EPSI
Administrateur Systèmes, Réseaux et Bases de Données
Bloc E6.1 — Concevoir et tester des solutions applicatives

NTL-SysToolbox

Dossier Technique et Fonctionnel

v1.0 — Avril 2026

PROJET

Client	NordTransit Logistics (NTL)
Projet	NTL-SysToolbox v1.0
Formation	ASRBD — EPSI
Bloc	E6.1
Langage	Python 3
Passage	Avril 2026
Repo Git	github.com/Enzo1401/MSPR_6.1

ÉQUIPE PROJET

Enzo Méresse
Damien Fernandes
Ioan Domingues
Côme Barluet de Beauchesne

Document confidentiel — Usage interne DSI NordTransit Logistics

Sommaire

1. Introduction 3
1.1 Contexte 3
1.2 Objet du document 3
1.3 Périmètre 3

2. Choix Technologiques 4
2.1 Langage : Python 3 4
2.2 Dépendances Python 4
2.3 Format de configuration : JSON 5
2.4 Source de référence EOL : endoflife.date 5

3. Architecture Logique 5
3.1 Structure des fichiers 5
3.2 Flux d'exécution général 6
3.3 Indépendance des modules 6

4. Description Fonctionnelle des Modules 6
4.1 Module Diagnostic 6
4.2 Module Sauvegarde WMS 8
4.3 Module Audit d'Obsolescence 9

5. Gestion de la Configuration et des Secrets 10

6. Ergonomie du Menu Interactif 11

7. Compatibilité Multiplateforme 12

8. Compromis et Limites Assumés 13

9. Workflow Git et Qualité de Code 13

10. Executive Summary (English) 14

Annexe Tableau des Machines Virtuelles Ciblées 15

1. Introduction

1.1 Contexte

NordTransit Logistics (NTL) est une PME de logistique implantée dans les Hauts-de-France. Elle opère un siège social à Lille et trois entrepôts à Lens (WH1), Valenciennes (WH2) et Arras (WH3), avec environ 240 salariés en régime de croisière.

L'infrastructure repose sur une vingtaine de machines virtuelles hébergées au siège, un Active Directory centralisé (DC01/DC02), et une application cœur de métier — le Warehouse Management System (WMS) — tournant sur Ubuntu 20.04 avec une base MySQL. L'indisponibilité de ce WMS stoppe immédiatement les opérations de réception et d'expédition sur tous les sites.

Les équipes IT constatent plusieurs fragilités : absence de tests de restauration des sauvegardes, supervision limitée à la présence des hôtes, authentification multifacteur non généralisée, et gouvernance insuffisante des espaces Microsoft 365.

1.2 Objet du document

Ce Dossier Technique et Fonctionnel (DTF) décrit l'architecture, les choix technologiques, le fonctionnement interne et les compromis retenus pour NTL-SysToolbox, un outil en ligne de commande développé en Python pour industrialiser les vérifications d'exploitation de NTL.

Il s'adresse à la DSI de NTL et à tout technicien chargé du déploiement, de la maintenance ou de l'évolution de l'outil.

1.3 Périmètre

NTL-SysToolbox couvre trois domaines fonctionnels indépendants :

- Module Diagnostic : vérification de l'état des services AD/DNS, MySQL, et des ressources système complètes (OS, uptime, CPU, RAM, disques) sur Windows et Linux.
- Module Sauvegarde WMS : export logique de la base MySQL au format SQL (dump complet) et export de table au format CSV.
- Module Audit d'obsolescence : inventaire réseau par ping, interrogation WMI des OS Windows, qualification EOL via l'API endoflife.date, et génération automatique d'un rapport JSON et TXT.

2. Choix Technologiques

2.1 Langage : Python 3

Python a été retenu comme langage principal pour les raisons suivantes :

- Multiplateforme natif : exécutable sous Windows et Linux sans recompilation, conformément à l'exigence du cahier des charges.
- Écosystème riche : les bibliothèques nécessaires sont disponibles via PyPI et couvrent l'ensemble des besoins sans développement bas niveau.
- Lisibilité et maintenabilité : la syntaxe de Python facilite la compréhension du code par une petite équipe IT.
- Délai : Python permet un prototypage rapide, adapté aux 19 heures de préparation allouées.

Les versions 3.11 et 3.12 sont recommandées pour une compatibilité optimale. Les versions ultérieures (3.13, 3.14+) sont également utilisables mais peuvent nécessiter une installation manuelle de mysql-connector-python==8.0.33 en cas d'incompatibilité.

2.2 Dépendances Python

Bibliothèque	Usage	Justification
paramiko	SSH vers serveurs Ubuntu	Bibliothèque SSH de référence en Python
pywinrm	WinRM / PowerShell vers Windows Server	Seule lib Python permettant d'exécuter PowerShell à distance via WinRM/NTLM
mysql-connector-python	Connexion MySQL (diagnostic + CSV)	Driver officiel Oracle, version 8.0.33 fixée pour compatibilité maximale
wmi	Interrogation WMI Windows	Windows-only — import conditionnel via platform.system()
pywin32	Dépendance de wmi	Windows-only — marqueur sys_platform dans requirements.txt
requests	Appels HTTP à l'API endoflife.date	Standard de facto pour les requêtes HTTP en Python
colorama	Colorisation terminal Windows/Linux	Résout la compatibilité des couleurs ANSI sur Windows
python-dotenv	Chargement variables d'environnement	Permet de surcharger la config via variables d'env

2.3 Format de configuration : JSON

La configuration est centralisée dans config/config.json. Ce format a été choisi pour sa lisibilité native, sa prise en charge directe par Python (module json standard), et sa facilité de surcharge par variables d'environnement via python-dotenv. Les paramètres sensibles (mots de passe) ne sont jamais stockés dans ce fichier.

2.4 Source de référence EOL : endoflife.date

Le module d'audit s'appuie sur l'API publique endoflife.date (https://endoflife.date/api/). Elle recense les cycles de vie de plus de 300 produits, propose une API REST sans authentification, et est maintenue par la communauté open source sur la base des annonces officielles des éditeurs.

Limite assumée : les données dépendent de la disponibilité du service externe. La date de validité des données correspond à la date d'exécution de la requête. Pour les cycles critiques, une vérification manuelle sur les sites officiels reste recommandée.

3. Architecture Logique

3.1 Structure des fichiers

Fichier / Dossier	Rôle
main.py	Point d'entrée — menu CLI interactif, gestion de session, routage vers les modules
modules/diagnostic.py	Module 1 — AD/DNS, MySQL, ressources Windows (WinRM) et Ubuntu (SSH)
modules/backup_wms.py	Module 2 — Sauvegarde SQL (mysqldump) et export CSV de table MySQL
modules/audit.py	Module 3 — Scan réseau, WMI, API EOL, classification et rapport JSON/TXT
modules/utils.py	Utilitaires transverses — chargement config, sortie structurée, timer session
config/config.json	Fichier de configuration (IP, utilisateurs, nom BDD) — sans secrets
backups/	Sauvegardes SQL et exports CSV, créé automatiquement
docs/	Rapports d'audit EOL horodatés (JSON + TXT), créé automatiquement
lancer_toolbox.bat	Script d'amorçage Windows — venv, dépendances, lancement
lancer_toolbox.sh	Script d'amorçage Linux — équivalent du .bat pour Ubuntu
requirements.txt	Liste figée des dépendances avec marqueurs sys_platform pour wmi/pywin32

3.2 Flux d'exécution général

Au lancement, main.py charge la configuration via utils.load_config(), puis entre dans une boucle principale affichant le menu à trois entrées. Chaque choix redirige vers le gestionnaire du module correspondant, qui présente à son tour un sous-menu spécifique.

Aucune donnée d'état n'est persistée entre les sessions, à l'exception des fichiers produits (sauvegardes SQL, exports CSV, rapports d'audit EOL).

3.3 Indépendance des modules

Les trois modules sont strictement indépendants : ils ne s'importent pas mutuellement et ne partagent aucun état global hormis la configuration et le compteur d'échecs d'authentification géré dans main.py. Cette architecture garantit qu'une erreur dans un module n'affecte pas les autres.

4. Description Fonctionnelle des Modules

4.1 Module Diagnostic

Objectif : confirmer rapidement que les briques critiques du siège sont disponibles et produire un état synthétique complet de chaque serveur.

4.1.1 check_ad_dns() — Vérification AD / DNS

Établit une session WinRM/NTLM vers DC01 et exécute Get-Service NTDS, DNS pour interroger l'état des services Active Directory et DNS. Windows uniquement — affiche un message explicite sous Linux.

4.1.2 check_mysql() — Test base MySQL

Tente une connexion TCP à la base MySQL (port 3306) via mysql-connector-python avec un timeout de 3 secondes. Disponible sous Windows et Linux.

4.1.3 verif_systeme_windows() — Ressources Windows Server

Via WinRM/NTLM, exécute un script PowerShell compact qui remonte en une seule passe : version OS + build, uptime (jours/heures/minutes), CPU (LoadPercentage), RAM utilisée/totale en Go, et disques (Win32_LogicalDisk, DriveType=3). Le fix d'encodage [Console]::OutputEncoding=UTF8 est appliqué pour les Windows Server en locale française. Windows uniquement.

4.1.4 verif_systeme_ubuntu_direct() — Ressources Ubuntu

Ouvre une connexion SSH via paramiko et exécute une chaîne de commandes bash séparées par le délimiteur ||| pour éviter tout problème de parsing : version OS (/etc/os-release), uptime (-p), CPU (vmstat 1 2), RAM (free -m), disques (df -h sans tmpfs). Disponible sous Windows et Linux.

4.2 Module Sauvegarde WMS

Objectif : garantir l'existence, l'intégrité et la traçabilité d'exports logiques de la base WMS.

4.2.1 backup_full_wms() — Sauvegarde complète SQL

Appelle mysqldump via subprocess.run() avec les paramètres lus dans config.json. Le fichier est horodaté (full_backup_<base>_YYYYMMDD_HHMMSS.sql) et déposé dans backups/. Le code de retour de mysqldump est analysé pour détecter tout échec. Prérequis : mysqldump doit être présent dans le PATH (MySQL Workbench sous Windows, mariadb-client sous Linux).

4.2.2 export_table_to_csv() — Export CSV d'une table

Liste les tables disponibles (SHOW TABLES), présente ce menu à l'opérateur, puis exécute un SELECT * sur la table choisie. Les lignes sont écrites dans un fichier CSV horodaté (délimiteur point-virgule, UTF-8) dans backups/exports_csv/. Les en-têtes sont extraits automatiquement depuis cursor.description.

4.3 Module Audit d'Obsolescence

Objectif : fournir un inventaire réseau minimal, qualifier le statut EOL des éléments détectés et produire un rapport exploitable.

4.3.1 scanner_plage_reseau() — Scan réseau

Itère sur la plage IP fournie et envoie un ping unique par adresse (ping -n 1 sous Windows, -c 1 sous Linux) via subprocess. La détection du TTL dans la réponse garantit que seules les vraies machines actives sont retournées.

4.3.2 recuperer_infos_os_auto() — Audit WMI

Utilise la bibliothèque wmi (Windows uniquement, import conditionnel) pour interroger Win32_OperatingSystem sur un hôte distant. Extrait le nom du serveur, la désignation de l'OS et en déduit le produit et la version pour l'API EOL.

4.3.3 classifier_statut() et formater_resultat_eol() — Qualification EOL

classifier_statut() retourne un tuple (statut_code, statut_label) avec 5 niveaux : OBSOLETE, BIENTOT_EOL (seuil configurable, défaut 180 jours), CONFORME, ETENDU, INCONNU. formater_resultat_eol() applique le code couleur correspondant pour l'affichage terminal.

4.3.4 generer_rapport() — Rapport JSON et TXT

Génère dans docs/ deux fichiers horodatés : un fichier JSON structuré (résumé chiffré + détail de chaque composant) et un fichier TXT lisible classé par catégorie (OBSOLÈTES, BIENTÔT EOL, CONFORMES, INCONNUS). Le rapport est déclenché automatiquement après chaque audit de masse CSV.

4.3.5 traiter_fichier_csv() — Audit de masse

Lit un fichier CSV (colonnes : nom, produit, version) et appelle l'API EOL pour chaque ligne. Génère automatiquement le rapport à la fin de l'audit.

4.3.6 lister_toutes_versions_os() — Consultation cycles

Récupère et affiche l'intégralité des cycles de vie d'un produit depuis l'API. Outil d'aide à la décision pour la planification des mises à jour.

5. Gestion de la Configuration et des Secrets

5.1 Fichier de configuration

Le fichier config/config.json contient exclusivement les paramètres non sensibles. Il peut être versionné sans risque.

Clé	Description
WMS_IP_DC	Adresse IP du contrôleur de domaine DC01 (192.168.10.10)
WMS_DB_IP	Adresse IP de la base MySQL WMS-DB (192.168.10.21)
WMS_METIER_IP	Adresse IP du serveur applicatif WMS-APP (192.168.10.22)
WIN_USER	Compte Windows pour WinRM et WMI
SSH_USER	Compte Linux pour SSH
WMS_DB_USER	Utilisateur MySQL
WMS_DB_NAME	Nom de la base de données WMS

5.2 Gestion des mots de passe en session

Aucun mot de passe n'est stocké en fichier. À chaque besoin d'authentification, l'outil invoque getpass.getpass() qui lit le mot de passe sans écho terminal. Ce mot de passe est mémorisé en mémoire pendant 5 minutes (300 secondes) via is_timeout_exceeded() dans utils.py. Trois échecs d'authentification consécutifs déclenchent l'arrêt immédiat via sys.exit(1).

6. Ergonomie du Menu Interactif

6.1 Principe général

L'outil s'utilise exclusivement via un menu CLI interactif à deux niveaux. Colorama est initialisée en mode autoreset. Chaque module utilise une couleur distincte pour ses encadrés (vert pour le diagnostic, jaune pour la sauvegarde, cyan pour l'audit).

6.2 Arborescence des menus

Menu principal	Sous-menu	Action déclenchée
1 — Diagnostic	1. État AD / DNS	check_ad_dns() — WinRM/NTLM sur DC01
	2. Test MySQL	check_mysql() — connexion TCP port 3306
	3. Ressources Windows	verif_systeme_windows() — OS, uptime, CPU, RAM, disques
	4. Ressources Ubuntu	verif_systeme_ubuntu_direct() — OS, uptime, CPU, RAM, disques
2 — Sauvegarde	1. Sauvegarde SQL	backup_full_wms() — mysqldump horodaté
	2. Export CSV	export_table_to_csv() — SELECT * vers CSV
3 — Audit	1. Audit cible unique	WMI + API EOL sur WMS_IP_DC (Windows uniquement)
	2. Scan plage réseau	Ping sweep + détection TTL
	3. Audit masse CSV	Lecture CSV + API EOL + rapport JSON/TXT auto
	4. Versions d'un OS	Liste tous les cycles EOL d'un produit

6.3 Codes de retour

La fonction print_result() de utils.py standardise les sorties : status 0 indique un succès (☑ OK), status 2 indique une erreur (✗ ERREUR). Ces codes sont exploitables par un outil de supervision externe comme Zabbix.

7. Compatibilité Multiplateforme

7.1 Principe

Le cahier des charges exige que la solution soit exécutable sous Windows et Linux. NTL-SysToolbox répond à cette exigence grâce à deux mécanismes complémentaires.

7.2 Import conditionnel

Les bibliothèques wmi et pywin32 sont Windows-only. Elles sont importées conditionnellement en début de diagnostic.py et audit.py via platform.system(). Si l'OS n'est pas Windows, les fonctions concernées affichent un message jaune explicite et retournent proprement sans lever d'exception.

7.3 Marqueurs dans requirements.txt

Le fichier requirements.txt utilise les marqueurs d'environnement pip pour ignorer automatiquement wmi et pywin32 lors d'une installation sous Linux :

Entrée requirements.txt	Comportement
wmi; sys_platform == "win32"	Installé uniquement sous Windows, ignoré sous Linux
pywin32; sys_platform == "win32"	Installé uniquement sous Windows, ignoré sous Linux
mysql-connector-python==8.0.33	Version fixée pour compatibilité Python 3.11 à 3.14+

7.4 Scripts de lancement

Deux scripts de lancement sont fournis : lancer_toolbox.bat pour Windows (crée le venv, installe les dépendances, lance main.py) et lancer_toolbox.sh pour Linux (même comportement, compatible bash Ubuntu).

7.5 Fonctionnalités par plateforme

Fonctionnalité	Windows	Linux
Vérification AD / DNS (WinRM)	✓ Disponible	✗ Windows-only
Test base MySQL	✓ Disponible	✓ Disponible
Ressources Windows Server (WinRM/WMI)	✓ Disponible	✗ Windows-only
Ressources Ubuntu (SSH)	✓ Disponible	✓ Disponible
Sauvegarde SQL (mysqldump)	✓ Disponible	✓ Disponible
Export CSV	✓ Disponible	✓ Disponible
Scan réseau (ping)	✓ Disponible	✓ Disponible
Audit WMI cible unique	✓ Disponible	✗ Windows-only

Fonctionnalité	Windows	Linux
Audit de masse CSV + rapport EOL	✓ Disponible	✓ Disponible
Consultation cycles EOL	✓ Disponible	✓ Disponible

8. Compromis et Limites Assumés

Fonctionnalité	Compromis retenu	Évolution possible
Diagnostic Windows/Ubuntu	OS, uptime, CPU, RAM et disques remontés — couverture complète du CDC	Déjà couvert
Compatibilité Linux	WMI/WinRM Windows-only par nature technique — message explicite sous Linux	Détection SSH/SNMP pour les hôtes Linux
Scan réseau	Ping-based + détection TTL — simple et non intrusif	Scan TCP de ports pour hôtes bloquant l'ICMP
Rapport d'audit	JSON + TXT horodatés générés automatiquement dans docs/	Export HTML ou envoi par mail automatique
Sorties JSON diagnostic	create_output() structure les données mais ne les écrit pas en fichier automatiquement	Écriture fichier log = évolution prioritaire
Tests automatisés	Aucun test unitaire ou d'intégration	Ajout de tests pytest sur utils et audit
Module AD/DNS	Vérification de présence des services uniquement	Vérification de réplication AD et zones DNS

9. Workflow Git et Qualité de Code

9.1 Stratégie de branches

- dev : branche de développement actif. Chaque fonctionnalité fait l'objet d'une Pull Request documentée vers test.
- test : branche de validation collective avant validation finale.
- main : branche de production. Seules les versions validées sont mergées ici.

9.2 Conventions et historique

Les commits sont rédigés en français avec un message court décrivant la modification. Les Pull Requests décrivent les fonctionnalités ajoutées et les points de vérification. Les versions stables sont balisées avec des tags Git (v1.0).

Le dépôt public est accessible à l'adresse : https://github.com/Enzo1401/MSPR_6.1 — 44 commits au moment de la livraison.

10. Executive Summary (English)

NTL-SysToolbox is a Python-based command-line tool developed for NordTransit Logistics (NTL), a logistics SME based in northern France. The tool addresses key operational gaps identified by the IT team: lack of automated health checks, untested backups, and no structured EOL tracking for the server fleet.

The tool exposes three independent modules accessed through an interactive CLI menu:

- Diagnostic module: checks Active Directory and DNS services via WinRM/NTLM, tests MySQL connectivity, and retrieves full system metrics (OS version, uptime, CPU, RAM, disk usage) from both Windows Server (WinRM/PowerShell/WMI) and Ubuntu servers (SSH/bash).
- WMS Backup module: performs a full logical dump of the WMS MySQL database using mysqldump and exports individual tables to timestamped CSV files.
- Obsolescence Audit module: performs ping-based network discovery with TTL validation, retrieves OS information from Windows hosts via WMI, cross-references detected versions against the endoflife.date API, and automatically generates a timestamped JSON and TXT report classifying systems as obsolete, soon end-of-life, or compliant.

Key design decisions: Python was chosen for its cross-platform support. Passwords are never stored — collected at runtime via getpass, cached for 5 minutes, cleared after 3 consecutive failures. Configuration is stored in a plain JSON file (no secrets). The wmi and pywin32 libraries are conditionally imported and marked with sys_platform markers in requirements.txt to ensure clean installation on Linux. Two launcher scripts are provided: lancer_toolbox.bat for Windows and lancer_toolbox.sh for Linux.

Known limitations: WMI-based OS audit can only run from a Windows host; diagnostic JSON output is structured in memory but not yet automatically persisted to disk; no automated test suite.

Annexe — Machines Virtuelles Ciblées

Extrait de l'Annexe C du cahier des charges NTL, rappelant les cibles principales de l'outil :

VM	OS / Système	Rôle	Criticité
DC01	Windows Server	Contrôleur de domaine AD/DNS principal	Haute — redondé par DC02
DC02	Windows Server	Contrôleur de domaine secondaire	Haute disponibilité AD
WMS-DB	Ubuntu 20.04	Base MySQL du WMS (192.168.10.21)	Critique — arrêt ops si indispo
WMS-APP	Ubuntu 20.04	Application WMS (192.168.10.22)	Serveur applicatif critique
SUPER-01	Windows Server	Supervision Zabbix (192.168.10.50)	Supervision technique
IPBX-VM	Linux CentOS	Serveur téléphonie IP	Point de défaillance unique appels