validée : utiliser uninstall.exe --mode unattended depuis le dossier du service

4.2 Problème d'exécution du script via NinjaOne

Lors des premiers tests, une seconde problématique est apparue : le script s'interrompait après la désinstallation de l'ancien agent, sans jamais installer le nouveau. La cause a été identifiée : NinjaOne injecte et exécute le script en utilisant son propre agent comme vecteur. Une fois

l'ancien agent désinstallé en cours d'exécution, NinjaOne perdait la main sur la machine et le script s'arrêtait.

> **Solution retenue**
> Le script est désormais injecté dans la machine via une tâche planifiée NinjaOne configurée pour s'exécuter « dès que possible au prochain démarrage ». Ainsi, NinjaOne dépose le script puis c'est Windows lui-même qui l'exécute au démarrage, sans dépendre de la continuité de l'agent NinjaOne.

5. Solution technique : le script PowerShell

5.1 Architecture du script

Le script PowerShell développé pour automatiser la migration se déroule en 5 étapes séquentielles :

Étape	Action	Détail technique
1 — Configuration	Définition des variables	URL du nouvel agent MSI (spécifique par client) + chemin temporaire de téléchargement
2 — Téléchargement	Récupération du MSI	Via BITS (Background Intelligent Transfer Service) pour ne pas saturer la bande passante
3 — Désinstallation	Suppression de l'ancien agent	Recherche du service Ninja*Agent* → localisation de uninstall.exe → exécution silencieuse (--mode unattended)
4 — Installation	Déploiement du nouvel agent	msiexec.exe /i /qn /norestart — mode totalement silencieux, sans redémarrage forcé
5 — Attente	Pause d'enregistrement	60 secondes pour laisser l'agent s'enregistrer sur les serveurs NinjaOne cloud

5.2 Script complet commenté

Le script ci-dessous est la version complète avec annotations techniques. Il est adapté pour chaque client en modifiant uniquement l'URL du MSI ($newAgentUrl), qui contient un identifiant unique (GUID) propre à chaque organisation NinjaOne.

```
# ================================================================
#  Script : Migration Agent NinjaOne
#  Auteur  : Evan Pellegrino — Asap Network
#  Objectif: Désinstaller l'ancien agent NinjaOne (instance Inersio)
#            puis installer le nouvel agent (instance Convergence).
```

```
#
#  Méthode : Le script est injecté via une tâche planifiée NinjaOne
#            afin de s'exécuter de façon invisible au prochain
#            démarrage de la machine.
# ================================================================

# === SECTION 1 : Configuration ====================================
#  URL de téléchargement du nouvel agent MSI (spécifique à chaque
#  client — l'URL contient un GUID unique lié à l'organisation).
$newAgentUrl =
"https://eu.ninjarmm.com/agent/installer/e57f4778-4e06-484a-abd3-057db1e2063b/11.0.5792
/NinjaOne-Agent-ASAPIGLOODISTRIBUTIONAUTOMATIQUEPR-Bureauprincipal-Auto.msi"

#  Chemin temporaire où le MSI sera téléchargé avant installation.
$tempPath    = "$env:TEMP\ninja-new.msi"

# === SECTION 2 : Téléchargement du nouvel agent ==================
#  BITS (Background Intelligent Transfer Service) est utilisé plutôt
#  que Invoke-WebRequest pour éviter de saturer la bande passante :
#  BITS transfère en arrière-plan en utilisant la bande passante libre.
Write-Output "Téléchargement du nouvel agent depuis $newAgentUrl..."
Start-BitsTransfer -Source $newAgentUrl -Destination $tempPath

# === SECTION 3 : Désinstallation de l'ancien agent ===============
#  Problème identifié : NinjaOne ne se désinstalle pas via le
#  panneau de configuration Windows (il se relance automatiquement).
#  La seule méthode fiable est d'utiliser son propre uninstall.exe,
#  localisé dans le dossier d'installation via le chemin du service.
Write-Output "Recherche de l'ancien agent Ninja à désinstaller (via le service)..."

#  Recherche des services Windows dont le nom ou l'affichage contient
#  'Ninja*Agent*' — méthode robuste car indépendante du chemin d'install.
$services = Get-WmiObject Win32_Service | Where-Object {
    $_.Name -like "Ninja*Agent*" -or $_.DisplayName -like "*Ninja*"
}

if ($services) {
    foreach ($svc in $services) {
        Write-Output "Service trouvé : $($svc.Name)"

        # Extraction du chemin de l'exécutable depuis le PathName du service.
        # Le PathName peut contenir des guillemets et des paramètres supplémentaires.
        $exePath  = $svc.PathName.Trim()
        $exePath  = $exePath.Trim('"')
        $exePath  = $exePath -replace '^(.+?\.exe).*$', '$1'

        # Remontée vers le dossier parent pour trouver uninstall.exe.
        $folder   = Split-Path $exePath
        $uninst   = Join-Path $folder 'uninstall.exe'

        if (Test-Path $uninst) {
            # Lancement silencieux via --mode unattended :
            # aucune fenêtre, aucune interaction utilisateur requise.
            Start-Process $uninst -ArgumentList "--mode unattended" -Wait
            Write-Output "Désinstallation terminée pour le service $($svc.Name)."
        }
```

```
        else {
            Write-Output "uninstall.exe introuvable dans $folder, aucune
désinstallation effectuée."
        }

        # Pause de sécurité : on laisse le temps aux processus Ninja de
        # se terminer proprement avant de lancer l'installation suivante.
        Start-Sleep -Seconds 10
    }
}
else {
    Write-Output "Aucun service Ninja*Agent* trouvé. Aucun ancien agent à
désinstaller."
}

# === SECTION 4 : Installation du nouvel agent ====================
#  Installation silencieuse via msiexec :
#    /i  = installer le package MSI
#    /qn = mode silencieux total (no UI)
#    /norestart = pas de redémarrage automatique forcé
Write-Output "Installation du nouvel agent..."
$installArgs = "/i `"$tempPath`" /qn /norestart"
Start-Process msiexec.exe -ArgumentList $installArgs -Wait

# === SECTION 5 : Attente d'enregistrement ========================
#  L'agent NinjaOne a besoin de quelques secondes après installation
#  pour s'enregistrer auprès des serveurs cloud et apparaître dans
#  la console d'administration de la nouvelle instance.
Start-Sleep -Seconds 60
Write-Output "Installation terminée. Vérifiez la présence dans le nouveau portail
Ninja."
```

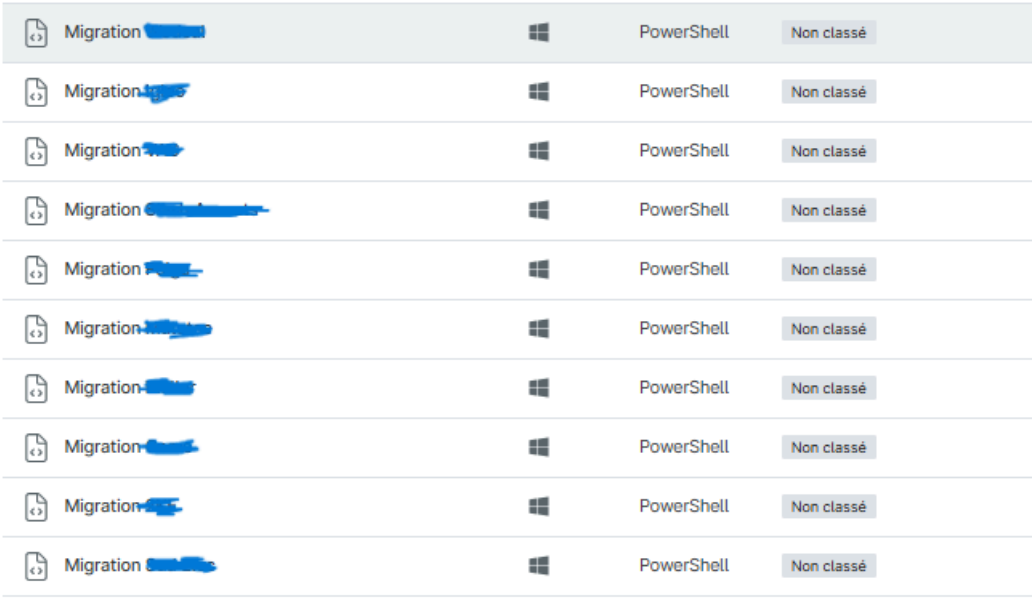
5.3 Adaptation par client

Le script est identique pour tous les clients à une exception près : l'URL de téléchargement du MSI ($newAgentUrl) est unique par organisation NinjaOne. Cette URL est générée depuis la console d'administration NinjaOne lors de la création d'un nouvel installeur pour l'organisation concernée. Un script distinct est donc créé pour chaque client dans la bibliothèque NinjaOne.

6. Protocole de déploiement dans NinjaOne

6.1 Étape 1 — Création du script dans la bibliothèque

Le script PowerShell est d'abord créé dans la bibliothèque de scripts de la console d'administration NinjaOne (Administration → Bibliothèque). Un script distinct est créé pour chaque client, nommé de façon explicite (ex. : « Migration Igloo », « Migration Palga »). La capture ci-dessous montre la liste des scripts de migration disponibles dans la bibliothèque :

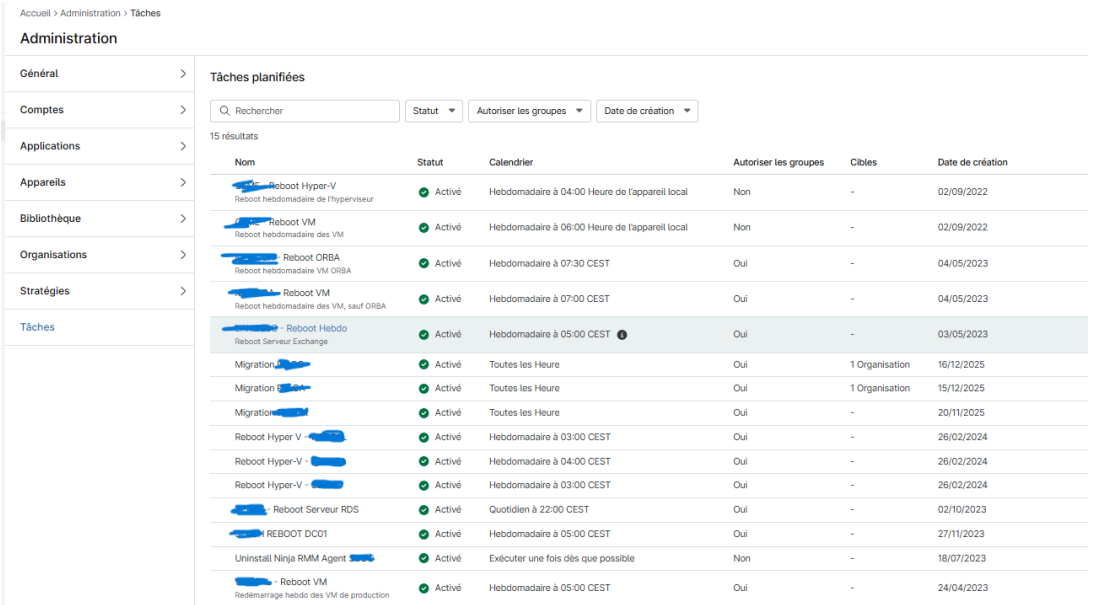


6.2 Étape 2 — Création de la tâche planifiée

Une tâche planifiée est ensuite créée dans NinjaOne (Administration → Tâches) pour chaque client à migrer. La tâche est configurée avec les paramètres suivants :

- Déclencheur : « Exécuter une fois dès que possible » — la tâche se lance au prochain démarrage de chaque machine ciblée
- Visibilité : invisible pour l'utilisateur — aucune fenêtre, aucune notification
- Cible : l'organisation ou le groupe de machines concerné
- Script : le script PowerShell correspondant au client

La capture ci-dessous montre la liste complète des tâches planifiées, avec les migrations en cours (statut « Toutes les heures ») et les redémarrages programmés pour d'autres clients :



6.3 Déroulement côté machine client

Étape	Ce qui se passe sur le poste
1 — Dépôt	NinjaOne injecte silencieusement le script PowerShell via l'ancienne instance
2 — Démarrage	Au prochain allumage du poste, Windows exécute la tâche planifiée
3 — Téléchargement	Le MSI du nouvel agent est téléchargé via BITS (en arrière-plan)
4 — Désinstallation	L'ancien agent est désinstallé proprement via son propre uninstall.exe
5 — Installation	Le nouvel agent MSI est installé silencieusement
6 — Enregistrement	Après 60 secondes, l'appareil apparaît dans la nouvelle instance Convergence

7. État d'avancement du projet

La migration est en cours au moment de la rédaction de cette documentation. Plusieurs organisations ont été migrées avec succès, d'autres sont encore en attente pour les raisons suivantes :

- Machines non allumées depuis le déploiement de la tâche planifiée — la migration s'effectuera au prochain démarrage
- Clients n'ayant pas encore validé la migration de leur côté

Organisation	Type de machines	Statut
Igloo Distribution Automatique	Postes + Serveurs	En cours (2 appareils — 3 alertes)
Muller Electricité	Postes	En cours (2 appareils — 2 alertes)
2KE	Postes	En cours (1 appareil — 1 alerte)
Mutatec	Postes + Serveurs	En cours (3 appareils — 1 alerte)
Asaptest	Environnement de test	Migré (0 appareils actifs)
AWBS	Postes	Migré
CEME	Postes + Hyper-V	Migré
PALGA	Postes	Migré

Organisation	Type de machines	Statut
Slash Avocats	Postes + RDS	Migré
Sud Elec	Postes	Migré
Valeur Ajoutée Conseil	Postes	Migré
Inersio	Ancienne instance	En attente (9 appareils restants)

8. Conclusion

Ce projet de migration NinjaOne illustre la capacité à analyser un problème technique complexe, à identifier ses contraintes spécifiques (désinstallation non standard d'un agent RMM, interruption du script liée à l'agent lui-même), et à concevoir une solution d'automatisation robuste et transparente pour le client final.

La combinaison NinjaOne + PowerShell + tâches planifiées a permis de traiter la majorité du parc sans aucune intervention manuelle poste par poste, en garantissant une expérience utilisateur non perturbée (exécution invisible, sans redémarrage forcé).

Ce projet mobilise des compétences directement issues du référentiel BTS SIO SISR : administration de systèmes Windows, scripting PowerShell, utilisation d'un outil RMM professionnel, et gestion de projet d'infrastructure en environnement de production multi-clients.