WAPT Documentation

Version 1.8.2

Tranquil-IT Systems

janv. 10, 2023

Contents

Bienvenue sur la documentation officielle de WAPT par Tranquil IT en date du 2023-01-10.

Cliquer ici pour une version PDF de la documentation complète.

WAPT est une solution de déploiement logiciel dont le coeur est Open-Source (GPLv3).

WAPT existe en deux versions, *WAPT Community et WAPT Enterprise*.

CHAPTER 1

Certification de sécurité ANSSI

Figure1: Visa de sécurité de l'ANSSI pour WAPT Enterprise 1.5.0.13

Les solutions de cyberscurité sont nombreuses et variées, mais toutes n'offrent pas le même niveau d'efficacité, de robustesse et de confiance. Les Visas de sécurité que délivre l'ANSSI permettent d'identifier facilement les plus fiables d'entre elles et reconnues comme telles à l'issue d'une évaluation réalisée par des laboratoires agréés selon une méthodologie rigoureuse et éprouvée.

CHAPTER 2

Qualification Elémentaire ANSSI

Suite à sa certification CSPN du 14 février 2018, WAPT a obtenu le 15 mars 2018 la Qualification Elémentaire de l'ANSSI.

CHAPTER 3

Principales fonctionnalités

Pour les administrateurs système :

- installer des logiciels de manière silencieuse ;
- tenir à jour le parc logiciel ;
- configurer les logiciels et le système pour diminuer la charge sur les équipes de support ;
- désinstaller des logiciels de manière silencieuse ;
- rendre les *Utilisateurs* plus autonomes pour installer des logiciels de manière sûre et sécurisée ;
- réduire autant que possible la consommation de bande passante pour la réserver aux usages productifs ;

Pour les RSSI

- faire converger le parc logiciel vers une norme de sécurité ;
- préparer votre entreprise à l'arrivée du RGPD et aider votre DPD, qui deviendra un proche collègue à tenir à jour son registre des traitements ;
- ne plus tolérer le fonctionnement des machines en mode *Administrateur* ;
- réduire l'exposition du parc aux vulnérabilités logicielles et aux attaques par propagation latérales ;
- remonter des indicateurs d'audit pour une meilleure connaissance de l'état et de la sécurité du parc ;
- déployer immédiatement pour réagir à une menace type Wannacry ou notPetya ou bien laisser faire automatiquement les choses du quotidien ;

Pour les utilisateurs finaux

- avoir ses logiciels configurés de base pour bien fonctionner dans le contexte de son Organisation et avoir confiance qu'ils fonctionneront correctement ;
- réduire le besoin de support par les équipes informatiques, dont les temps de réaction sont parfois long du fait de leurs charges de travail ;

- avoir une meilleure qualité de travail et rendre prédictible le parc informatique grâce à une configuration standarde pour les logiciels;

CHAPTER 4

Code Source

Vous pouvez accéder au code source sur notre dépôt GitHub : https://github.com/tranquilit/WAPT.

CHAPTER 5

Contribuer

Vous pouvez consulter notre *guide de contribution*.

CHAPTER 6

Support WAPT

- Support commercial : https://www.tranquil.it/page/offreswapt
- Forum : https://forum.tranquil.it/
- Liste de diffusion : https://lists.tranquil.it/listinfo/wapt

6.1 Présentation des grands principes de WAPT

6.1.1 A quoi sert WAPT ?

WAPT déploie, met à jour et et supprime des logiciels et des configurations sur un parc Windows. Le déploiement de logiciels (Firefox, MS Office, ...) s'effectue de manière centralisée avec la console graphique. Le fonctionnement de WAPT s'inspire fortement du gestionnaire de paquets du système GNU / Linux Debian apt, d'où son nom. WAPT est libre et soumis à la licence **GPLv3**. L'édition **Enterprise** est soumise à une licence propriétaire.

WAPT est destiné aux gestionnaires de parcs de PC, de portables, de tablettes Windows fonctionnant avec des versions client de Windows (XP à 10) ou des versions serveur de Windows (2003 à 2019).

Des entreprises de toutes tailles, françaises et internationales, des lycées, des collèges, des écoles, des rectorats, des universités, des centres de recherche, des mairies, des communautés d'agglomération, des communautés de communes, des hôpitaux, des Ministères utilisent **WAPT**.

WAPT existe en deux versions, **Community** et **Enterprise**.

WAPT est particulièrement efficace pour traiter le cas récurrent des **mises à jour Flash et Java** et c'est souvent pour couvrir ce besoin que WAPT est initialement adopté ; ensuite il devient un outil apprécié, voire indispensable pour le quotidien de l'administrateur système.

Si vous êtes un développeur, WAPT en mode ligne de commande peut vous aider à configurer votre PC de développement comme le font Chocolatey / NuGet.

6.1.2 Genèse WAPT

Notre constat après 15 ans d'infogérance

L'administration d'un large parc de PC sous MS Windows est à l'heure actuelle un exercice acrobatique en environment sécurisé :

- les méthodes généralement utilisées (mastérisations type *ghost* ou *clonezilla*) sont efficaces si les parcs machines et les parcs applicatifs sont homogènes et que les profils utilisateurs sont itinérants ;
- les outils de télé-déploiement (*OCSInventory* ou *WPKG*) diffusent les logiciels mais ne permettent pas d'effectuer de manière simple les personnalisations qui évitent les demandes de support utilisateur ;
- les logiciels de petits éditeurs nécessitent souvent des droits *Administrateur Local* pour fonctionner ;
- les solutions développées pour adresser ces trois problèmes s'appuient sur des méthodes soit trop chères, soit insuffisantes, et dans tous les cas trop complexes ;

Les hypothèses de travail qui animent le développement de WAPT

Le développement de WAPT est animé par deux principes :

- ce qui est **compliqué** doit être rendu **simple** ;
- ce qui est **simple** doit être rendu **trivial** ;

WAPT s'appuie sur un jeu d'hypothèses fondamentales :

- les adminsys connaissent les langages de script, et WAPT a choisi python pour la profondeur et l'étendue de ses librairies ;
- les adminsys qui ne connaissent pas encore très bien les langages de script doivent pouvoir s'inspirer d'exemples simples et efficaces qu'ils adapteront à leurs situations spécifiques ;
- les adminsys doivent pouvoir communiquer l'efficacité de leurs actions à leur direction et restituer les écarts de process aux auditeurs de leur société ;
- les adminsys doivent pouvoir collaborer avec des collègues adminsys dans un climat de confiance ; ainsi les dépôts WAPT internes et privés servent des logiciels qu'ils signent eux-mêmes. Sinon, ils choisissent des dépôts publics qui leur fournissent les garanties de sécurité qu'ils jugent suffisantes ;
- les adminsys sont conscients que les postes utilisateurs servent des objectifs métier et une certaine personnalisation doit être rendue possible ; l'adaptation du parc aux enjeux métiers est facilitée par la notion de groupes qui permet de sélectionner des machines respectant un critère de tri pour personnaliser leur configuration ;

6.1.3 Principes fondamentaux

Principe de Paquets / Dépôts

Les paquets WAPT

La structure d'un paquet WAPT est similaire à celle d'un paquet **.deb** de Debian Linux. Chaque paquet WAPT embarque avec lui les binaires qui seront exécutés et les autres fichiers dont il aura besoin.

Un paquet est transportable facilement.

Voici la structure d'un paquet WAPT :

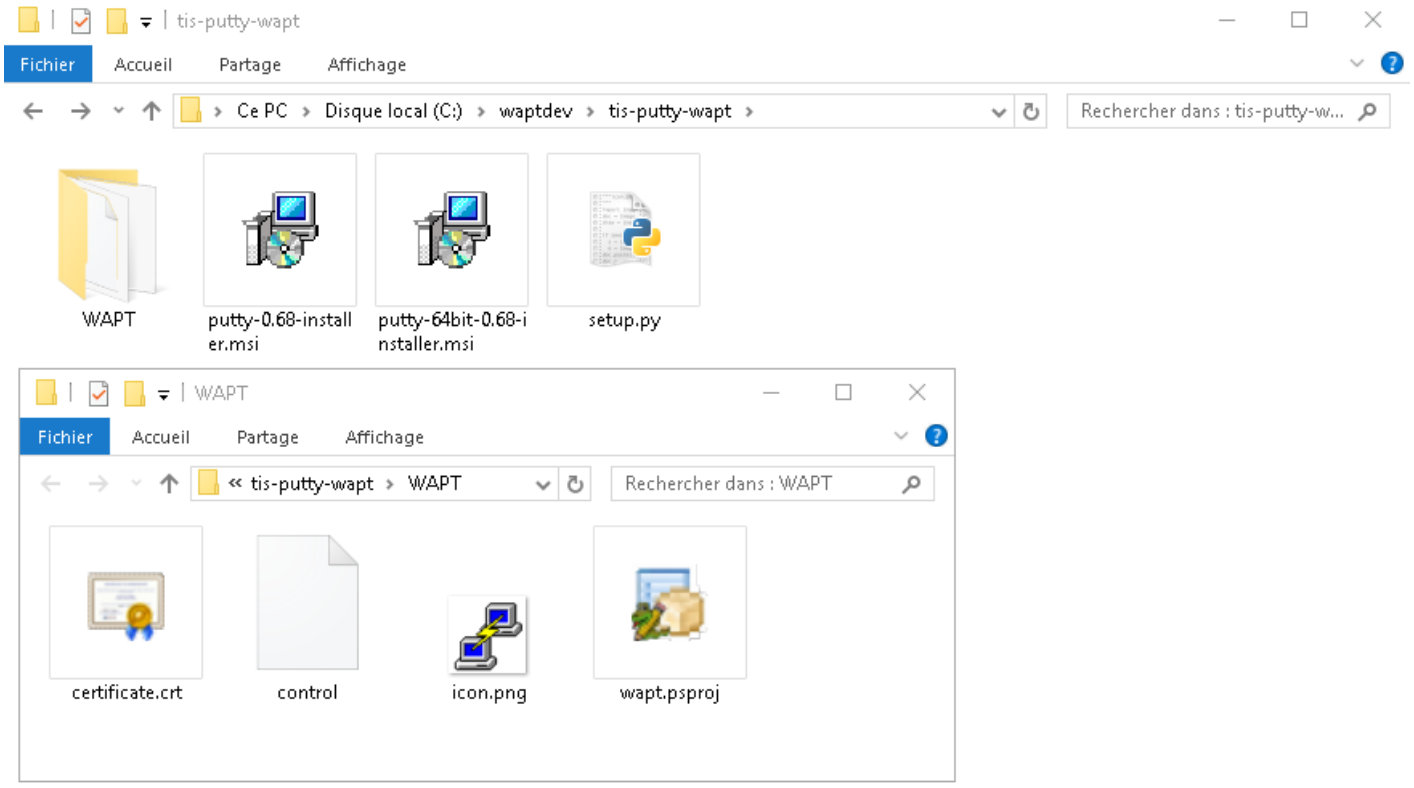


Figure1: Structure d'un paquet WAPT

Les dépôts WAPT

Les paquets sont stockés dans un répertoire web. Ils ne sont pas stockés dans une base de données.

Ils sont servis par le serveur web **`Nginx`**, disponible pour Linux et Windows.

Seul le fichier d'index `Packages` est nécessaire. Le fichier `Packages` permet d'indiquer au client la liste des paquets disponibles sur le dépôt, et les informations de base de chacun des paquets du dépôt.

Ce fonctionnement permet de mettre en place facilement un mécanisme de réplication entre plusieurs dépôts.

Types de paquets WAPT

Il existe 7 types de paquets WAPT:

Figure2: Représentation d'un paquet WAPT simple

Les paquets *base*

Ce sont les paquets logiciels classiques.

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *group*

Ce sont des paquets qui permettent de faire des groupes de logiciels ou des groupes de machines.

Indication:

- un groupe de logiciels correspond en réalité à un profil de machine (ex: **compta**) ;
- un groupe de machines correspond en réalité à une salle, un bâtiment, etc ;
- une machine peut appartenir à plusieurs groupes (ex : un ou plusieurs profils de machine dans une salle dans un bâtiment) ;

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *host*

Les paquets « machines » portent le nom *UUID* Bios de la machine.

Chaque Poste client va par défaut chercher son paquet machine pour connaître la liste des paquets qu'il doit installer (*dépendances*).

Ces paquets sont stockés dans le répertoire https://srvwapt.mydomain.lan/wapt-host/.

Les paquets *unit*

Nouveau dans la version 1.6: Enterprise

Les paquets « unit » portent le nom complet d'une OU (Unité Organisationnelle), exemple : **OU=BUREAU1,OU=prod,OU=computers,DC=mydomain,DC=lan**.

Chaque Poste client va par défaut chercher les paquets *unit* dont il fait partie :

- OU=BUREAU1,OU=prod,OU=computers,DC=mydomain,DC=lan ;
- OU=computers,DC=mydomain,DC=lan ;
- DC=mydomain,DC=lan ;

et ensuite installer la liste des dépendances associées.

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *unit* ne sont pas explicitement attribués à l'hôte (c'est-à-dire en tant que dépendances dans le paquet *host*) mais sont implicitement pris en compte par le moteur de dépendances de l'agent WAPT lors de la mise à niveau WAPT.

Note: Si la machine change d'OU, alors les paquets *unit* obsolètes seront désinstallés.

Les paquets *waptwua*

Les paquets *waptwua* contiennent le liste des mises à jour Windows autorisées et interdites.

Lorsque ce paquet est installé sur le terminal, la prochaine analyse de mise à jour effectuée par WAPT choisira les mises à jour Windows en fonction de ce filtrage.

Si la machine a plusieurs listes de paquets *waptwua* applicables, alors WAPT combine les règles.

Lorsque ce paquet est installé sur l'hôte, le prochain **`update`** recherchera les mises à jour officielles de Windows applicables à l'hôte en fonction de ce filtrage.

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *self-service*

Nouveau dans la version 1.7: Enterprise

Les paquets *self-service* contiennent une liste de groupes ou d'utilisateurs (Active Directory ou local) et leurs listes associées de paquets que les utilisateurs seront autorisés à installer par eux-mêmes.

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *profile*

Nouveau dans la version 1.7: Enterprise

Les paquets *profile* sont similaires aux paquets *groupe*.

Cependant, les paquets *profile* fonctionnent un peu différemment et sont plus utiles lorsqu'un serveur Active Directory fonctionne dans l'*Organisation* :

- l'agent WAPT listera les groupes Active Directory auxquels appartient l'hôte ;
- si un paquet *profile* porte le même nom que le groupe Active Directory, l'agent WAPT installera automatiquement le paquet *profile* pour le groupe Active Directory auquel il appartient ;

Si l'hôte n'est plus membre de son groupe Active Directory, le paquet *profile* sera désinstallé.

Ils sont stockés dans le répertoire web https://srvwapt.mydomain.lan/wapt/.

Les paquets *profile* ne sont pas explicitement attribués à la machine (c'est-à-dire en tant que dépendances dans le paquet *host*) mais sont implicitement pris en compte par le moteur de dépendances de l'agent WAPT lors de la mise à niveau WAPT.

Note: Pour des raisons de performance, cette fonctionnalité n'est activée que si l'option *use_ad_groups* est activée dans `wapt-get.ini`.

Principe de dépendances

Dans WAPT tout fonctionne selon le principe des dépendances.

Par défaut, le client WAPT va chercher à installer son paquet machine. Le paquet *host* listera les autres paquets à installer sur la machine.

Ainsi, le paquet *host* sera correctement installé si toutes ses dépendances sont satisfaites.

Chaque dépendance devra satisfaire aux sous-dépendances et ainsi de suite.

Quand toutes les dépendances sont satisfaites, la machine remonte en état **OK** dans la console, signifiant que la machine est conforme au profil que l'*Administrateur* ou le *Déployeur de Paquets* aura défini pour elle.

Indication: Lorsqu'on attribue un paquet logiciel à une machine, on lui affecte le nom canonique du logiciel sans son numéro de version (ex: je veux que **`Freemind`** soit installé sur cette machine dans sa dernière version et que **`Freemind`** soit configuré pour que l'*Utilisateur* ne m'appelle pas car il ne trouve pas l'icône sur son bureau!).

Pour chaque paquet en dépendance, l'agent WAPT local s'occupera d'aller chercher automatiquement la dernière version du paquet. Donc si plusieurs versions de **`Freemind`** sont disponibles sur le dépôt, l'agent WAPT ira toujours chercher la dernière version, à moins que vous ayez épinglé la version pour cause de compatibilité avec d'autres outils.

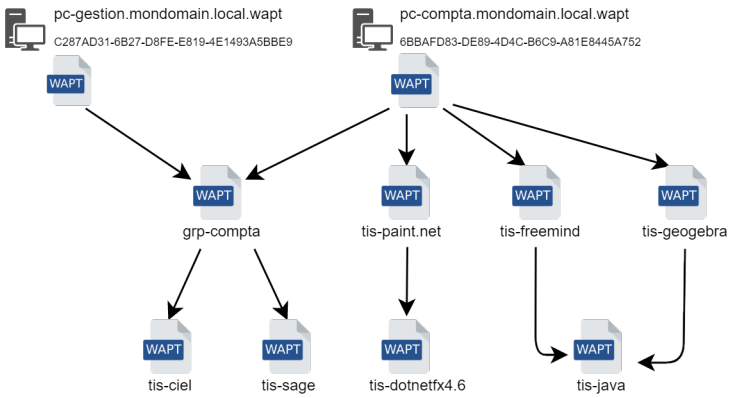


Figure3: Schéma conceptuel de la notion de dépendance

Ensuite, lorsque l'agent contacte le dépôt pour vérifier s'il y a de nouvelles mises à jour, il compare les versions des paquets du dépôt avec sa propre liste locale de paquets déjà installés sur la machine.

Si une mise à jour d'un paquet déjà installé est disponible, le client basculera le statut du paquet en **NEED UPGRADE**. Il installera ainsi les mises à jour au prochain **`upgrade`**.

Principe de clé privée / clé publique

Introduction

Comme **APK** sous Android, les paquets WAPT sont signés et embarquent une somme de contrôle de la liste des fichiers contenus dans le paquet.

Cette méthode de signature permet de garantir la provenance et l'intégrité du paquet.

Principe de clé privée / clé publique

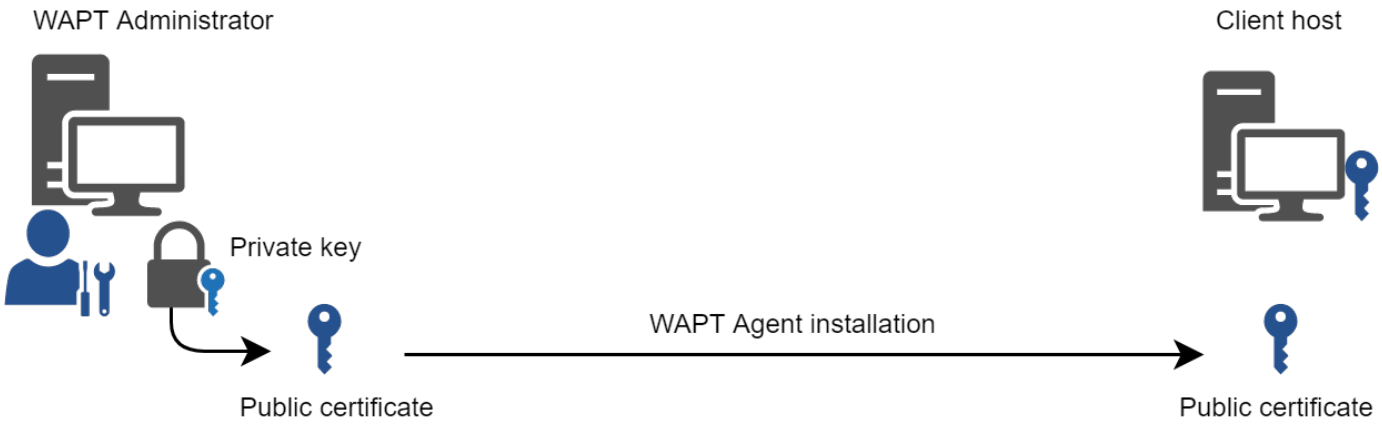


Figure4: Principe de clé privée / clé publique

L'usage de WAPT nécessite un couple clé privée / certificat public (auto-signé, issu d'une *Autorité de Certification* interne, ou commercial).

La **clé privée** servira à **signer** les paquets WAPT et un **certificat public** associé (CA associé ou certificat auto-signé) sera distribué sur chaque client WAPT pour qu'il puisse valider les fichiers signés avec la clé privée.

Les différents certificats publics seront stockés dans un dossier `ssl` de l'agent WAPT, ce dossier peut contenir plusieurs certificats publics.

Vérification des paquets

Lorsqu'un paquet WAPT est téléchargé, l'agent WAPT (**`waptagent`**) vérifie l'integrité des fichiers, puis vérifie que le paquet a été correctement **signé**.

Si la signature du paquet WAPT ne correspond à aucune des clés publiques situées dans `C:\Program Files (x86)\wapt\ssl`, l'agent WAPT refusera d'installer le paquet.

Pour en savoir plus rendez-vous dans la section de documentation pour *protéger l'intégrité du processus d'installation des paquets WAPT*.

Importance de la clé privée

Attention: La clé privée ne doit pas être stockée sur le serveur WAPT, ni sur aucune machine potentiellement accessible à une entité non-autorisée car toute la sécurité de WAPT repose sur la confidentialité des jeux de clés privées.

Elle doit être stockée en lieu sûr car **celui qui contrôle votre clé contrôle votre parc** !

Enfin, pour un maximum de sécurité, la clé privée pourra être une smartcard ou un jeton cryptographique que les *Administrateurs* et *Déployeurs de Paquets* transporteront physiquement sur eux, utilisant leur smartcard ou leur jeton cryptographique ponctuellement pour signer un paquet WAPT.

Note: Depuis la version 1.5 de WAPT, la clé privée est protégée par un mot de passe.

6.1.4 Fonctionnement et architecture WAPT

remontée des informations d'inventaire

WAPT fait un inventaire matériel et logiciel de chaque machine.

Cet inventaire est stocké dans une petite base de données incorporée dans l'agent WAPT.

- lors d'un premier enregistrement, le client WAPT remonte la totalité de son inventaire (BIOS, machine, logiciels) au serveur ;
- lors de chaque mise à jour du client, l'agent WAPT remonte le delta de l'inventaire au serveur ;

Cet inventaire centralisé permettra de filtrer les postes par matériel, logiciel ou de filtrer sur tout autre paramètre remonté dans l'inventaire.

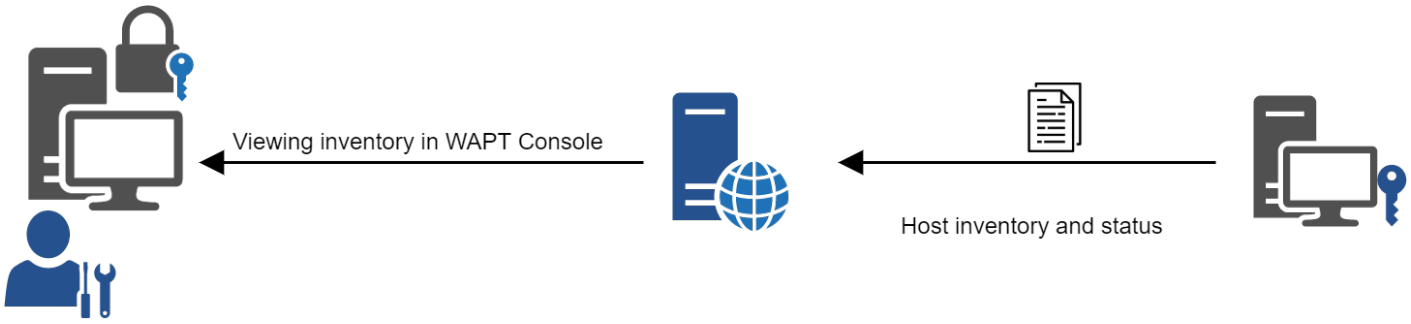


Figure5: Fonctionnement de la remontée d'inventaire

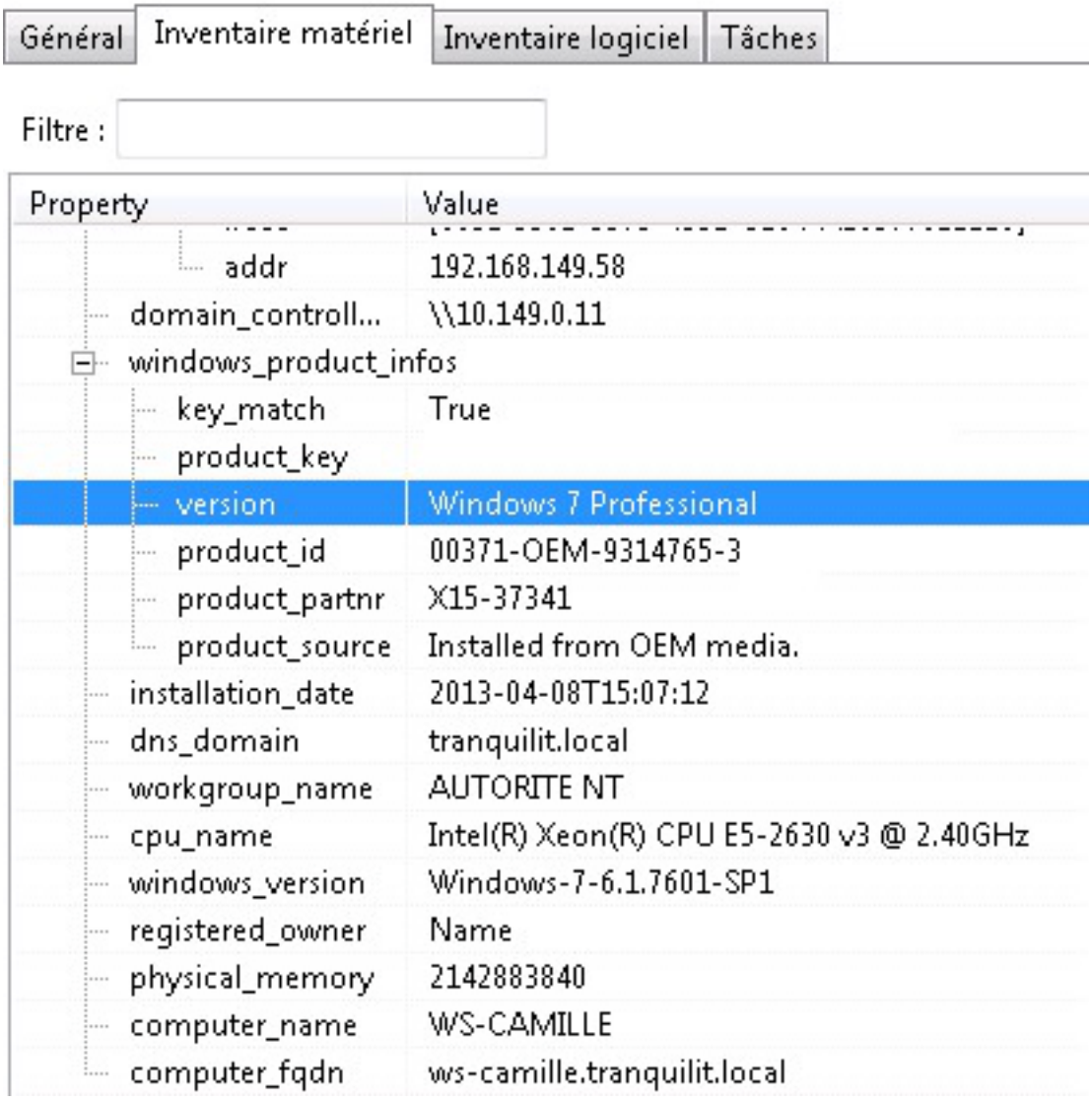


Figure6: L'inventaire dans la console WAPT

La remontée d'informations

L'agent WAPT remonte également le statut des paquets WAPT.

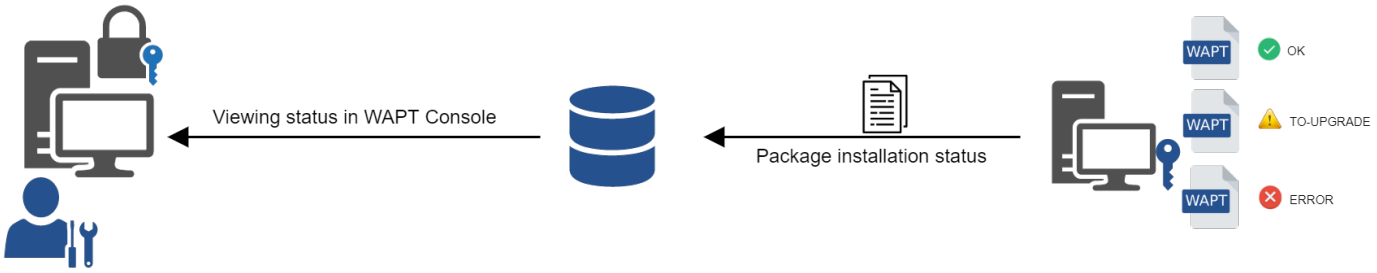


Figure7: La remontée du statut des paquets vers le serveur WAPT

En cas d'erreur lors de l'installation du paquet, l'information sera transmise au serveur WAPT. La machine apparaîtra alors en **ERREUR** dans la console.

Figure8: Elément en erreur dans la console WAPT

L'"*Administrateur* pourra ensuite constater l'erreur dans la console et la corriger.

À chaque **`upgrade`**, WAPT tentera une nouvelle fois l'installation du paquet en erreur.

Note: Pour plus de sécurité, depuis la version 1.3.13, l'inventaire remontée par l'agent wapt est signé par l'agentwapt.

Pour en savoir plus rendez-vous sur la documentation pour *signer les remontées d'inventaire*.

Diagramme complet du fonctionnement de WAPT

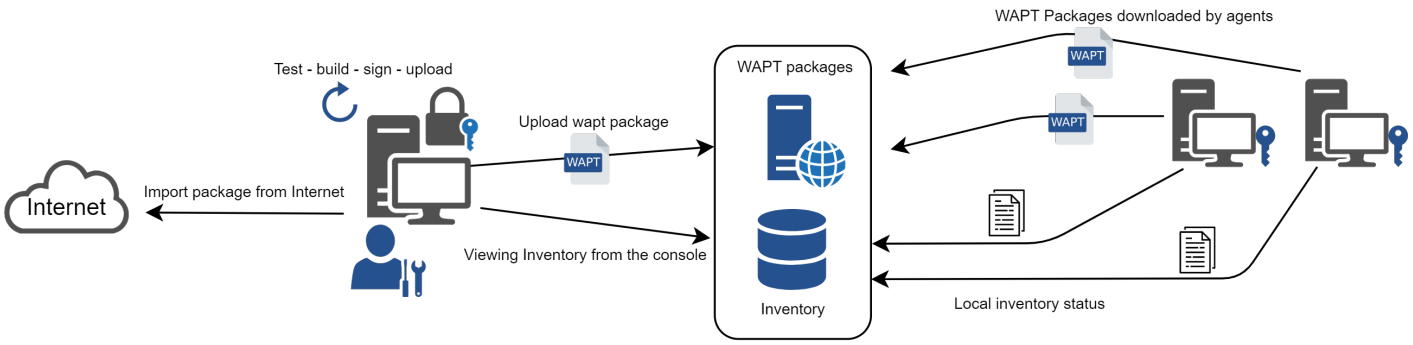


Figure9: Fonctionnement général de WAPT

On retrouve ici le comportement classique de WAPT, de la duplication d'un paquet depuis un dépôt externe sur Internet jusqu'à son déploiement sur les clients.

Il faut lire ce schéma dans le sens des aiguilles d'une montre :

- importer des paquets depuis un dépôt (ou créer ex-nihilo un paquet) ;
- tester, ensuite valider, puis construire et enfin signer le paquet ;
- charger le paquet sur le dépôt principal ;
- ensuite, téléchargement automatique des paquets par les clients WAPT ;
- exécution des paquets selon la méthode sélectionnée :
  - L'*Administrateur* force l'**upgrade** ;
  - l'*Utilisateur* choisit le moment opportun pour lui ;
  - une tâche planifiée lance l'exécution de la mise à jour ;
  - la mise à jour est exécutée à l'extinction de la machine ;
- remontée des informations d'inventaire ;
- consultation de la remontée d'inventaire via la console ;

Comportement de l'agent WAPT pour installer / supprimer / configurer en contexte utilisateur / auditer un paquet

Un concept clé qui peut être difficile à comprendre est le comportement de l'agent WAPT lors de l'installation d'un paquet et les considérations qui l'entourent.

L'installation du paquet d'agents WAPT peut être divisée en étapes :

- paquet téléchargé dans le cache de l'agent;
- décompression du paquet dans un dossier temporaire ;
- Le contenu de `setup.py` est stocké dans la base de données de l'agent WAPT située dans `C:\Program Files (x86)\wapt\db\waptdb.sqlite`;
- les logiciels sont installés à partir des fichiers décompressés :
- en cas de succès : le paquet téléchargé + les fichiers décompressés sont supprimés et le statut est envoyé au serveur ;
- en cas d'échec : le paquet téléchargé est conservé - les fichiers décompressés sont supprimés - le statut d'erreur est envoyé au serveur ;

Ce comportement est important car il a un impact sur les actions futures.

Par exemple, quand on supprime un paquet WAPT, les étapes suivantes sont effectuées :

- Le contenu de `setup.py` est récupéré depuis la base de données située dans `C:\Program Files (x86)\wapt\db\waptdb.sqlite` ;
- la valeur de *UninstallString* récupérée depuis la base de données, est exécutée ;
- Si elle est définie, la fonction **`uninstall()`** est exécutée à partir du code source récupéré ;

Des étapes similaires sont reproduites lors de l'exécution de **`session_setup`** et **`audit`**.

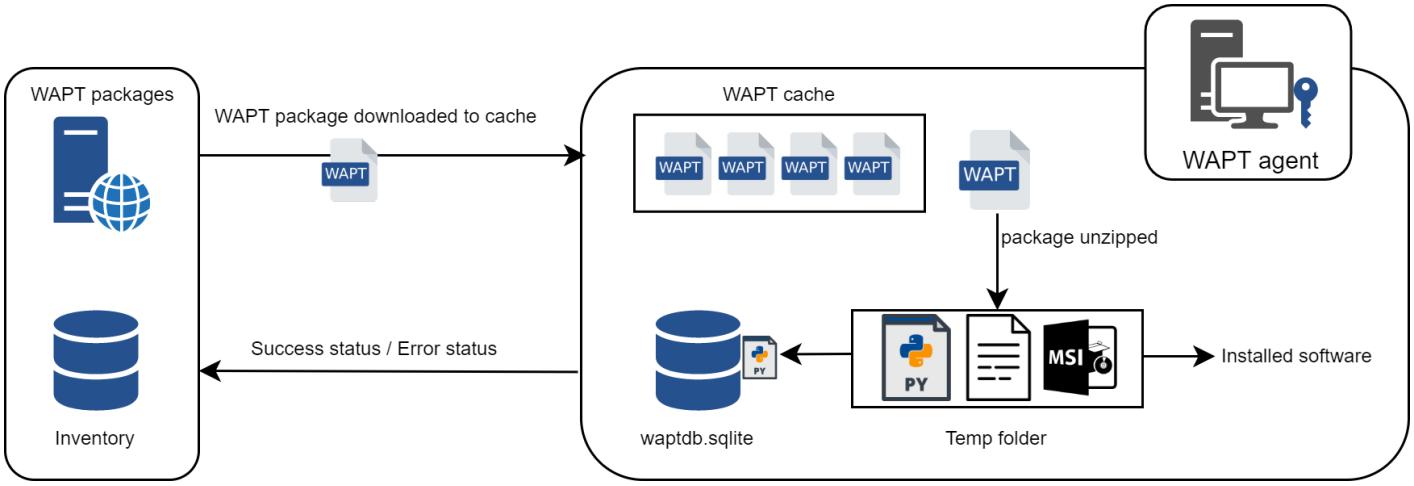


Figure10: Comportement de l'installation d'un paquet WAPT

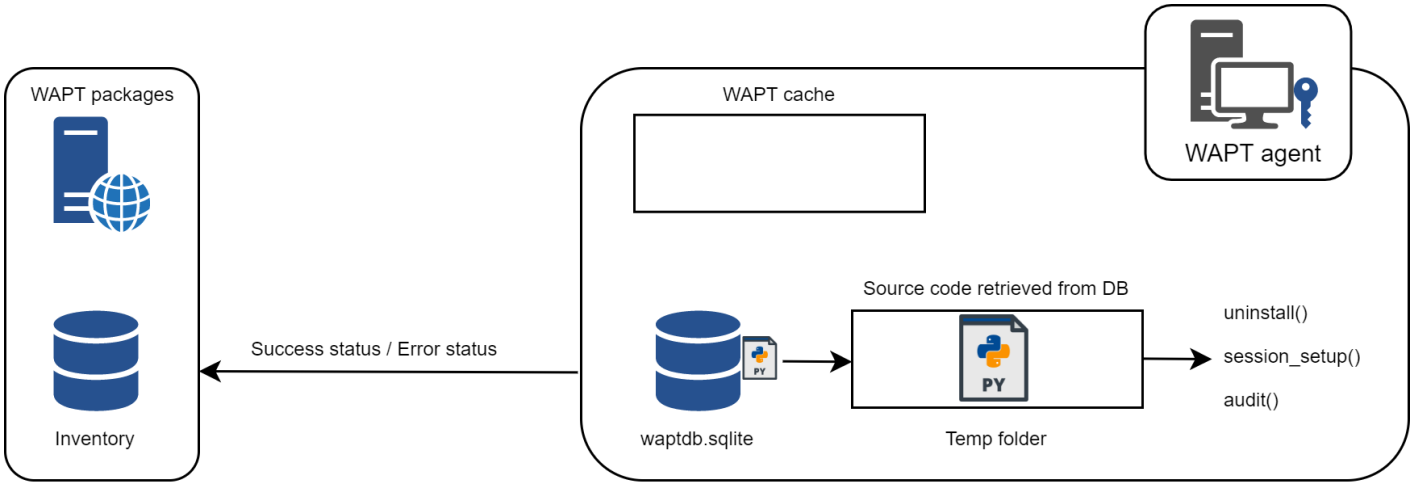


Figure11: Comportement de l'agent WAPT avec la désinstallation, le session_setup et l'audit

Architecture du serveur WAPT

L'architecture du serveur WAPT repose sur plusieurs rôles distincts et en partie dissociables :

- rôle de dépôt : *distribution de paquets* ;
- rôle d'inventaire : *inventaire matériel et logiciel* ;
- rôle de proxy : *proxy de commande depuis la console* ;

Rôle de dépôt

Le serveur WAPT a pour premier rôle celui de dépôt de fichiers web.

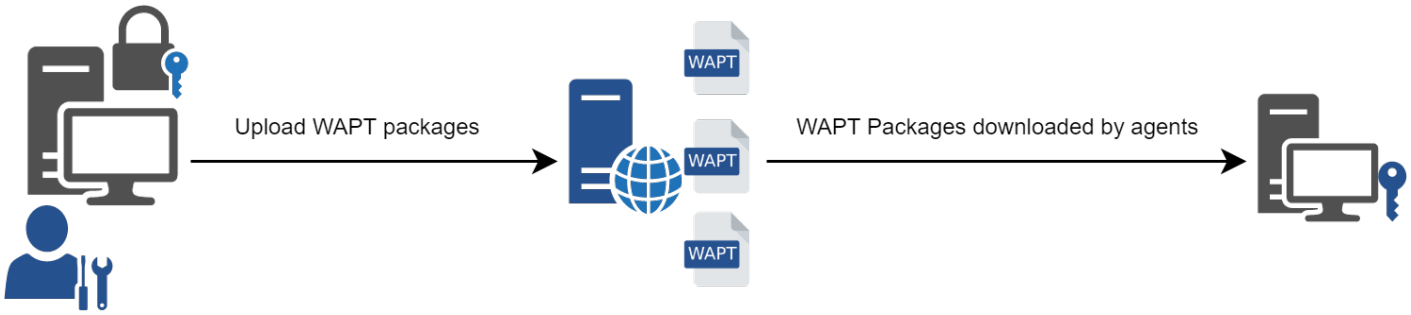


Figure12: Fonctionnement du dépôt WAPT

- ce rôle de dépôt est accompli par un serveur web Nginx ;
- le dépôt permet la distribution des paquets WAPT, des installeurs **`waptagent`** et **`waptsetup`** ;
- les paquets WAPT sont accessibles avec un navigateur web à l'adresse : https://srvwapt.mydomain.lan/wapt ;
- les paquets machines sont contenus dans un répertoire inaccessible par défaut (https://srvwapt.mydomain.lan/wapt/wapt-host/) ;

Rôle d'inventaire

Le serveur WAPT a pour deuxième rôle celui de serveur d'inventaire.

Le serveur d'inventaire est un service passif qui collecte les informations que les agents WAPT lui envoient :

- inventaire matériel ;
- inventaire logiciel ;
- statut des paquets WAPT ;
- état des tâches (*running*, *pending*, *error*) ;

Note: Le service n'est pas actif dans le sens où le serveur d'inventaire ne fait que recevoir de l'information. Aucun service actif interroge directement les agents WAPT. Par conséquent en cas de défaillance du serveur d'inventaire, l'inventaire se régénérera automatiquement à partir des remontées d'inventaire des agents déployés.

En version **Community**, ce service d'inventaire n'a pas d'interface web pour la visualisation des données, l'accès aux données d'inventaire se fait au travers de la console WAPT.

En version **Enterprise**, l'inventaire est accessible à partir d'une console web de visualisation des données de type *Business Intelligence*.

Rôle de Proxy

Le serveur WAPT a pour troisième rôle celui de Proxy de commande.

Il agit comme un relai entre la console et les agents WAPT déployés.

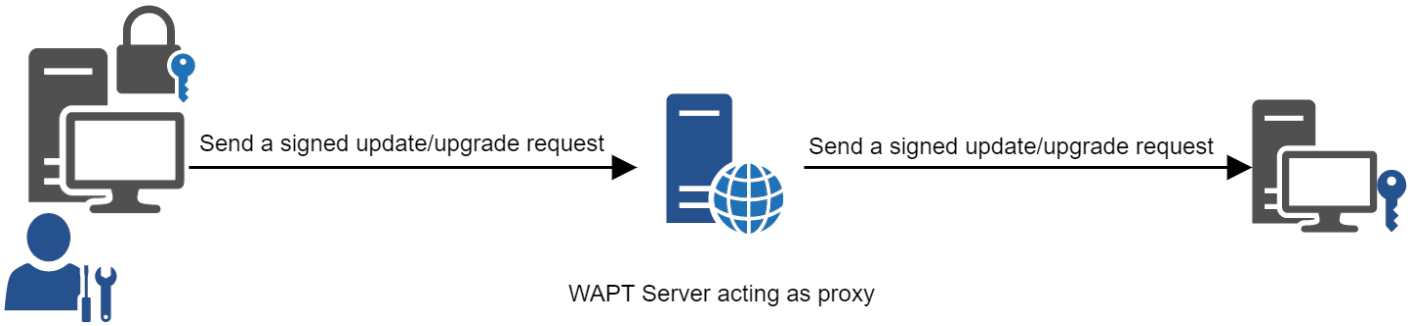


Figure13: Fonctionnement du proxy WAPT

Note: Toutes les actions envoyées aux agents WAPT sont signées par la clé privée de l'*Administrateur*. Sans la clé privé, aucune action ne pourra être envoyée sur les postes. Pour en savoir plus a propos de la signature des actions rendez-vous dans la documentation sur la *signature des actions envoyées aux agents WAPT*.

Les interactions classiques de WAPT

update

Lorsqu'on lance la commande **`update`** sur un agent (depuis la console, en ligne de commande ou depuis le **`wapttray`**), on dit à l'agent d'aller vérifier sur le dépôt WAPT si des nouveaux paquets ou des nouvelles versions de paquets sont disponibles. Par défaut, l'agent WAPT interroge son dépôt toutes les deux heures.

Si la date du fichier `Packages` a changé depuis le dernier **`update`**, alors l'agent WAPT télécharge le nouveau fichier `Packages` (entre 20 et 100ko).

L'agent WAPT compare ensuite le nouveau fichier `Packages` avec sa base de données locale.

Si l'agent WAPT constate qu'un paquet doit être ajouté, supprimé ou mis à jour, il basculera l'état du paquet en *NEED-UPGRADE*.

Il ne lancera pas immédiatement l'installation du paquet. Il attendra l'ordre **`upgrade`** pour exécuter la mise à jour.

upgrade

Lorsqu'on lance la commande **upgrade** (depuis la console, en ligne de commande, depuis une tâche planifiée sur le poste ou manuellement depuis le *wapttray*), on ordonne à l'agent d'installer les paquets en statut *NEED-UPGRADE*.

Un **update** doit précéder un **upgrade**, sinon l'agent ne saura pas que des mises à jour sont devenues disponibles.

Fonctionnement de l'agent WAPT

Par défaut l'agent WAPT fera un **update / download-upgrade** au démarrage du service, puis toutes les deux heures pour vérifier s'il a des choses à faire.

C:\Program Files (x86)\wapt\

Un **upgrade** sera ensuite lancé automatiquement à l'arrêt de l'ordinateur avec le **waptexit**. Un *Administrateur* pourra également forcer un **upgrade** depuis la console.

Si le serveur WAPT n'est pas disponible au moment de l'"**upgrade**, l'agent WAPT pourra tout de même installer ses paquets en attente car ils seront disponibles dans le cache.

L'inventaire sera ensuite envoyé quand le serveur WAPT sera de nouveau disponible.

Les 4 buts de l'agent WAPT sont donc :

- d'installer un paquet *host*, *group* ou *unit* si celui-ci est disponible ;
- de supprimer les paquets obsolètes ;
- de résoudre les dépendances et les conflits ;
- de maintenir tous ses paquets WAPT installés à jour par rapport à ceux du dépôt ;
- d'envoyer régulièrement son inventaire matériel et l'état d'installation de ses paquets ;

6.1.5 La création de paquets WAPT

Le langage de WAPT et les environments de développement

WAPT est édité avec le langage de programmation Python.

N'importe quel environment de développement rapide (RAD) destiné à la programmation python convient.

WAPT a développé quelques extensions très utiles dans le RAD PyScripter https://sourceforge.net/projects/pyscripter.

Tranquil IT recommande l'usage de **PyScripter**, disponible dans le paquet *tis-waptdev*.

Les grands concepts du développement de paquets WAPT

La puissance de Python

Toute la puissance de **Python** peut être avantageusement exploitée.

De nombreuses librairies existent en standard avec Python pour :

- faire des boucles de vérification (if ... then ... else ...) ;
- copier, coller, déplacer des fichiers ou des répertoires ;

- tester l'existence ou l'absence de fichiers ou de répertoires ;
- tester l'existence ou l'absence d'éléments de base de registre ;
- tester des droits, les modifier ;
- interroger des sources de données extérieures (annuaires, bases de données, autres référentiels) ;
- etc . . .

La puissance de WAPT

Functions most commonly used with WAPT were simplified within libraries called *Setuphelpers*.

Setuphelpers libraries simplify the process of creating and testing WAPT packages, thus validating WAPT's main objectives:

- **ce qui était compliqué est rendu simple avec WAPT** ;
- **ce qui était simple est rendu trivial avec WAPT** ;

Je veux maintenant *installer mon serveur WAPT* !!

6.2 Principes de sécurité de WAPT

Sont documentés ici différents principes avancés de sécurité incorporés dans WAPT.

La lecture de cette documentation n'est pas indispensable pour utiliser WAPT ; elle est cependant recommandée pour vous permettre de mieux comprendre certains choix architecturaux.

6.2.1 Présentation des principes de sécurité

Préambule et définitions

Attention: Le service WAPT fonctionne en compte système **privilégié**.

Attention: A partir de la version 1.5 de WAPT, le répertoire d'installation par défaut devient `C:\Program Files (x86)\wapt`.

Indication: les sous-composantes **`wapttray`**, **`waptservice`** et **`waptexit`** de l'agent WAPT peuvent être optionnellement désactivées en fonction du contexte d'usage.

Périmètre à sécuriser

Les éléments à sécuriser qui concernent strictement WAPT sont :

- **le serveur WAPT** (*waptserver*) ;
- **les agents WAPT** (*wapt-get*) et ses sous-composantes (*wapttray*, *waptservice* et *waptexit*) ;
- **la console de management** (*waptconsole*) ;
- **les communications réseaux** entre ces différentes composantes ;

En complément des éléments listés ci-dessus, un exploitant de WAPT devra choisir et suivre une méthodologie adaptée au contexte de son *Organisation* pour :

- assurer un téléchargement sûr de tous les autres fichiers servant à constituer un paquet WAPT ;
- rédiger le script python d'installation d'un paquet WAPT de telle manière à éviter toute faille de sécurité ou de confidentialité exploitable ;
- gérer de manière sûre les clés privées de signature des paquets ;
- gérer de manière sûre les Autorités de certification et de révocation des certificats SSL et HTTPS ;

La gestion sûre de ces éléments complémentaires est exclue du périmètre de cette documentation.

Description des utilisateurs typiques

Les rôles suivants doivent être compris pour évaluer les principes de sécurité présents dans WAPT :

- **Utilisateur**

  individu / utilisateur d'une machine équipée de l'agent WAPT (**Enterprise** et **Community**) ;

- **Déployeur de Paquets**

  individu pouvant signer des paquets ne contenant pas de code python (en général les paquets de type *group*, *unit* et *host*) et les charger sur le dépôt principal (**Enterprise**) ;

- **Développeur de Paquets**

  individu pouvant signer des paquets, qu'ils intègrent ou non du code python et les charger sur le dépôt principal (**Enterprise**) ;

- **SuperAdmin**

  compte unique ayant tous les droits dans WAPT (**Community**) ;

- **Administrateur Local**

  utilisateur disposant des droits d'administration locaux sur les postes équipés de WAPT (**Enterprise** et **Community**) ;

Note: En fonction du contexte de la documentation et de la version du produit, un *Administrateur* désignera un *Déployeur de Paquets*, un *Développeur de Paquets* ou bien le *SuperAdmin*.

Note: Les *Utilisateurs* membres du groupe de sécurité Active Directory **waptselfservice** sont considérés comme des *Administrateurs Locaux* du point de vue de la sécurité WAPT.

Description des biens sensibles

Par définition, un bien sensible est une donnée (ou fonction) jugée comme ayant de la valeur pour un attaquant.

Sa valeur est estimée selon des critères de sécurité (aussi appelés besoins de sécurité) :

- disponibilité ;
- intégrité ;
- confidentialité ;
- authenticité ;

Les biens sensibles à protéger sont les suivants :

Bien sensible B1 : Les communications

Les communications entre le serveur central et les agents ainsi que les communications entre la console et le serveur sont un bien sensible et doivent être protégées.

Note: Besoin de sécurité des communications :

- intégrité ;
- confidentialité ;
- authenticité ;

Bien sensible B2 : Les données d'inventaire

Les informations sur l'état de déploiement des paquets, ainsi que configuration matérielle et logicielle des postes clients sont un bien sensible et doivent être protégées.

Note: Besoin de sécurité des données d'inventaire :

- intégrité ;
- confidentialité ;

Bien sensible B3 : Les journaux d'historique

Les journaux générés par WAPT sur le serveur central et les agents sont un bien sensible et doivent être protégés.

Note: Besoin de sécurité des journaux d'historique

- disponibilité ;

Bien sensible B4 : Les valeurs de configuration

Les valeurs de configuration du serveur (clés du serveur https, configuration accès à la base de données, configuration de l'authentification au serveur) sont un bien sensible et doivent être protégés.

Note: Besoin de sécurité des valeurs de configuration :

- intégrité ;
- confidentialité ;

Bien sensible B5 : Les exécutables WAPT installés sur les postes client

Les exécutables WAPT installés sur les postes client managés (contenu du répertoire wapt incluant les binaires, les dll, les fichiers de configuration et la base de données) sont un bien sensible et doivent être protégés.

Note: Besoin de sécurité des valeurs de configuration :

- intégrité ;

Bien sensible B6 : L'authentification

Les données d'authentification à la console d'administration ainsi que les données d'authentification des agents sur le serveur (clé publique de chaque agent WAPT) sont un bien sensible et doivent être protégées.

Note: Besoin de sécurité de l'authentification :

- intégrité ;
- confidentialité ;

Description des hypothèses sur l'environment d'exploitation de WAPT

Par définition, les hypothèses sont des déclarations portant sur le contexte d'emploi de WAPT ou de son environment.

Les hypothèses suivantes sur l'environment d'exploitation de WAPT doivent être considérées :

Hypothèse H1 : Les Administrateurs et Gestionnaires de Déploiement WAPT sont formés

Les *Administrateurs* et les *Déployeurs de Paquets* sont formés à l'usage de WAPT. En particulier, ils doivent s'assurer que leurs identifiants et clés de sécurité restent secrets.

Hypothèse H2 : Les systèmes subjacents à WAPT sont sains

Les systèmes d'exploitation sur lesquels les agents WAPT s'exécutent mettent en oeuvre des mécanismes de protection adéquats (confinement, contrôle d'accès, etc.) paramétrés et configurés selon les bonnes pratiques.

Les systèmes d'exploitation sont à jour des correctifs en vigueur au moment de l'installation, ils sont sains et exempts de virus, chevaux de Troie, etc.

Hypothèse H3 : Les binaires nécessaires au fonctionnement de WAPT sont intègres

Toutes les librairies et outils nécessaires au fonctionnement de WAPT sont considérées saines. A la réception d'une requête par l'agent WAPT, il vérifie que la requête est correctement signée ;

Hypothèse H4 : Les paquets WAPT sont construits de manière sûre

Il est de la responsabilité de l'"*Administrateur* de s'assurer que les fichiers destinés à être intégrés dans des paquets WAPT proviennent de sources sûres et sont en particuliers exempts de virus, chevaux de Troie, etc.

Hypothèse H5 : Les Utilisateurs des postes client ne sont pas Administrateurs Locaux

Un *Utilisateur* n'a pas les droits d'administration de son poste de travail. Sinon l'*Utilisateur* est considéré comme un *Administrateur Local*.

En particulier, l'*Utilisateur* n'a pas les droits d'écriture dans le répertoire d'installation du client WAPT.

Hypothèse H6 : Les Administrateurs Locaux des postes client sont formés

L'*Administrateur Local* d'un poste client doit être formé à l'exploitation de WAPT, ou à défaut ne pas modifier les fichiers d'installation se trouvant dans le dossier d'installation de WAPT.

Description des menaces pesant sur les biens sensibles WAPT

Par définition, une menace est une action ou un événement susceptible de porter préjudice à la sécurité globale de la machine équipée de WAPT.

Les agents menaçants à considérer pour l'évaluation de sécurité sont les suivants :

- **Entités non-autorisées** : il s'agit d'un attaquant humain ou d'une entité qui interagit avec WAPT mais qui ne dispose pas d'un accès légitime à celui-ci.

Note: Les *Administrateurs* et les *Administrateurs Locaux* ne sont pas considérés comme des attaquants.

Les menaces qui portent sur les biens sensibles WAPT définis ci-dessus sont les suivantes :

Menace M1 : Installation d'un logiciel malveillant par une entité non-autorisée

Cette menace correspond à un attaquant qui parviendrait à utiliser une composante de l'agent WAPT pour installer une application malveillante de façon pérenne, ou pour désinstaller ou désactiver une composante de sécurité du poste sur lequel l'agent WAPT est installé.

Menace M2 : Altération de valeurs de configuration par une entité non-autorisée

Cette menace correspond à un attaquant qui parviendrait à modifier ou à supprimer le paramétrage d'un élément de WAPT défini par un *Administrateur* légitime de WAPT.

Menace M3 : Accès illégitime par une entité non-autorisée

Cette menace correspond à un attaquant qui parviendrait à récupérer les données d'authentification d'un *Administrateur*, à contourner le mécanisme d'authentification de manière à accéder ou à altérer un bien sensible stocké sur le serveur. Elle correspond également à un attaquant qui parviendrait à se faire passer pour un agent WAPT.

Menace M4 : Écoute du réseau par une entité non-autorisée

Cette menace correspond à un attaquant qui parviendrait à intercepter et prendre connaissance des communications réseaux entre les agents et le serveur hébergeant WAPT.

Menace M5 : Altération du trafic réseau par une entité non-autorisée (Type *Man In the Middle*)

Cette menace correspond à un attaquant qui parviendrait à modifier les communications réseaux entre les agents et le serveur hébergeant WAPT ou les communications réseau entre la console et le serveur WAPT.

Description des fonctions de sécurité de WAPT

Par définition, les fonctions de sécurité sont l'ensemble des mesures techniques et mécanismes mis en œuvre pour protéger de façon proportionnée les biens sensibles contre les menaces identifiées.

Fonction de sécurité F1 : Authentification des contrôles d'accès

Fonction de sécurité F1A : Authentification d'une machine lors de son enregistrement initial dans la base de données WAPT

Nouveau dans la version 1.5.

Note: risques traités :

- éviter l'inscription d'une machine illégitime dans la base de données ;
- éviter une attaque par déni de service par surcharge de la base de données ;
- éviter l'enregistrement d'un inventaire falsifié dans la base de données ;

Solution mise en place

Pour exister dans la base de données et ainsi apparaître dans la console WAPT, une machine doit s'enregistrer auprès du serveur WAPT avec une commande **`register`**.

La commande **`register`** peut être exécutée automatiquement lors de l'installation ou de la mise à jour de l'agent WAPT si la machine est correctement enregistrée avec un compte machine Kerberos dans le domaine Active Directory de l'*Organisation*.

Si la machine ne présente pas au serveur WAPT un ticket Kerberos valide pour le serveur WAPT, alors la commande **`register`** échoue.

Note: La méthode avec Kerberos assume que le serveur Active Directory répond au moment du **`register`**.

Fonction de sécurité F1B : Vérification des certificats HTTPS du serveur par les clients WAPT

Nouveau dans la version 1.5.

Note: risques traités (notamment MITM) :

- éviter l'envoi d'informations sensibles à un serveur WAPT illégitime et non-autorisé ;
- éviter la récupération d'informations sensibles par une entité non-autorisée ;
- éviter l'affichage d'informations falsifiées dans la console de l'*Administrateur* ;
- éviter qu'une mauvaise date soit envoyée lors d'une requête HEAD (demande de modification de date de fichier) empêchant les futures mise à jours ;
- éviter d'envoyer le mot de passe de la console à un serveur WAPT illégitime et non-autorisé ;

Solution mise en place

Pour fonctionner correctement en version sécurisée :

- une option de vérification du certificat HTTPS serveur est introduite dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini` des agents WAPT qui **force la vérification du certificat serveur par les agents WAPT** ;
- une option de vérification du certificat HTTPS serveur est introduite dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini` des agents WAPT qui **force la vérification du certificat serveur par les agents WAPT** ;

L'implémentation technique peut être basée sur deux méthodes :

- utiliser un utilitaire de vérification de certificat implémenté dans la configuration du service **`Nginx`** du serveur WAPT ; cette méthode est généralement fournie par une *Autorité de Certification* validée pour votre réseau ;
- utiliser la méthode d'*épinglage de certificat* qui consiste a fournir à l'agent WAPT une liste de certificats de confiance qui sera stockée et maintenue dans le dossier `C:\Program Files (x86)\wapt\ssl\server` ;

Fonction de sécurité F1C : Aucun port n'écoute sur les agents WAPT

Nouveau dans la version 1.5.

Note: risques traités :

- éviter qu'une entité non-autorisée utilise un port ouvert à mauvais escient ;

Solution mise en place

Les connexions vers le serveur WAPT sont exclusivement initiées pas les clients, et les différentes actions instantanées (**`update`** / **`upgrade`** / **`install`** ...) passent au travers d'une connexion permanente par une Websocket initiée par l'agent WAPT.

Note: si HTTPS est activé, l'agent vérifie que la Websocket s'établit bien avec le bon serveur.

Fonction de sécurité F1D : Signature des remontées d'inventaire

Nouveau dans la version 1.3.12.13.

Note: risques traités :

- éviter qu'une entité non-autorisée envoie un inventaire falsifié d'une machine existante dans la base de données WAPT ;

Solution mise en place

- Au premier **`register`**, chaque machine crée un couple clé privée / certificat public dans le répertoire `C:\Program Files (x86)\wapt\private` accessible en lecture uniquement aux *Administrateurs Locaux*. Une fois la machine enregistrée, la clé publique est envoyée au serveur WAPT ;
- Lors d'une mise à jour de l'inventaire, le nouvel inventaire est envoyé signé avec la clé privée de la machine et déchiffré par la clé publique enregistrée dans la base de données ;
- Le serveur refusera de valider tout inventaire signé avec une mauvaise clé ;

Fonction de sécurité F1E : Vérification des droits avant l'exécution de certaines actions WAPT

Note: risques traités :

- éviter l'exécution de tâches sensibles par des entités non-autorisées ;

Solution mise en place

Les *Utilisateurs* interagissent avec WAPT au travers des interfaces WAPT (**`wapt-get`** en ligne de commande, **`wapttray`**, **`waptexit`**, **`waptselfservice`**).

Les interfaces peuvent ensuite déléguer l'exécution des tâches souhaitées au service WAPT local fonctionnant en compte système.

Les actions qui enclenchent des modifications listées ci-dessous ne nécessitent pas d'authentification auprès du service WAPT :

- `wapt-get update` (mettre à jour la liste des paquets disponibles) ;
- `wapt-get upgrade` (lancer l'installation des mises à jour en attente) ;
- `wapt-get download-upgrade` (télécharger les mises à jour en attente) ;

- `wapt-get clean` (supprimer des paquets restés en cache après installation) ;
- stopper n'importe quelle tâche WAPT en cours ;
- stopper / relancer le service WAPT ;

Les autres actions nécessitent que l'*Utilisateur* s'authentifie et que son compte appartienne au groupe de sécurité Active Directory **waptselfservice** ou que l'*Utilisateur* soit *Administrateur Local*, exemple d'action :

- `wapt-get install` : ordonner à l'agent WAPT d'installer un paquet WAPT marqué **MISSING** sur la machine ;
- `wapt-get remove` : ordonner à l'agent WAPT de supprimer un paquet WAPT ;
- `wapt-get forget` : ordonner à l'agent WAPT d'oublier l'existence d'un paquet WAPT installé sur la machine sans le désinstaller ;

Fonction de sécurité F2 : Protection de l'intégrité du processus d'installation des paquets WAPT

Fonction de sécurité F2A : Signature des paquets WAPT

Note: risques traités :

- éviter qu'une entité non-autorisée modifie le contenu ou le comportement d'un paquet WAPT ;

Solution mise en place

- quand un *Administrateur* ou un *Déployeur de Paquets* construit un paquet WAPT, un fichier `WAPTmanifest.sha256` est créé qui liste les sommes de contrôle de tous les fichiers du paquet ;
- a file `signature.sha256` **encrypted** with the WAPT agent's private key is then created in the folder `WAPT`; it contains the control sum of the file `manifest.sha256`;
- l'ensemble est archivé avec l'extension *.wapt* ;
- when a WAPT agent downloads a WAPT package, the agent checks that the file `signature.sha256` has been signed with the private key that matches the certificate present in the folder `WAPT`;
- l'agent WAPT vérifie ensuite que le certificat (ou la chaîne de certificat) `certificate.crt` a bien été signé avec une clé privée correspondant à un des certificats présents dans le dossier `C:\Program Files (x86)\wapt\ssl` ;
- the WAPT agent then generates the control sum of all the files contained in the package (except the files `signature.sha256` and `certificate.crt`) and verifies that it matches the file `manifest.sha256` contained in the package;
- si l'une de ces étapes n'est pas validée, alors cela signifie qu'un fichier a été modifié / ajouté / supprimé. Alors, l'exécution du **`setup.py`** est annulée ;
- le paquet en défaut est ensuite supprimé du cache local ; l'évènement est rapporté dans les log de l'agent ;

Note: Depuis la version 1.5, le fichier `manifest` est passée de *sha1* à *sha256*.

Fonction de sécurité F2B : Signature des attributs du fichier *control*

Nouveau dans la version 1.4.

Note: risques traités :

- éviter qu'une entité non-autorisée modifie des dépendances WAPT sur la machine en falsifiant le fichier `https://waptserver/wapt/Packages` ;

Solution mise en place

Lors de la signature d'un paquet WAPT, les attributs sensibles du paquet sont listés dans l'attribut **signed_attributes**.

Note: Exemple d'une liste *signed_attributes* :

package, *version*, *architecture*, *section*, *priority*, *maintainer*, *description*, *depends*, *conflicts*, *maturity*, *locale*, *min_os_version*, *max_os_version*, *min_wapt_version*, *sources*, *installed_size*, *signer*, *signer_fingerprint*, *signature_date*, *signed_attributes*,

Les attributs listés dans *signed_attributes* sont signés avec la clé privée de l'*Administrateur* et la signature est stockée dans l'attribut *signature* du fichier `control`.

The certificate matching the private key is stored in `WAPT\certificate.crt` inside the WAPT package.

Sur le serveur WAPT, lors de l'opération **`wapt-scanpackages`** (déclenchée par un ajout ou suppression de paquet), l'index `Packages` des paquets est régénéré.

Le serveur WAPT extrait de chaque paquet le certificat du signataire et l'ajoute dans le fichier ZIP `Packages`, dans le répertoire `ssl`. Chaque certificat est nommé avec sa fingerprint encodée en hexadécimal.

Lorsque le client WAPT effectue un **`update`** (mise à jour des paquets disponibles), il télécharge le fichier index `Packages`, qui contient à la fois les attributs signés de tous les paquets et les certificats des signataires.

Si le certificat du signataire des attributs d'un paquet est approuvé (ce qui signifie que ce certificat est signé par une *Autorité de Certification* ou que le certificat lui-méme est de confiance), **ET** que le certificat du signataire peut vérifier la signature des attributs, le paquet est ajouté à l'index des paquets disponibles, sinon il est ignoré.

Fonction de sécurité F2C : Restriction d'accès au répertoire d'installation de l'agent WAPT

Note: risques traités :

- éviter qu'une entité non-autorisée modifie le comportement de l'agent WAPT ;

Le répertoire d'installation `C:\Program Files (x86)\wapt` est accessible en lecture et modification :

- aux *Administrateurs Locaux* à travers un accès local au répertoire d'installation de l'agent WAPT ;
- aux *Administrateurs* à travers le mécanisme de déploiement des mises à jour de l'agent WAPT ;

Ni les *Déployeurs de Paquets*, ni les *Utilisateurs* n'ont d'accès en écriture au répertoire d'installation de l'agent WAPT.

Fonction de sécurité F2D : Restriction totale d'accès au répertoire de stockage du couple clé privé / certificat de signature d'inventaire

Note: risques traités :

- éviter qu'une entité non-autorisée falsifie une remontée d'inventaire ;
- éviter qu'une entité non-autorisée usurpe l'identité d'une machine ;

No access right is granted to any *User* to `C:\Program Files (x86)\wapt\private`, whomever he may be. Only the WAPT agent has a write and read access to this folder.

Note: Le stockage du couple clé privée / certificat découle d'un choix technique qui consiste à dire que la machine détient seule toutes les informations qui la concernent.

Fonction de sécurité F3 : Sécurisation des communications entre les différents composants WAPT

Fonction de sécurité F3A : Signature des requêtes envoyées aux agents WAPT

Nouveau dans la version 1.5.

Note: risques traités :

- éviter qu'une entité non-autorisée envoie des requêtes falsifiées aux agents WAPT ;

Solution mise en place

Les commandes ci-dessous sont signées par le serveur WAPT avant d'être envoyées au travers de la Websocket à l'agent WAPT destinataire de la commande :

- `wapt-get install` : ordonner à l'agent WAPT d'installer un paquet WAPT marqué **MISSING** sur la machine ;
- `wapt-get remove` : ordonner à l'agent WAPT de supprimer un paquet WAPT ;
- `wapt-get forget` : ordonner à l'agent WAPT d'oublier l'existence d'un paquet WAPT installé sur la machine sans le désinstaller ;
- `wapt-get update-status` : ordonner à l'agent WAPT de renvoyer l'état de son inventaire actuel au serveur WAPT ;
- `wapt-get upgrade` : ordonner à l'agent WAPT d'exécuter les paquets marqués **NEED UPGRADE** ;
- `wapt-get update` : ordonner à l'agent WAPT de mettre à jour la liste des paquets disponibles ;

L'ensemble des attributs de cette requête sont signés :

- l'*UUID* du poste ;
- l'action (ex: install) ;
- les arguments (ex: tis-firefox) ;
- l'horodatage des demandes ;

Le certificate associé à la signature est également passé :

- à la réception d'une requête par l'agent WAPT, il vérifie que la requête est correctement signée ;
- il vérifie ensuite que la date fournie en argument ne dépasse pas une 1 minute de décalage ;
- il vérifiera enfin que le certificat associé est autorisé à lancer des commandes ;

6.2.2 Présentation des processus cryptographiques

Date	janv. 10, 2023
Rédacteur	Hubert TOUVET
Applicable pour WAPT	>= 1.5.0.17
Version du document	1.5.0.17-0

Les processus cryptographiques sont utilisés dans les activités suivantes :

- signature et vérification des **fichiers d'un paquet** ;
- signature et vérification des **attributs d'un paquet** ;
- signature et vérification des **actions immédiates** sur les client WAPT ;
- signature des inventaires et **statut des clients WAPT** ;
- authentification de la connexion Websockets du client WAPT sur le serveur ;
- communication https entre les clients WAPT et le serveur WAPT ;
- communication https entre la console WAPT et le serveur WAPT ;
- communication https entre les clients WAPT et les dépôts WAPT ;

Répertoires et fichiers référencés dans ce document

- `<WAPT>` : répertoire d'installation de WAPT. Par défaut `%Program Files (x86)%WAPT` ;
- `<WAPT>wapt-get.ini` : fichier de configuration du client WAPT (**`wapt-get`** et **`waptservice`**) ;
- `<WAPT>ssl` : répertoire par défaut pour les certificats de confiance des paquets et actions ;
- `<WAPT>sslserver` : répertoire par défaut pour stocker les certificats https du serveur (pinning) ;
- `<WAPT>\private` : répertoire par défaut pour les certificats permettant de signer l'inventaire et les connexions Websocket ;
- `%LOCALAPPDATA%waptconsolewaptconsole.ini` : fichier de configuration de la console et des actions de développement de l'outil **`wapt-get`** ;
- `%appdata%waptconsolessl` : répertoire par défaut pour les certificats de confiance pour l'import de paquets depuis un dépot externe (c.à.d. les modèles de paquets) ;

Acteurs

- **Organisation**

  il s'agit du périmètre de responsabilité dans lequel est exploitée la solution WAPT ;

- **Autorité de Certification**

  elle détient les clés qui ont signé les certificats des *Déployeurs de Paquets*, et des serveurs HTTPS ;

- **Administrateurs**

  ils sont en possession d'une clé RSA personnelle et d'un certificat signé par l'"*Autorité de Certification* de l'"*Organisation* ; ils ont aussi un identifiant et un mot de passe pour accéder à la console WAPT ;

- **Postes clients WAPT**

  PC sous Windows à administrer pour lesquels les *Administrateurs* ont reçu l'accréditation de la part de l'"*Organisation*. Les postes clients sont joints au domaine Active Directory de l'"*Organisation*;

- **Dépôts WAPT internes**

  il s'agit d'un ou de plusieurs serveur Linux / Nginx qui diffusent aux Postes clients WAPT en HTTPS des paquets WAPT signés ;

- **Serveur WAPT**

  il s'agit d'un serveur Linux / Nginx / PostgreSQL / WAPT de l'"*Organisation* qui gère l'inventaire et le statut des postes clients WAPT.

  Par défaut, le serveur WAPT joue également le rôle de dépôt WAPT interne. Le serveur WAPT a un compte ordinateur dans l'Active Directory de l'"*Organisation*.

- **Dépôts WAPT externes**

  il s'agit de dépôts WAPT publics que les *Déployeurs de Paquets* peuvent utiliser pour importer des paquets conçus par d'autres *Organisations*, sous condition d'en vérifier l'adéquation aux normes internes de sûreté et de sécurité ;

- **Serveur Active Directory**

  serveur gérant le domaine de l'"*Organisation* ;

Synthèse des modules crypto mis en oeuvre par la solution WAPT

Coté client WAPT (WAPT 1.5.0.12) :

- module *ssl* standard de **Python 2.7.13** linké sur **`OpenSSL 1.0.2j 26 Sep 2016`** pour les connexions https entre les clients WAPT et serveur WAPT ;
- **cryptography==1.9** linké sur **`openssl 1.1.0f`** pour toutes les opérations crypto RSA, génération de clés, de certificat X509, de signature et vérification ;
- **kerberos-sspi==0.2** et **requests-kerberos==0.11.0** pour l'authentification du client WAPT lors de son enregistrement initial sur le serveur ;
- **pyOpenSSL==17.0.0** : pour récupérer la chaîne de certificats du serveur WAPT ;
- **certifi==2017.4.17** : base de certificats d'autorité racine ;
- **dll Openssl 1.0.2l** pour la partie waptcommon.pas écrite avec la bibliothèque FPC Indy et la classe TIdSSLIOHandlerSocketOpenSSL ;

Coté serveur WAPT :

- **nginx/1.10.2**: configurée pour TLS1.2, chiffre "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
- module *ssl* standard de **python 2.7.5** linké sur **`OpenSSL 1.0.1e-fips 11 Feb 2013`** ;
- **cryptography==1.9** linké sur **`OpenSSL 1.0.1e-fips 11 Feb 2013`** pour toutes les opérations crypto RSA, X509, signature et vérification ;

Gestion des clés et des certificat de l'Administrateur

Contexte

Les paquets et actions de l'Administrateur sont signés pour n'autoriser que les *Administrateurs* de confiance à intervenir sur les postes.

L'"*Administrateur* de la solution WAPT a en sa possession :

- une clé privée *RSA* de 2048 bits chiffrée par l'algorithme aes-256-cbc ;
- un certificat *X509* signé par une *Autorité de Certification* approuvée par l'"*Organisation* ;

Note: Le processus d'émission de ces clés, la signature du certificat, la distribution et la révocation sont à la charge de l'"*Organisation* utilisant WAPT et sortent donc du périmètre fonctionnel de WAPT.

Cependant, pour facilement tester la solution, WAPT propose une fonction pour générer une clé RSA et un certificat X509 :

- la clé RSA générée est de 2048 bits, chiffrée par l'algorithme aes-256-cbc et encodée en format PEM avec l'extension *.pem* ;
- le certificat est soit autosigné, soit signé par une autorité dont on a à disposition la clé et le certificat en format PEM ;
- si le certificat est autosigné, son attribut *KeyUsage* comporte le flag *keyCertSign* ;
- si l'"*Administrateur* est habilité par l'"*Organisation* à signer des paquets contenant du code python (présence du fichier `setup.py`), l'attribut du certificat *extendedKeyUsage* comporte le flag **CodeSigning** ;
- le certificat *X509* est encodé et remis à l'"*Administrateur* en format PEM avec l'extension *.crt* ;

Validité du certificat de l'Administrateur

Jusqu'à la version 1.5.0.12 incluse, Le client WAPT ne gère pas la vérification de la révocation du certificat de l'"*Administrateur* lors du processus de vérifcation des paquets, attributs et actions de l'"*Administrateur*.

Il ne vérifie que les dates de validité (attributs *notValidBefore* / *notValidAfter*). Le certificat est valide si (**Now** >= *notValidBefore* et **Now** <= *notValidAfter*).

Autoriser le certificat de l'Administrateur à signer un paquet

Le certificat utilisé par la console WAPT pour signer les paquets et actions est défini avec le paramètre *personal_certificate_path* de la section `[global]` du fichier `%LOCALAPPDATA%waptconsolewaptconsole.ini`.

WAPT demande à l'*Administrateur* son mot de passe pour permettre de rechercher la clé privée (encodée au format PEM) correspondant au certificat parmi les fichiers `.pem` du répertoire contenant les certificats.

Lors de la signature de paquet, WAPT refusera le certificat si le paquet contient un fichier `setup.py` et que le certificat n'est pas de type *CodeSigning*.

Gérer les clés et certificats du Client WAPT

Contexte

Le client WAPT (**`waptservice`**) utilise des clés RSA et un certificat X509 pour interagir avec le serveur WAPT.

Le certificat du client WAPT est utilisé dans les situations suivantes :

- lors de la mise à jour du statut du poste sur le serveur. (:command:*update_server_status*): **signature des informations**;
- lors de la connexion Websocket du poste vers le serveur (**`waptservice`**) : **signature de l'UUID du poste** ;

Émission initiale et mise à jour du certificat du client WAPT

- à l'issue du processus d'installation de l'agent WAPT sur le poste client, l'agent WAPT s'enregistre automatiquement auprès du serveur WAPT en émettant une requête https authentifiée par Kerberos qui utilise le TGT du compte machine ;

  L'agent WAPT utilise les API Kerberos de Windows en s'appuyant sur les modules python **`kerberos-sspi`** et **`requests-kerberos`** ;

Note: Cette procédure fonctionne si et seulement si le Poste client est joint au domaine Windows pour lequel le serveur WAPT est configuré.

- si la clé et les certificats n'ont pas encore été générés, ou s'ils ne correspondent pas au *FQDN* actuel de la machine, l'agent WAPT génère une clé RSA et un certificat X509 autosigné avec les paramètres suivants :
  - la clé est de type RSA 2048 bits encodée en PEM et stockée dans le fichier `<WAPT>\private\<fqdn du poste>.pem` ;
  - le certificat généré a les attributs suivants :
    - *Subject.COMMON_NAME* = <device FQDN>;
    - *Subject.ORGANIZATIONAL_UNIT_NAME* = nom de l'*Organisation* de la machine tel qu'enregistré dans la base de registre ;
    - *SubjectAlternativeName.DNSName* = <device FQDN> ;
    - *BasicConstraint.CA* = True ;
    - *validity* = 10 ans ;
    - *serialnumber* = aléatoire ;
  - le certificat est sauvegardé dans le fichier `<WAPT>private<device FQDN>.crt` ;

Note: Seuls le compte machine et les *Administrateurs Locaux* ont accès au répertoire `<WAPT>\private` car des ACL spécifiques sont appliquées à l'installation de l'agent WAPT sur le poste.

- l'inventaire ou les mises à jour de status du client sont envoyés au serveur WAPT par requête https POST ;
- on authentifie la requête https POST en ajoutant deux headers http spécifiques :
  - *X-Signature* :
    - encodage en JSON des informations d'inventaire ou de status ;
    - signature du json avec la clé privée du client WAPT : hachage *sha256* et padding *PKCS#1 v1.5* ;
    - encodage de la signature en *base64* ;
  - *X-Signer*: *Subject.COMMON_NAME* ou UUID du client WAPT.
- après avoir initialement authentifié le client WAPT avec Kerberos, le serveur reçoit le certificat envoyé par le client et il le stocke dans son inventaire, dans la table *hosts* (champ *host_certificate* en format PEM).

Note: Si le poste client WAPT est renommé, la paire de clés et le certificat sont recréés.

Lors de la tentative de mise à jour de status du client vers le serveur, la requête POST sera refusée, car la machine est enregistrée dans la base de données avec un autre certificat.

La machine tentera alors de se ré-enregistrer (**`register`**) avec authentification kerberos ; ainsi le nouveau certificat sera enregistré dans la base de données.

Déployer les certificats d'autorité pour vérifier les paquets et les actions sur les clients

Les certificats au format PEM stockés dans des fichiers avec l'extension .crt ou .pem et présents dans le répertoire défini par le paramètre *public_certs_dir* du fichier `<WAPT>wapt-get.ini` sont réputés **de confiance**.

Ce paramètre *public_certs_dir* est initialisé par défaut à `<WAPT>ssl`.

Le déploiement de ces certificats d'autorité est effectué lors de l'installation initiale de l'agent WAPT par l'installeur.

Depuis la console, l'*Administrateur* compile un installeur personnalisé en vue de son déploiement par *GPO* sur les Postes clients.

La console WAPT incorpore dans cet installeur les certificats présents dans le répertoire `<WAPT>\ssl` du poste depuis lequel l'installeur est compilé.

L'*Administrateur* doit s'assurer d'enregistrer dans `<WAPT>\ssl` uniquement les certificats d'autorité nécessaires avant de lancer la compilation de l'agent.

Le déploiement ou la mise à jour de certificats de l'*Autorité de Certification* pour la vérification des paquets et actions peuvent être également assurés à postériori par une GPO Active Directory ou par un paquet WAPT.

Déployer les certificats d'autorité pour la communication HTTPS entre les clients WAPT et le serveur WAPT

Le service WAPT ainsi que l'outil en ligne de commande **wapt-get** communiquent avec le serveur WAPT pour envoyer l'inventaire (**register**) et le statut de déploiement des paquets (**update-status**).

Ces deux types de connexions vérifient le certificat https du serveur.

Paramètre *verify_cert* de la section `[global]` du fichier `<WAPT>wapt-get.ini` :

- *verify_cert* = 1

  Vérifie le certificat du serveur https en utilisant le bundle fourni par le module **certifi**. Ne fonctionnera bien que si le serveur https est configuré pour renvoyer son certificat et les certificats intermédiaires à l'initialisation de la connexion TLS ;

- *verify_cert* = <chemin vers fichier .pem>

  Vérifie le certificat du serveur https en utilisant le bundle de certificats indiqué. Tous les certificats de CA intermédiaires et root doivent être rassemblés dans un fichier au format *.pem* ;

- *verify_cert* = 0

  Ne pas vérifier le certificat du serveur https ;

Conventionnellement, on stocke le bundle de l'*Autorité de Certification* approuvées dans le répertoire `<WAPT>sslserver`.

La console WAPT comporte une fonction pour faciliter la récupération initiale de la chaîne de certificats du serveur et pour la stocker au format .pem dans le fichier `<WAPT>sslserver<FQDN serveur>.pem`.

Il est de la responsabilité de l'*Administrateur* de s'assurer que la chaîne ainsi récupérée est authentique.

Lors de la compilation de l'installeur de l'agent WAPT, les certificats ou le bundle de certificats sont intégrés dans l'installeur.

Lors du déploiement de l'installeur sur les clients WAPT, le bundle est copié dans `<WAPT>sslserver` et le paramètre *verify_cert* de la section `[global]` du fichier `<WAPT>wapt-get.ini` est renseigné pour désigner le bundle.

Communications HTTPS entre les clients WAPT et les dépôts WAPT

Déployer des certificats d'autorité

Les connexions HTTPS de l'agent WAPT vers le dépôt principal utilisent les mêmes méthodes que les communications entre l'agent WAPT et le serveur WAPT.

L'agent WAPT utilise le même bundle de certificats pour communiquer en HTTPS avec le dépôt principal, avec le serveur WAPT, et avec les dépôts secondaires.

Client WAPT

La connexion https est mise en œuvre par les modules python **requests**, **urllib3** et **ssl**.

Le certificat transmis par le serveur HTTPS du dépôt est vérifié par le module **urllib3.contrib.pysopenssl.PyOpenSSLContext** et **urllib3.util.ssl_wrap_socket**.

Communications Websockets entre les clients WAPT et le serveur WAPT

Pour permettre des actions immédiates sur les clients WAPT, le service WAPT déployé sur les clients tente d'établir et de maintenir une connexion Websocket vers le serveur WAPT.

Cette connexion s'effectue sur un connexion chiffrée avec le protocole TLS et utilise côté client le même bundle de certificat que la connexion HTTPS Client vers Serveur WAPT.

Communications entre la console WAPT et le serveur WAPT

Déployer des certificats d'autorité

Paramètre *verify_cert* de la section `[global]` du fichier `%LOCALAPPDATA%waptconsolewaptconsole.ini` :

- *verify_cert* = 1

  Vérifie le certificat du serveur https en utilisant le bundle fourni par le module **`certifi`**. Ne fonctionnera bien que si le serveur https est configuré pour renvoyer son certificat et les certificats intermédiaires à l'initialisation de la connexion TLS ;

- *verify_cert* = <chemin vers fichier .pem>

  Vérifie le certificat du serveur https en utilisant le bundle de certificats indiqué. Tous les certificats de CA intermédiaires et root doivent être rassemblés dans un fichier au format *.pem* ;

- *verify_cert* = 0

  Ne pas vérifier le certificat du serveur https ;

Conventionnellement, on stocke le bundle de l'*Autorité de Certification* approuvées dans le répertoire `<WAPT>sslserver`.

La console WAPT comporte une fonction pour faciliter la récupération initiale de la chaîne de certificats du serveur et la stocker au format *.pem* dans le fichier `<WAPT>sslserver<FQDN serveur>`.

Il est de la responsabilité de l'*Administrateur* de s'assurer que la chaîne ainsi récupérée est authentique.

Il est également possible de récupérer la chaîne de certificats du serveur et de renseigner le paramètre *verify_cert* avec la commande **`wapt-get enable-check-certificate`**.

Déployer des certificats d'autorité pour vérifier les paquets importés dans le dépôt local

Contexte

Dans la console WAPT / onglet *Dépôt privé*, un bouton *Importer depuis internet* permet de télécharger un paquet depuis un dépôt externe dont l'URL est fournie par le paramètre *repo_url* de la section `[wapt_templates]` du fichier `%LOCALAPPDATA%waptconsolewaptconsole.ini`.

Une case à cocher *Vérifier la signature de paquet* permet de s'assurer que le paquet est signé avec un certificat provenant d'une Autorité de confiance.

Certificats d'Autorités de confiance

Les certificats d'autorité présents dans le répertoire désigné par le paramètre *public_certs_dir* de la section `[wapt_templates]` du fichier `%LOCALAPPDATA%waptconsolewaptconsole.ini` sont réputés de confiance.

Si le paramètre n'est pas mentionné explicitement, il est initialisé à `%appdata%waptconsolessl`.

Ce répertoire n'est pas automatiquement rempli par WAPT. Il est de la responsabilité de l'*Administrateur* de copier/coller les fichiers PEM d'autres *Administrateurs* de confiance ou les certificats des Autorités de Certification de confiance.

Les certificats d'autorité sont encodés en format PEM et stockés dans des fichiers avec l'extension *.pem* ou *.crt*. On peut stocker plusieurs certificats dans chaque fichier `.crt` ou `.pem`.

Il n'est pas nécessaire d'avoir la chaîne complète de certificats, WAPT acceptera le signataire d'un paquet à partir du moment que :

- le certificat du paquet est également présent dans le répertoire *public_certs_dir*. Le test d'égalité est fait avec l'empreinte du certificat ;
- le certificat de l'Autorité ayant signé le certificat du paquet est présent dans le répertoire *public_certs_dir*. La recherche est faite avec l'attribut *issuer_subject_hash* du certificat. La signature du certificat est effectuée par la classe **`x509.verification.CertificateVerificationContext`** ;

Processus de signature d'un paquet

Le processus de signature du paquet est lancé lors des actions suivantes :

- action `wapt-get.exe build-upload <répertoire>` ;
- action `wapt-get.exe sign-package <chemin-fichier-paquet.wapt>` ;
- commande shell `wapt-signpackage.py <liste de fichiers paquets WAPT>` ;
- sauvegarde d'un paquet *host* dans la console WAPT ;
- sauvegarde d'un paquet *group* dans la console WAPT ;
- import direct d'un paquet depuis un dépôt externe ;
- wizard de création de paquets à partir de MSI et de setup ;

Paramètres initiaux

- fichier ZIP du paquet ;
- clé privée RSA du signataire encodée en format *.pem* et chiffrée (par l'algorithme *aes-256-cbc* de openssl si la clé a été créée dans la console WAPT) ;
- certificat *X509* du signataire correspondant à la clé privée ;
- si le paquet à signer contient un fichier `setup.py`, le certificat *X509* doit avoir l'extension *advanced Key Usage* : *codeSigning (1.3.6.1.5.5.7.3.3)* ;

Signature des attributs du fichier control

Le fichier `control` d'un paquet décrit les métadonnées du paquet, en particulier son nom, sa version, ses dépendances et ses conflits. C'est la fiche d'identité du paquet.

Ces métadonnées sont primitivement utilisées par l'agent WAPT pour déterminer si un paquet doit être mis à jour, et quels autres paquets doivent être installés ou désinstallés préalablement.

Ces informations sont donc signées pour garantir aux Postes client leur intégrité et leur authenticité.

Etapes du processus :

- les attributs *signed_attributes*, *signer*, *signature_date*, *signer_certificate* sont ajoutés à la structure du fichier `control` :
  - *signed_attributes* : liste des noms d'attributs avec séparateur virgule (,) ;
  - *signer* : commonName de l'objet du certificat du signataire ;
  - *signature_date* : date et heure en cours (UTC) sous la forme "%Y-%m-%dT%H:%M:%S" ;
  - *signer_fingerprint* : empreinte `sha256` du certificat encodée en hexadécimal obtenue par la fonction **`fingerprint`** de la classe **`cryptography.x509.Certificate`** ;
- les attributs de la structure control sont encodés en json ;
- le JSON BLOB (Binary Large Object) résultant est signé avec un hachage *sha256* et un remplissage *PKCS#1 v1.5* ;
- la signature est encodée en base64 et stockée dans le JSON dans l'attribut *signature* du fichier `control` ;

Signature des fichiers du paquet

- les attributs du fichier control sont signés et sérialisés en JSON. Le résultat est stocké dans le fichier `<WAPT>control` du ZIP du paquet ;
- le certificat X509 du signataire depuis le fichier est stocké dans le fichier `<WAPT>certificate.crt` du paquet WAPT ;
- les empreintes *sha256* de tous les fichiers contenus dans le paquet WAPT sont codées en hexadécimal et stockées sous forme de liste JSON [(nom de fichier, hachage),] dans le fichier `<WAPT>manifest.sha256` dans le paquet WAPT ;
- le contenu du fichier `<WAPT>manifest.sha256` est signé avec la clé privée de l'*Administrateur* (clé RAS 2048 bits), avec un hachage *sha256* et un remplissage *PKCS#1 v1.5* :
  - la procédure de signature fait appel à la fonction sign de la classe **`cryptography.rsa.RSAPrivateKey.signer`** ;
  - **`cryptography.rsa.RSAPrivateKey.signer`** repose sur les fonctions OpenSSL de **`EVP_DigestSignInit`** ;
- la signature est encodée en base64 et stockée dans le fichier `<WAPT>signature.sha256` du paquet WAPT ;

Vérifier la signature des attributs d'un paquet

Elle a lieu :

- lors de la mise à jour de l'index des paquets disponibles à partir de l'index `Packages` du dépôt ;
- lorsqu'une signature de paquet est vérifiée (installation, téléchargement) lorsqu'elle n'est pas en mode *développement*, c'est-à-dire si l'installation se fait à partir d'un fichier ZIP et non d'un répertoire de développement ;

Elle consiste à :

- lire les attributs du fichier control depuis le fichier `<WAPT>\control` du ZIP du paquet ;
- récupérer le certificat X509 du signataire depuis le fichier `<WAPT>\certificate.crt` du ZIP du paquet ;
- décoder l'attribut signature du control depuis le format base64 ;
- construire une structure JSON avec les attributs devant être signés (tels que définis dans la classe **`PackageEntry`**) ;
- vérifier si la clé publique du certificat du titulaire peut vérifier le hachage de la liste structurée des attributs JSON et la signature du fichier `control`, en utilisant le hachage *sha256* et le remplissage *PKCS#1 v1.5* ;
- vérifier si le certificat est de confiance (soit présent en tant que tel dans les certificats de confiance, soit signé par une *Autorité de Certification* de confiance) ;

Dans le cas où nous devons vérifier les attributs sans avoir le paquet WAPT à disposition, nous récupérons la liste des certificats des détenteurs potentiels de certificats à partir du fichier d'index `Packages` sur le dépôt WAPT. Les certificats sont nommés `ssl/<hexadecimal formated certificate fingerprint>.crt`.

Un attribut de la structure `control` du paquet indique l'empreinte du certificat du signataire du fichier `control`.

Vérifier la signature d'un paquet

Elle a lieu :

- lors de l'installation d'un paquet sur un Poste client ;
- lors de l'édition d'un paquet existant ;
- lors de l'import d'un paquet depuis un dépôt externe (si option cochée dans la console) ;

Elle consiste à :

- récupérer le certificat X509 du signataire depuis le fichier `<WAPT>\certificate.crt` du ZIP du paquet ;
- vérifier que le certificat a été signé par une autorité de confiance dont le certificat est présent dans le fichier `ssl` du client WAPT ;
- vérifier la signature du fichier `<WAPT>\manifest.sha256` avec la clé publique ;

Signature d'une action immédiate

Contexte

Depuis la console, l'*Administrateur* peut déclencher des actions directes sur le client WAPT, s'il est connecté au serveur par le mode Websockets.

La console WAPT signe ces actions avec la clé et le certificat du *Administrateur* avant de les envoyer au serveur WAPT en utilisant une requête HTTPS POST ; la requête est ensuite transmise aux clients WAPT ciblés.

Les actions possibles sont :

- **`trigger_host_update`** ;
- **`trigger_host_upgrade`** ;
- **`trigger_install_packages`** ;
- **`trigger_remove_packages`** ;
- **`trigger_forget_packages`** ;
- **`trigger_cancel_all_tasks`** ;
- **`trigger_host_register`** ;

Processus de signature

- l'action est définie par son nom et des attributs dépendants de l'action. Les attributs sont *uuid*, *action*, *force*, *notify_server*, et *packages* (pour les actions impliquant une liste de paquets) ;
- les attributs *signed_attributes*, *signer*, *signature_date*, *signer_certificate* sont ajoutés à la structure de l'action :
  - *signed_attributes* : liste des noms des attributs qui sont signés ;
  - *signer* : commonName de l'objet du certificat du signataire ;
  - *signature_date* : date et heure en cours (UTC) sous la forme "%Y-%m-%dT%H:%M:%S" ;
  - *signer_certificate* : certificat `X509` du signataire encodé en base64 ;
- la structure est encodée en JSON ;
- la signature du JSON est calculée à partir de la clé privée RSA du signataire en utilisant un algorithme de hachage *sha256* et un remplissage *PKCS1 v1.5* ;
- la signature est encodée en base64 et stockée dans le JSON dans l'attribut *signature* ;

Vérifier la signature d'une action immédiate

Contexte

Depuis la console, l'*Administrateur* peut déclencher des actions directes sur le client WAPT, s'il est connecté au serveur par le mode Websockets.

Les actions sont encodées en JSON, signées avec la clé et le certificat de l'*Administrateur* et relayées vers le client WAPT visé par le serveur WAPT.

Les actions possibles sont :

- **trigger_host_update** ;
- **trigger_host_upgrade** ;
- **trigger_install_packages** ;
- **trigger_remove_packages** ;
- **trigger_forget_packages** ;
- **trigger_cancel_all_tasks** ;
- **trigger_host_register** ;

L'action **get_tasks_status** ne demande pas d'authentification ssl.

Processus de vérification

Sur réception d'un évènement par la connexion Websocket du client WAPT :

- le certificat X509 du signataire de l'action est extrait du json (format PEM) ;
- le client WAPT teste si le certificat est un certificat de confiance, c'est-à-dire présent dans `<WAPT>ssl` ou signé par une autorité de confiance (certificat de l'autorité présent dans `<WAPT>\ssl` ;
- le client WAPT teste si le certificat peut vérifier la signature présente dans la structure JSON de l'action ce qui consiste en :
  - extraire la signature encodée en base64 dans le json depuis l'attribut *signature* ;
  - extraire la date de signature formatée sous la forme "%Y-%m-%dT%H:%M:%S" depuis l'attribut *signature_date* ;
  - vérifier que la date de signature n'est pas trop ancienne ou dans le futur de plus de 10 minutes ;
  - reconstruire une représentation json des attributs de l'action ;
  - vérifier que la clé publique du certificat peut vérifier le JSON avec la signature en utilisant un algorithme de hachage *sha256* et un remplissage *PKCS1 v1.5* ;

Vérification du téléchargement complet d'un paquet

Pour chaque paquet, une somme *md5* du paquet est calculée et disponible dans l'index `Packages` du dépôt.

Lors de l'installation d'un paquet, le client vérifie si le paquet est déjà disponible localement dans le répertoire `<WAPT>\cache`.

Si le fichier est présent, sa somme *md5* est comparée avec la somme *md5* présente dans l'index. Si elles diffèrent, le paquet en cache local est effacé.

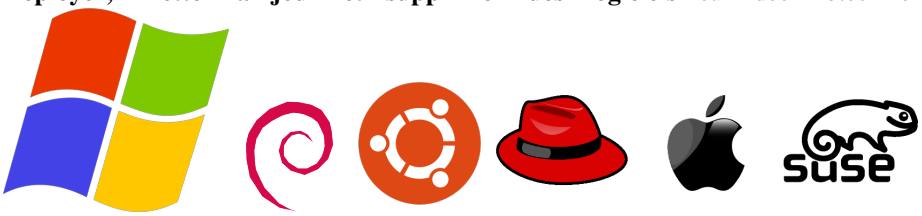
Cette somme *md5* ne sert qu'à s'assurer qu'un paquet a été téléchargé complètement.

La vérification de la signature du paquet permettra d'être effectivement assuré de l'intégrité et de l'authenticité du paquet.

6.3 Comparaison des caractéristiques entre les versions WAPT Enterprise et Community

6.3.1 Liste des caractéristiques actuelles en date du 2023-01-10

Table1: Comparaison des caractéristiques entre les versions WAPT Enterprise et Community en date du 2023-01-10

Caractéristique	Enterprise	Community
Déployez, mettez à jour et supprimez des logiciels sur des hôtes fonctionnant sous 	✓	✓
Le dépôt est hébergé sur une infrastructure sous votre contrôle et non sous le contrôle d'une grande MegaCorp, Inc. qui privilégiera ses propres intérêts plutôt que les vôtres	✓	✓
Déployez et mettez à jour les **configurations dans le contexte du SYSTÈME**	✓	✓
Déployez et mettez à jour les **configurations dans le contexte de l'UTILISATEUR**	✓	✓
Obtenez un **inventaire complet** du matériel, des logiciels et des progiciels appliqués avec WAPT	✓	✓
Bénéficiez du **libre-service différencié** (les utilisateurs autorisés peuvent installer les logiciels autorisés à partir de magasins de paquets WAPT autorisés)	✓	✗
Bénéficiez de **Mises à jour Windows simplifiées** qui fonctionnent bien mieux qu'un WSUS standard (seuls les KB requis sont téléchargés depuis Microsoft)	✓	✗
Simplifiez et structurez votre charge de travail administrative en appliquant des paquets WAPT à vos UO (Unités d'Organisation)	✓	✗
Configurez et gérez facilement les relais WAPT **pour préserver la bande passante** dans vos scénarios multi-sites	✓	✗
Accédez à des dépôts de **paquets WAPT prêts à être déployés** pour des logiciels communs libres ou gratuits	✓	✓
Travaillez avec des **recettes python facilement vérifiables** pour l'installation, la mise à jour et la suppression de logiciels et de configurations, les recettes peuvent intégrer du code Powershell ou des scripts réalisés avec d'autres langages (ex : pour personnaliser un logiciel en utilisant un répertoire LDAP)	✓	✓
Bénéficiez de **centaines de Helpers** pour simplifier le conditionnement de vos logiciels	✓[1]	✓
Chiffrez vos données sensibles pour le transport (clés de licence du logiciel, login, mot de passe, FQDN du serveur, informations API pour l'enregistrement du logiciel auprès du vendeur, etc)	✓	✗
Automatisez l'audit de vos configurations pour une conformité **facile, automatisée et toujours à jour**	✓	✗

6.3.2 Caractéristiques à venir prochainement

Vous trouverez ci-dessous une liste de fonctionnalités que nous avons identifiées comme étant réellement utiles pour WAPT et la communauté des utilisateurs de WAPT et sur lesquelles nous avons déjà commencé à travailler. Aucun calendrier n'est promis, restez à l'écoute, nous vous promettons seulement que nous travaillons très dur pour atteindre ces objectifs.

Caractéristique	Enterprise	Community
Mode multi-tenants, multi-clients avec ACL (Access Control Lists) pour les infogérants et les grandes organisations multi-départementales ou internationales utilisant un mécanisme interne basé sur une PKI (Public Key Infrastructure).	✓	✗
Partage d'écran simple à utiliser pour l'assistance aux utilisateurs, construit avec le même niveau de sécurité et de confidentialité que le WAPT	✓	✗
Historique des actions effectuées via WAPT pour un rapport complet du cycle de vie de la maintenance logicielle d'une machine	✓	✗
Authentification des Administrateurs WAPT à l'aide de jetons cryptographiques (ex : cartes à puce)	✓	✗
Accès à des paquets WAPT prêts à être déployés ou à des recettes de logiciels commerciaux sous licence (logiciels commerciaux communs pour l'industrie, le secteur médical, les bureaux, les collectivités publiques, la cybersécurité, etc.)	✓	✗
Accès à des extensions de paquets WAPT prêtes à être déployées pour simplifier le blindage des postes de travail à l'aide d'Applocker ou de ses équivalents.	✓	✗
Poursuite du support Windows XP avec WAPT pour les machines-outils d'usine, les équipements médicaux des hôpitaux, les instruments de recherche coûteux, etc	✓[4]	✗
Outil de déploiement d'images du système d'exploitation intégré au WAPT	✓	✗
Intégration d'un sous-ensemble utile de l'inventaire WAPT avec les outils populaires d"ITSM (IT Service Management) et déclenchement d'actions depuis la console ITSM des utilisateurs	✓	✗

[1] La version Enterprise intègre plus de fonctions SetupHelper que la version Community.
[2] Dans la version Community, le mot de passe WAPT SuperAdmin est partagé entre les personnes qui gèrent le serveur WAPT.
[3] Un volume minimal de licences doit être souscrit afin de bénéficier du support téléphonique de Tranquil IT pour l'exploitation quotidienne du logiciel. Une assistance supplémentaire payante est disponible pour vous aider à répondre à vos besoins en matière de packaging WAPT.
[4] Windows XP ne fonctionne pas avec Python > 2.7. Une branche spéciale de WAPT sera donc gelée avec la dernière version de l'agent WAPT fonctionnant avec 2.7. Cette version de l'agent sera bien sûr exclue de la cible d'évaluation lors des futures certifications de sécurité.

6.3.3 Résumé des principes de fonctionnement de WAPT

- WAPT est basé sur un agent qui n'autorise aucun port ouvert entrant dans les pare-feux des machine ; l'agent lance une *websocket* bidirectionnelle sécurisée avec le serveur pour permettre des remontées d'informations et des actions en temps réel ;
- Peut fonctionner avec des passerelles de rupture protocolaire en fonctionnant avec le principe des tâches planifiées ;
- Fonctionne sur le principe de l'extraction en douceur des mises à jour et de leur application à un moment opportun (fonctionne avec une bande passante faible / intermittente, une latence élevée, une forte gigue) ;
- Ne nécessite pas d'AD (fonctionne également avec l'édition Home de Windows), mais affichera l'hôte dans son arborescence Active Directory si l'hôte est joint à un AD ;
- Méthodes de déploiement de l'agent WAPT :
  1. en utilisant une GPO (Group Policy Object) ou un script Ansible ;
  2. manuellement après avoir téléchargé l'agent depuis le serveur WAPT ou en utilisant SSH (Secured Shell) ;
- Méthodes d'enregistrement des machines auprès du serveur WAPT :
  1. automatiquement, en utilisant le compte kerberos de la machine ;
  2. manuellement avec le login et le mot de passe WAPT Superadmin ;
- Les mises à niveau peuvent être déclenchées :
  1. lors de l'arrêt de la machine, c'est le mode standard ;
  2. par un Administrateur WAPT autorisé en cas d'urgence (ex : vulnérabilités critiques courant dans la nature) ;
  3. par l'utilisateur au moment qu'il choisit (ex : chariot de soins infirmiers fonctionnant 24h/24 et 7j/7 et mis à jour pendant la pause déjeuner avec un simple clic) ;
  4. via une tâche planifiée se déroulant à un moment prédéterminé (le mieux pour les serveurs) ;
- La sécurité est assurée avec :
  1. la signature des paquets WAPT en utilisant la cryptographie asymétrique ;
  2. l'authentification des machines par rapport au serveur WAPT en utilisant la cryptographie symétrique lors de l'enregistrement ;
  3. la confidentialité du serveur WAPT en utilisant des certificats clients déployés par WAPT ;

6.4 WAPT Community et WAPT Enterprise

6.4.1 Principaux ajouts fonctionnels de la version Enterprise

WAPT est conçu pour aider les administrateurs de systèmes informatiques à gérer le cycle de vie de leur base d'applications Windows, de pilotes, de systèmes d'exploitation et de configurations en contexte système et en contexte utilisateur.

Avec WAPT Enterprise, vous bénéficiez des fonctions de base disponibles dans WAPT pour déployer, mettre à jour et supprimer des logiciels et des configurations sur vos périphériques Windows à partir d'une console centralisée et simple à utiliser.

WAPT est un modèle *opencore* et la version **Enterprise** est basée sur la version **Community** de WAPT avec les améliorations suivantes conçues pour répondre aux besoins des moyennes et grandes organisations :

- **Authentification Active Directory** des développeurs de paquets WAPT, des déployeurs de paquets, des utilisateurs en libre-service et pour l'enregistrement initial des périphériques sur le serveur WAPT. De plus, l'affichage des périphériques dans la console de gestion WAPT suit la structure hiérarchique de vos UO Active Directory ;
- **Séparation des rôles entre les développeurs de paquets et les déployeurs de paquets**. De cette façon, les paquets WAPT peuvent être conçus par les équipes informatiques de l'entreprise pour suivre les directives de sécurité de l'entreprise et ils sont ensuite déployés par les équipes informatiques locales qui connaissent mieux les besoins de leurs utilisateurs. Une telle ségrégation est mise en œuvre au moyen de jeux de clés différenciés (c'est-à-dire des certificats SSL individuels à signature de code pour les développeurs de paquets et des certificats SSL standards pour les déployeurs de paquets) ;
- **Self-service différencié** WAPT Enterprise vous permet d'associer des listes de paquets autorisés à des groupes de sécurité Active Directory. Les utilisateurs autorisés peuvent installer par eux-mêmes des paquets provenant de vos dépôts approuvés sans avoir à soumettre de demande de support aux équipes IT.
- **WAPT WUA** WAPT permet de gérer les mises à jour Windows sur vos terminaux.
- **Reporting avancé pour les équipes de direction**. Ces rapports complètent les rapports opérationnels déjà disponibles dans la console de gestion ; ils sont conçus pour aider les développeurs et les exploitants de WAPT à démontrer leur efficacité grâce à leur usage de WAPT en éliminant la complexité et en sécurisant correctement leur parc de périphériques et d'applications ;
- **Configuration dynamique du dépôt**. À partir de WAPT 1.8, la réplication du dépôt peut être activée à l'aide d'un agent WAPT installé sur une machine existante, une *appliance* dédiée ou une machine virtuelle.

  Le rôle de réplication est déployé par le biais d'un paquet WAPT qui active le serveur web **`Nginx`** et configure le déclenchement des mises à jour, les types de paquets, la synchronisation des paquets, et bien plus encore.

  Cette fonctionnalité permet aux agents WAPT de trouver dynamiquement leur dépôt WAPT disponible le plus proche à partir d'une liste de règles stockées sur le serveur WAPT.

La version Enterprise de WAPT est particulièrement recommandable pour les Organisations :

- exploitant de vastes parcs (généralement au dessus de 300 machines) ;
- éclatées géographiquement avec de multiples agences ou sites de production ;
- nécessitant une traçabilité forte des actions réalisées sur le parc pour des raisons d'audit et de sécurité ;

6.4.2 Description des services disponibles dans le cadre du contrat WAPT Enterprise

Accès aux améliorations futures dans WAPT Enterprise

En souscrivant à WAPT Enterprise et en maintenant votre abonnement valide, vous bénéficierez bien entendu des futures améliorations apportées au coeur de WAPT et vous accéderez automatiquement aux futures extensions de la version Enterprise.

L'interruption de votre abonnement vous basculera automatiquement dans la version Community et les fonctions avancées de la version Enterprise ne seront plus accessibles dans votre installation de WAPT.

Assistance téléphonique pour votre utilisation quotidienne de WAPT

Quand votre abonnement atteint un certain volume, Tranquil IT, le créateur de WAPT, vous accorde un accès privilégié à son équipe centrale de développeurs et d'experts WAPT.

Nous mettons à votre disposition une hot-line téléphonique dédiée avec une réponse en direct pour satisfaire vos besoins de support.

Nous nous engageons à vous fournir des réponses fiables et pertinentes sur le périmètre souscrit, rapidement.

En souscrivant ou en renouvelant votre contrat WAPT Enterprise, vous recevrez une notification indiquant les modalités à suivre pour accéder à notre support.

Attention: L'assistance concerne uniquement le support sur le logiciel WAPT ; l'assistance pour l'adaptation, la personnalisation, la création de paquets WAPT peut être souscrite en plus sous forme de tickets d'assistance prépayés.

Jusqu'à deux personnes désignées dans votre organisation peuvent être inscrites pour communiquer avec notre support direct.

Prix et accès préférentiels à la formation WAPT

Vous pouvez choisir de former votre équipe informatique sur n'importe quel aspect de WAPT.

Les abonnés Entreprise bénéficient d'un accès privilégié aux conseillers en formation de Tranquil IT et d'une remise de 50 % par rapport aux tarifs standard de formation.

6.4.3 Pourquoi Tranquil IT affiche une limite à 300 postes pour WAPT Community dans ses brochures commerciales ?

Tranquil IT affiche une limite à 300 postes sur ses brochures commerciales pour la version Community de WAPT.

Note: Il n'y a **aucune restriction technique quant à l'usage de la version Community**, elle est librement disponible.

Chez Tranquil IT, nous voulons rendre au logiciel libre un peu de ce que nous lui avons pris et donc nous tenons à respecter notre promesse de laisser en libre un mécanisme pour aider les administrateurs système à déployer, mettre à jour et supprimer des logiciels et des configurations sur un parc Windows.

Ce que vous trouverez dans la version Enterprise, ce sont des choses destinées à faciliter l'exploitation de grands parcs et nous rencontrons de nombreuses personnes qui sont dans ce cas d'usage.

En règle générale, nous observons que les besoins avancés de gestion arrivent quand le parc dépasse 300-400 machines.

6.5 Installer le serveur WAPT

Le serveur WAPT s'installe sur Linux Debian, Linux CentOS / RedHat ou Microsoft Windows **64bit uniquement**.

6.5.1 Préconisations d'installation

Quelques considérations de sécurité sont à prendre en compte afin de garantir un fonctionnement optimal de WAPT :

- nous vous conseillons d'installer le serveur WAPT sur un serveur Linux (Debian ou CentOS) en suivant les recommandations de sécurité de l'*ANSSI* ou les recommandations de votre agence nationale de cyberdéfense.
- l'installation d'un serveur WAPT doit être réalisée sur un serveur **réservé uniquement à cet usage** ;

Attention: Dans toutes les étapes de la documentation, vous n'utiliserez aucun accent ni caractère spécial pour :

- le login des utilisateurs ;
- le chemin de la clé privée et du certificat ;
- le CN (Common Name) ;
- le chemin d'installation de WAPT ;
- le noms de groupes ;
- le nom des machines et du serveur ;
- le chemin ver le répertoire `C:\waptdev` ;

6.5.2 Préconisations matérielles

WAPT s'installe dans une machine virtuelle ou directement sur une machine physique.

Les préconisation RAM et CPU pour le serveur WAPT sont :

Taille de parc	CPU	RAM
De 0 a 200 postes	1 CPU	1024MB
A partir de 200 postes	4 CPU	4096Mo

Un minimum de 10 Go d'espace libre est nécessaire pour le système, la base de données et les fichiers de journalisation. **Pour de meilleures performances, Tranquil IT recommande que la base de données soit stockée sur des supports rapides, tels que des disques SSD ou des SSD sur PCIe**.

Le besoin global en disque dépendra du nombre et de la taille de vos paquets WAPT (logiciels) que vous stockerez sur votre dépôt principal ; 30 Go est un bon début. Il n'est pas strictement nécessaire de stocker les paquets WAPT sur des lecteurs rapides.

6.5.3 Configurer les DNS

Configurer les DNS de l'Organisation pour WAPT

Note: La configuration DNS n'est pas obligatoire ; elle est fortement recommandée.

Pour faciliter la configuration des clients, il est fortement recommandé de configurer le serveur *DNS* pour inclure le *champ A* ou *champ CNAME* suivants :

- *srvwapt.mydomain.lan* ;
- *wapt.mydomain.lan* ;

Remplacer *mydomain.lan* par le suffixe *DNS* utilisé sur votre réseau.

Ces champs seront utilisés par les agents WAPT pour trouver le serveur WAPT ou un dépôt secondaire WAPT de proximité sur le réseau.

Configurer les champs DNS avec les « Outils d'administration de serveur distant » Microsoft (RSAT).

- le champ *A* pointe vers l'adresse IP du serveur WAPT ;

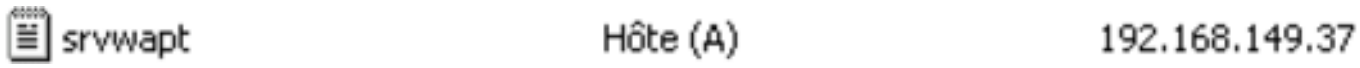


Figure14: Configuration du champ A dans la MMC Windows

Vous pouvez maintenant installer votre serveur WAPT sur l'OS de votre choix :

- *installer le serveur WAPT sur Debian* ;
- *installer le serveur WAPT sur CentOS ou RedHat* ;
- *installer le serveur WAPT sur Windows* ;

6.5.4 Installer le serveur WAPT sur Linux Debian

Préparer le serveur Debian

Pour installer une Linux Debian 10 *Buster* neuve (machine physique ou virtuelle) sans interface graphique, référez vous à la documentation officielle : Debian GNU/Linux Installation Guide.

Configurer les paramètres réseau

Les différents paramétrages préconisés ci-dessous ne sont pas spécifiques à WAPT ; vous devrez les adapter à votre environnement.

Modifier les fichiers suivant afin d'obtenir une configuration de nommage (*FQDN*) et réseau correcte (adressage IP fixe).

Dans l'exemple suivant :

- le nom *FQDN* est : *srvwapt.mydomain.lan* ;
- le nom du serveur WAPT est *srvwapt* ;

- le suffixe *DNS* est *mydomain.lan* ;
- l'adresse IP est *10.0.0.10/24* ;

Configurer le nom du serveur

Indication: Le nom du serveur WAPT ne doit pas dépasser 15 caractères (limite liée au sAMAccountName dans Active Directory).

Le nom du serveur doit être un nom FQDN (Fully Qualified Domain Name), c'est à dire à la fois le nom de machine et le suffixe DNS.

- modifier le fichier `/etc/hostname` et y renseigner le nom *FQDN* du serveur ;

```
# /etc/hostname du waptserver
srvwapt.mydomain.lan
```

- dans `/etc/hosts`, renseigner le nom *FQDN* et le nom court du serveur ;

```
# /etc/hosts du waptserver
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.10   srvwapt.mydomain.lan     srvwapt
```

Indication:

- sur la ligne correspondant à l'adresse IP du serveur WAPT mettre d'abord le nom *FQDN*, puis le nom court ;
- ne pas modifier les lignes contenant le terme `localhost` ;

Configurer l'adresse IP du serveur WAPT

- configurer les paramètres réseau du serveur WAPT dans le fichier `/etc/network/interfaces` ;

```
# /etc/network/interfaces du serveur wapt
auto eth0
iface eth0 inet static
  address 10.0.0.10
  netmask 255.255.255.0
  gateway 10.0.0.254
```

- appliquer la configuration réseau en redémarrant la machine avec un `reboot` ;
- si ce n'est pas déjà fait, *créer l'entrée DNS du serveur dans l'Active Directory* de l'Organization;
- *Configurer les DNS de l'Organisation pour WAPT*
- après le redémarrage, configurer la langue du système en anglais afin de faciliter la recherche de problèmes dans les logs ;

```
apt install locales-all
localectl set-locale LANG=en_US.UTF-8
localectl status
```

- vérifier que la machine est à l'heure (avec NTP installé) ;

```
dpkg -l | grep ntp
service ntp status
date
```

Indication: Si le paquet NTP n'est pas installé.

```
apt install ntp
systemctl enable ntp
systemctl start ntp
```

- mettre à jour Debian ;

```
apt update
apt upgrade -y
```

- installer systemd ;

```
apt install systemd
```

- installer les certificats ;

```
apt install ca-certificates
```

- redémarrer la machine ;

```
reboot
```

Vous pouvez maintenant passer à l'étape suivante et *installer WAPT sur votre Debian*.

Installer le Serveur WAPT sur Linux Debian

Attention: La procédure est différente pour la mise à jour du serveur WAPT. Pour une mise à jour, rendez-vous sur : *upgrade-wapt*.

L'installation de la partie serveur de WAPT se décompose en plusieurs étapes :

- configurer les dépôts ;
- installer les paquets complémentaires ;
- installer et initialiser la base de données PostgreSQL ;
- post-configurer le serveur ;

Note: Les paquets tis-waptserver, tis-waptsetup sont signés et il est nécessaire de récupérer la clef gpg ci-dessous pour éviter les messages de warning lors de l'installation.

Configurer les dépôts et installer les paquets WAPT et PostgreSQL

La configuration des dépôts en **WAPT Enterprise** et en **WAPT Community** diffère. **Assurez-vous de choisir la bonne procédure !**

L'installation peut vous demander le royaume kerberos. Appuyez sur `Entrée` pour passer l'étape.

Important: **Suivez cette procédure pour obtenir les bons paquets pour l'édition WAPT Enterprise**. Pour l'édition Community de WAPT, veuillez vous référer au bloc suivant.

Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **deb** pour accéder au dépôt WAPT Enterprise.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo "deb https://user:password@srvwapt-pro.tranquil.it/entreprise/debian/wapt-1.8/
↪$(lsb_release -c -s) main" > /etc/apt/sources.list.d/wapt.list
```

Important: **Suivez cette procédure pour obtenir les bons paquets pour l'édition Community de WAPT**. Pour l'édition WAPT Enterprise, veuillez vous référer au bloc précédent.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo "deb https://wapt.tranquil.it/debian/wapt-1.8/ $(lsb_release -c -s) main" > /etc/apt/
↪sources.list.d/wapt.list
```

Installer des paquets complémentaires

```
apt update
apt install tis-waptserver tis-waptsetup
```

Post-Configuration

Attention: La Post-Configuration suppose que vous avez correctement configuré le *hostname* de l'ordinateur. Pour vérifier, `echo $(hostname)` doit renvoyer l'adresse DNS qu'utiliseront les agents WAPT.

Indication: Ce script de post-configuration doit être exécuté en tant qu'utilisateur **root**.

- lancer le script :

```
/opt/wapt/waptserver/scripts/postconf.sh
```

- cliquer sur *Yes* pour accepter le lancement du postconf :

```
do you want to launch post configuration tool?

       < yes >          < no >
```

- entrer un mot de passe pour le compte *SuperAdmin* du serveur WAPT (10 caractères minimum) ;

```
Please enter the wapt server password (min. 10 characters)

*****************

< OK >          < Cancel >
```

- confirmer le mot de passe ;

```
Please enter the server password again:

*****************

< OK >          < Cancel >
```

- choisir le mode d'authentification pour l'enregistrement des machines ;
  - le choix #1 permet un enregistrement sans authentification (Comme en version 1.3). Le serveur ne vérifie pas le poste qui s'enregistre ;
  - le choix #2 active l'enregistrement initial basé sur Kerberos. (vous pourrez l'activer aussi plus tard) ;
  - le choix #3 n'active pas l'authentification kerberos pour l'enregistrement des machines mais demande une authentification wapt pour l'enregistrement des machines. The serveur WAPT réclamera le mot de passe du *SuperAdmin* à chaque nouvel enregistrement ;

```
WaptAgent Authentication type?

-------------------------------------------------------------------------------------------
 ↪-----------------------------------------
(*) 1 Allow unauthenticated registration, same behavior as wapt 1.3
( ) 2 Enable kerberos authentication required for machines registration. Registration␣
 ↪will ask for password if kerberos not available
( ) 3 Disable Kerberos but registration require strong authentication
-------------------------------------------------------------------------------------------
 ↪-----------------------------------------
                                                  < OK >          < Cancel >
```

- sélectionner *OK* pour démarrer le serveur WAPT ;

```
Press OK to start waptserver

       < OK >
```

- sélectionner *Yes* pour configurer Nginx ;

```
Do you want to configure nginx?

   < Yes >        < No >
```

- entrer le nom *FQDN* du serveur WAPT ;

```
FQDN for the WAPT server (eg. wapt.mydomain.lan)

-------------------------------------------
wapt.mydomain.lan
-------------------------------------------

      < OK >          < Cancel >
```

- faites *OK* et un certificat autosigné est généré, cette étape peut être longue ...

```
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....................................+.............................+...
```

Nginx est maintenant configuré, sélectionner *OK* pour redémarrer **Nginx** :

```
The Nginx config is done.
We need to restart Nginx?

       < OK >
```

La post-configuration est maintenant terminée.

```
Postconfiguration completed.
Please connect to https://wapt.mydomain.lan/ to access the server.

                 < OK >
```

Détail des arguments possibles du script de post-configuration :

Flag	Description
`--force-https`	Configure **Nginx** de sorte que *port 80 soit redirigé en permanence vers 443*

Le serveur WAPT est prêt.

Passez maintenant à l'étape suivante pour *installer la console WAPT*.

6.5.5 Installer le serveur WAPT sur CentOS7

Configurer le serveur CentOS / RedHat

Pour installer une CentOS7 neuve (machine physique ou virtuelle) sans interface graphique (installation de type **minimale**), référez vous à la documentation officielle : https://docs.centos.org/. Cette documentation est aussi valable pour une machine RedHat7.

Configurer les paramètres réseau

Les différents paramétrages préconisés ci-dessous ne sont pas spécifiques à WAPT ; vous devrez les adapter à votre environnement.

Modifier les fichiers suivant afin d'obtenir une configuration de nommage (*FQDN*) et réseau correcte (adressage IP fixe).

Dans l'exemple suivant :

- le nom *FQDN* est : *srvwapt.mydomain.lan* ;
- le nom du serveur WAPT est *srvwapt* ;
- le suffixe *DNS* est *mydomain.lan* ;
- l'adresse IP est *10.0.0.10/24* ;

Configurer le nom du serveur

Indication: Le nom du serveur WAPT ne doit pas dépasser 15 caractères (limite liée au sAMAccountName dans Active Directory).

Le nom du serveur doit être un nom FQDN, c'est à dire à la fois le nom de machine et le suffixe DNS.

- modifier le fichier `/etc/hostname` et y renseigner le nom *FQDN* du serveur ;

```
# /etc/hostname du waptserver
srvwapt.mydomain.lan
```

- dans `/etc/hosts`, renseigner le nom *FQDN* et le nom court du serveur ;

```
# /etc/hosts du waptserver
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.10   srvwapt.mydomain.lan srvwapt
```

Indication:

- sur la ligne correspondant à l'adresse IP du serveur WAPT mettre d'abord le nom *FQDN*, puis le nom court ;
- ne pas modifier les lignes contenant le terme `localhost` ;

Configurer l'adresse IP du serveur WAPT

- modifier le fichier `/etc/sysconfig/network-scripts/ifcfg-eth0` et définir une adresse IP statique. Le nom du fichier peut être différent, par exemple `ifcfg-ens0` ;

```
# /etc/sysconfig/network-scripts/ifcfg-eth0 du serveur wapt
TYPE="Ethernet"
BOOTPROTO="static"
NAME="eth0"
ONBOOT="yes"
IPADDR=10.0.0.10
```

(suite sur la page suivante)

(suite de la page précédente)

```
NETMASK=255.255.255.0
GATEWAY=10.0.0.254
DNS1=10.0.0.1
DNS2=10.0.0.2
```

- appliquer la configuration réseau en redémarrant la machine avec un `reboot` ;

```
reboot
```

- si ce n'est pas déjà fait, *créer l'entrée DNS du serveur dans l'Active Directory* de l'*Organisation* ;
- après le redémarrage, configurer la langue du système en anglais afin de faciliter la recherche de problèmes dans les logs ;

```
localectl set-locale LANG=en_US.utf8
localectl status
```

- vérifier que la machine est à l'heure (avec NTP installé), que SElinux et que le firewall sont activés ;

```
yum list installed | grep ntp
service ntpd status
date
sestatus
systemctl status firewalld
```

Indication: Si le paquet NTP n'est pas installé.

```
yum install ntp
systemctl enable ntpd.service
systemctl start ntpd
```

- mettre à jour CentOS7 et installer le dépôt EPEL (Extra Packages for Enterprise Linux) ;

```
yum update
yum install epel-release wget sudo
```

Vous pouvez maintenant passer à l'étape suivante et *installer WAPT sur votre CentOS / RedHat*.

Installer le serveur WAPT sous CentOS / RedHat

Attention: La procédure est différente pour la mise à jour du serveur WAPT. Pour une mise à jour, rendez-vous sur : *upgrade-wapt*.

L'installation de la partie serveur de WAPT se décompose en plusieurs étapes :

- configurer les dépôts ;
- installer des paquets complémentaires ;
- installer et initialiser la base de données PostgreSQL ;
- post-configurer le serveur ;

Configurer les dépôts et installer les paquets WAPT et PostgreSQL

La configuration des dépôts en **WAPT Enterprise** et en **WAPT Community** diffère. **Assurez-vous de choisir la bonne procédure!**

L'installation peut vous demander le royaume kerberos. Appuyer sur `Entrée` pour passer l'étape.

Important: **Suivez cette procédure pour obtenir les bons paquets pour l'édition WAPT Enterprise**. Pour l'édition Community de WAPT, veuillez vous référer au bloc suivant.

Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **baseurl** pour accéder au dépôt WAPT Enterprise.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://user:password@srvwapt-pro.tranquil.it/entreprise/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF
```

Important: **Suivez cette procédure pour obtenir les bons paquets pour l'édition Community de WAPT**. Pour l'édition WAPT Enterprise, veuillez vous référer au bloc précédent.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF
```

Installer des paquets complémentaires

```
wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/centos7/RPM-GPG-KEY-TISWAPT-7"; rpm --
↪import /tmp/tranquil_it.gpg
yum install epel-release
yum install postgresql96-server postgresql96-contrib tis-waptserver tis-waptsetup cabextract
```

Post-Configuration

- initialiser la base de données et activer les services :

```
sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb
sudo systemctl enable postgresql-9.6 waptserver nginx
sudo systemctl start postgresql-9.6 nginx
```

Attention: La Post-Configuration suppose que vous avez correctement configuré le hostname de l'ordinateur. Pour vérifier, `echo $(hostname)` doit renvoyer l'adresse DNS qu'utiliseront les agents WAPT.

Indication: Ce script de post-configuration doit être exécuté en tant qu'utilisateur **root**.

- lancer le script :

```
/opt/wapt/waptserver/scripts/postconf.sh
```

- cliquer sur *Yes* pour accepter le lancement du postconf :

```
do you want to launch post configuration tool?

       < yes >          < no >
```

- entrer un mot de passe pour le compte *SuperAdmin* du serveur WAPT (10 caractères minimum) ;

```
Please enter the wapt server password (min. 10 characters)

*****************

        < OK >          < Cancel >
```

- confirmer le mot de passe ;

```
Please enter the server password again:

*****************

        < OK >          < Cancel >
```

- choisir le mode d'authentification pour l'enregistrement des machines ;
  - le choix #1 permet un enregistrement sans authentification (Comme en version 1.3). Le serveur ne vérifie pas le poste qui s'enregistre ;
  - le choix #2 active l'enregistrement initial basé sur Kerberos. (vous pourrez l'activer aussi plus tard) ;
  - le choix #3 n'active pas l'authentification kerberos pour l'enregistrement des machines mais demande une authentification wapt pour l'enregistrement des machines. The serveur WAPT réclamera le mot de passe du *SuperAdmin* à chaque nouvel enregistrement ;

```
WaptAgent Authentication type?

------------------------------------------------------------------------------------------------------------
↪----------------------------------------
(*) 1 Allow unauthenticated registration, same behavior as wapt 1.3
( ) 2 Enable kerberos authentication required for machines registration. Registration will␣
↪ask for password if kerberos not available
( ) 3 Disable Kerberos but registration require strong authentication
------------------------------------------------------------------------------------------------------------
↪----------------------------------------
                                                    < OK >          < Cancel >
```

- sélectionner *OK* pour démarrer le serveur WAPT ;

```
Press OK to start waptserver

       < OK >
```

- sélectionner *Yes* pour configurer Nginx ;

```
Do you want to configure nginx?

   < Yes >        < No >
```

- entrer le nom *FQDN* du serveur WAPT ;

```
FQDN for the WAPT server (eg. wapt.acme.com)

-------------------------------------------
wapt.mydomain.lan
-------------------------------------------

      < OK >          < Cancel >
```

- faites *OK* et un certificat autosigné est généré, cette étape peut être longue . . .

```
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....................................+.............................+...
```

Nginx est maintenant configuré, sélectionner *OK* pour redémarrer **`Nginx`** :

```
The Nginx config is done.
We need to restart Nginx?

      < OK >
```

La post-configuration est maintenant terminée.

```
Postconfiguration completed.
Please connect to https://wapt.mydomain.lan/ to access the server.

                 < OK >
```

Détail des arguments possibles du script de post-configuration :

Flag	Description
`--force-https`	Configure **`Nginx`** de sorte que *port 80 soit redirigé en permanence vers 443*

Le serveur WAPT est prêt.

Passez maintenant à l'étape suivante pour *installer la console WAPT*.

6.5.6 Installer le serveur WAPT sur Windows

Installer WAPT sur Microsoft Windows

Attention:

- Le serveur WAPT ne peut pas être installé sur une machine qui écoute déjà sur le port 80 et 443 (exemple WSUS avec IIS).
- les ports 80, 443 et 8080 sont utilisés par le serveur WAPT et doivent être disponibles ;
- Si les port 80 et 443 sont déjà occupés par un autre service, vous pouvez suivre la documentation pour *changer les ports d'écoute*.
- le port 8088 est utilisé sur les postes clients par l'agent WAPT et doit être disponible ;
- WAPT Serveur n'est pas installable sur une version x86 de Windows ;
- L'installation de WAPTServer doit être faite avec le compte **Administrateur Local** de la machine ;

Attention: À partir de la version WAPT 1.5, le **SEUL** serveur web supporté est **`Nginx`**. **Les serveurs web Apache ou IIS (notamment en cohabitation avec un WSUS sur les ports 80 et 443), ne sont plus supportés officiellement**.

En cas de difficulté lors de l'installation de WAPT, visitez *les questions fréquentes*.

Note:

- l'installation de WAPT sur un serveur Linux est la méthode à privilégier ;
- le serveur WAPT peut s'installer sur Windows 7, 8.1, 10 et Windows Server 2008 et R2, 2012 et R2, 2016 et 2019 en **64bit uniquement** ;

- Community: download and execute waptserversetup.exe;
- Enterprise: download and execute waptserversetup.exe;
- choisir la langue d'installation ;
- accepter la licence GNU Public Licence et cliquer sur *Suivant* pour passer à l'étape suivante ;
- choisir le répertoire d'installation (laisser par défaut) et cliquer sur *Suivant* pour passer à l'étape suivante ;
- choisir une tâche supplémentaire (laisser en l'état pour une première installation) ;
- choisir le mot de passe pour le serveur WAPT ;
- créer une clé s'il s'agit de votre première installation, sinon sélectionner la clé existante ;

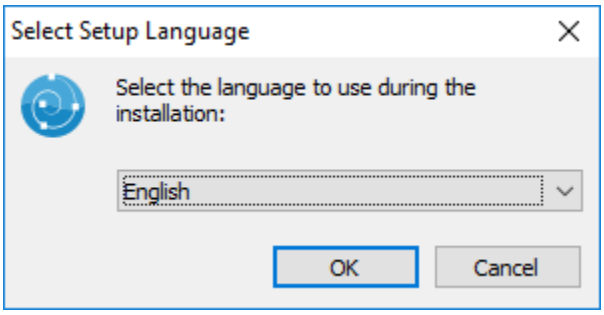


Figure15: Choisir la langue pour WAPT

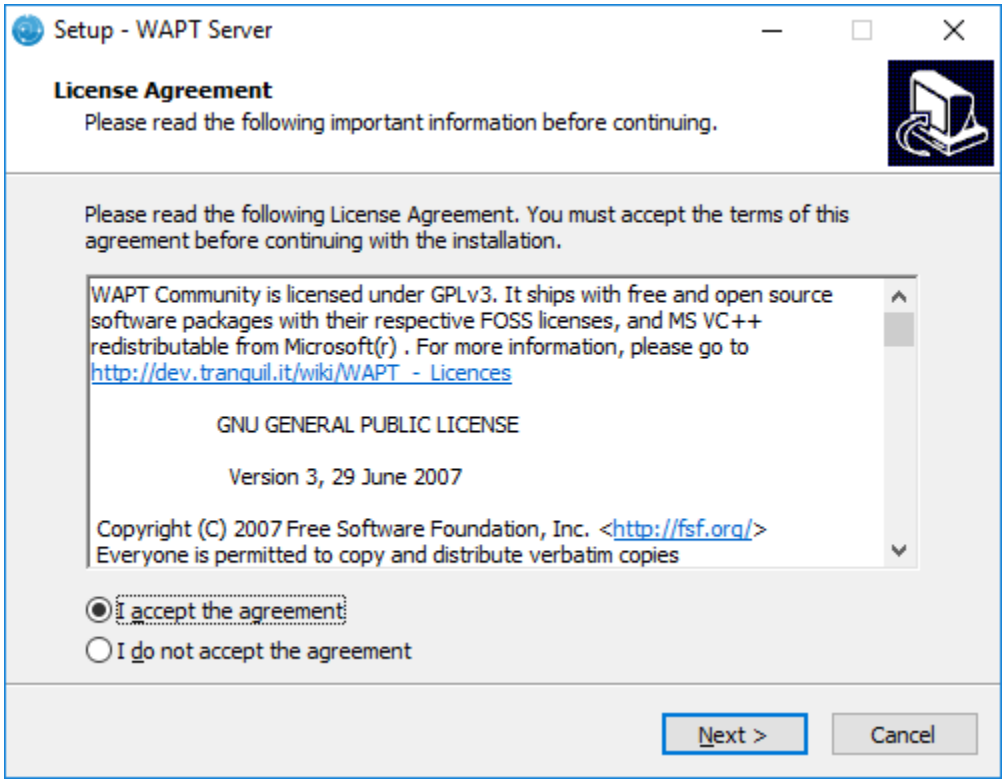


Figure16: Accepter la licence pour le logiciel WAPT

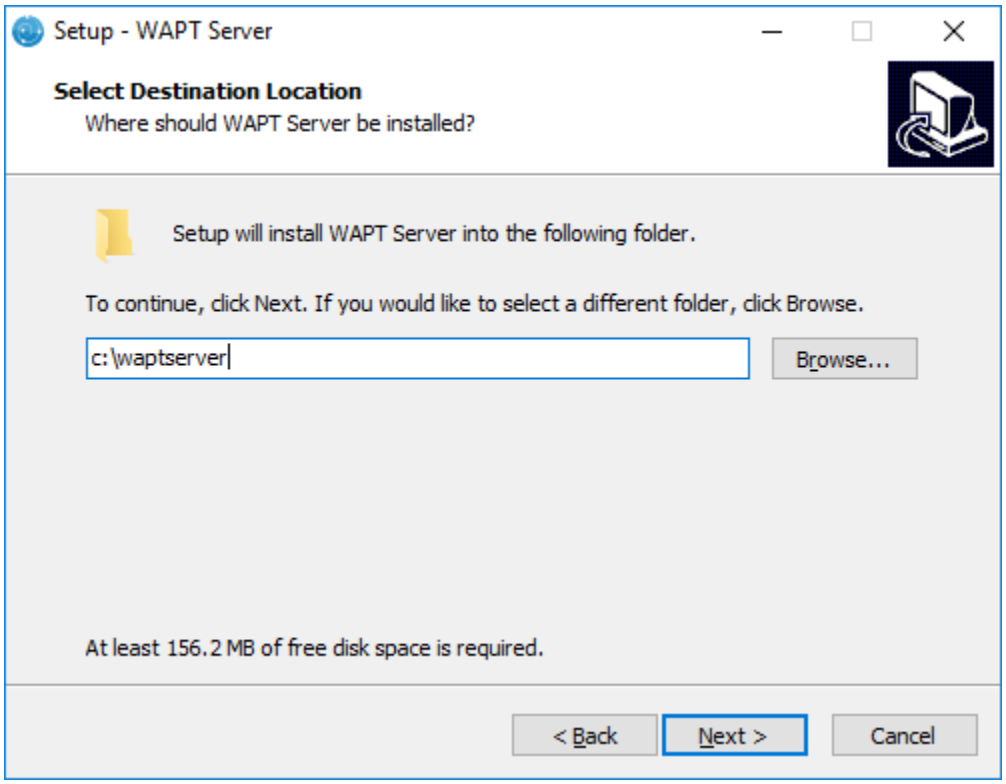


Figure17: Choisir le dossier d'installation de WAPT

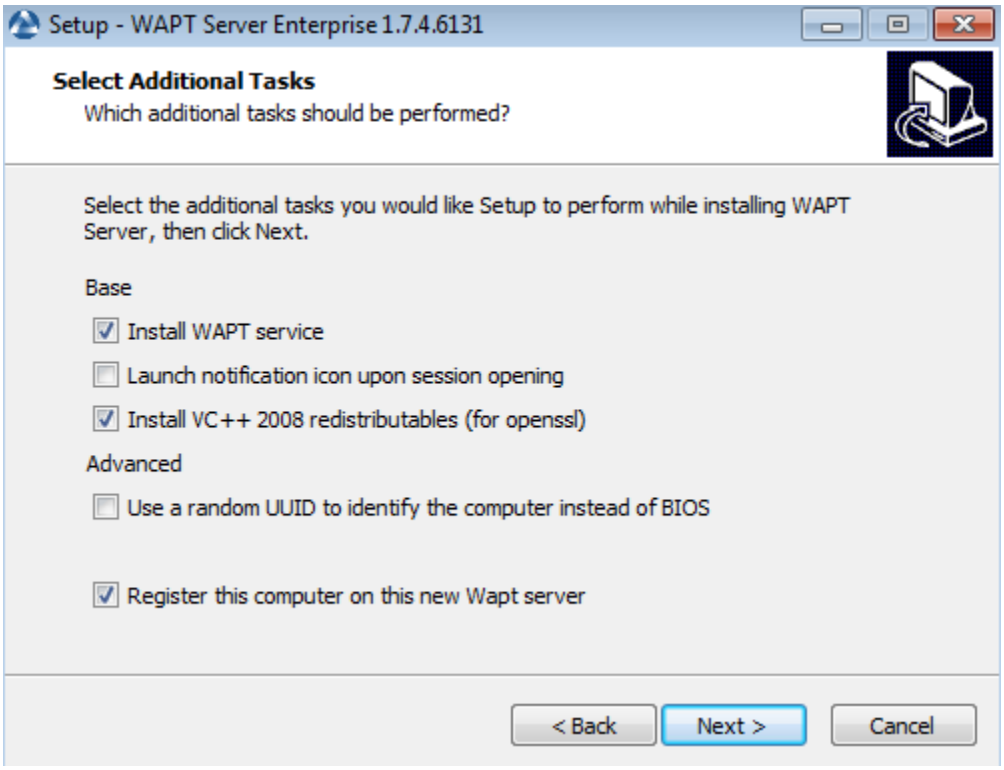


Figure18: Choisissez une tâche supplémentaire

- choisir le mot de passe pour la clé privée ;
- construire l'agent WAPT ;
- choisir le préfixe utilisé pour nommer vos paquets WAPT ;
- cliquer sur *Installer* pour passer à l'étape suivante ;
- cliquer sur *Terminé* pour terminer l'installation du serveur WAPT ;

Le serveur WAPT sur Windows est prêt.

Vous pouvez maintenant passer à la documentation pour *installer l'agent WAPT* !!

Changer le port d'écoute du serveur

Note: Dans certains cas, il n'est pas possible d'installer le serveur WAPT sur un serveur Windows car un service occupe déjà les ports 80 et 443.

C'est le cas par exemple si un IIS est actif sur la machine. (Exemple : serveur anti-virus, WSUS, Serveur Web . . .)

Dans ce cas, nous allons donc changer le port d'écoute du serveur web **`Nginx`** intégré au Serveur WAPT.

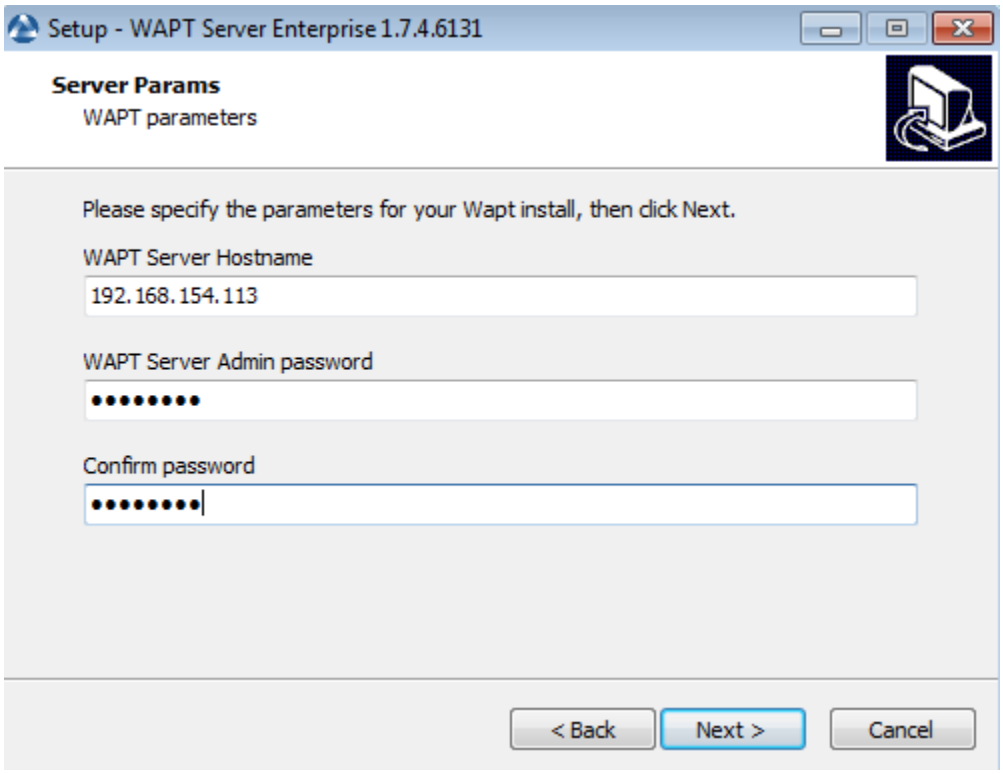


Figure19: Choisir un mot de passe

Installer le Serveur WAPT

- l'installation de WAPT a tout de même besoin que les ports 80 et 443 soient disponibles au moment de l'installation, donc couper le service qui écoute sur le port 80 et / ou 443 (IIS / Antivirus).
- lancer maintenant l'installation du serveur WAPT et suivre la procédure du postconf, mais ne pas lancer pas la console WAPT. Si vous avez besoin d'aide, vous pouvez suivre la documentation pour *installer le serveur WAPT sur Windows*.
- stopper maintenant le service **`Nginx`** et le service WAPT :

```
net stop WAPTNginx
net stop waptservice
```

- redémarrer enfin le service qui écoute sur le port 80 et / ou 443 (IIS / Antivirus / Serveur Web . . .) ;

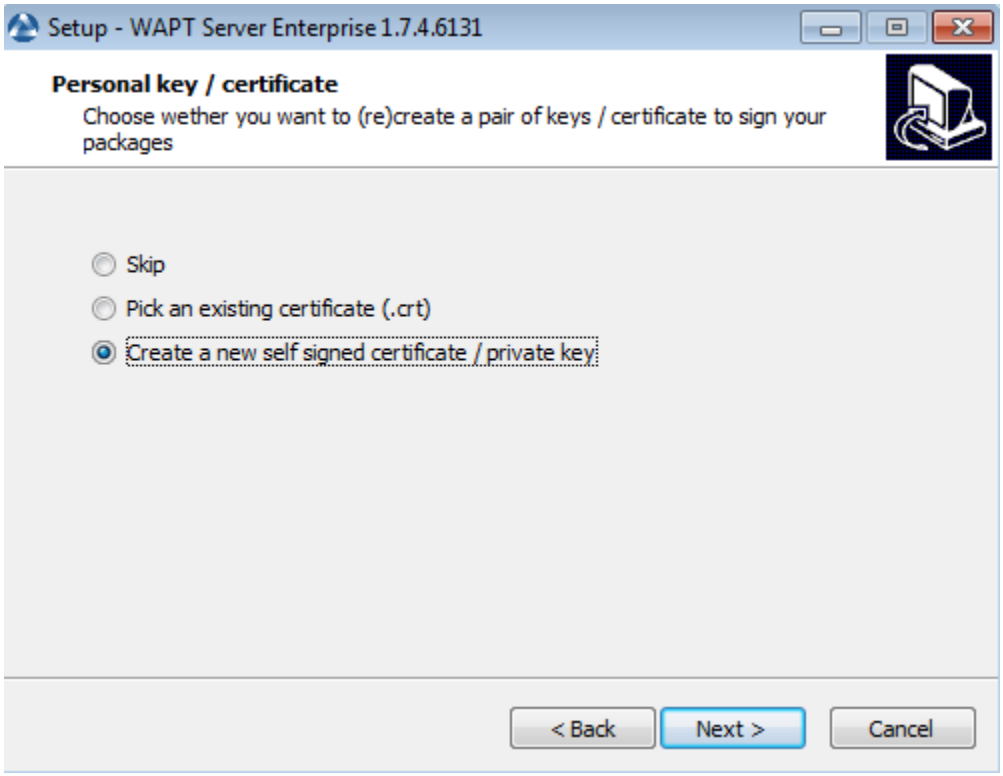


Figure20: Créer la clé de signature

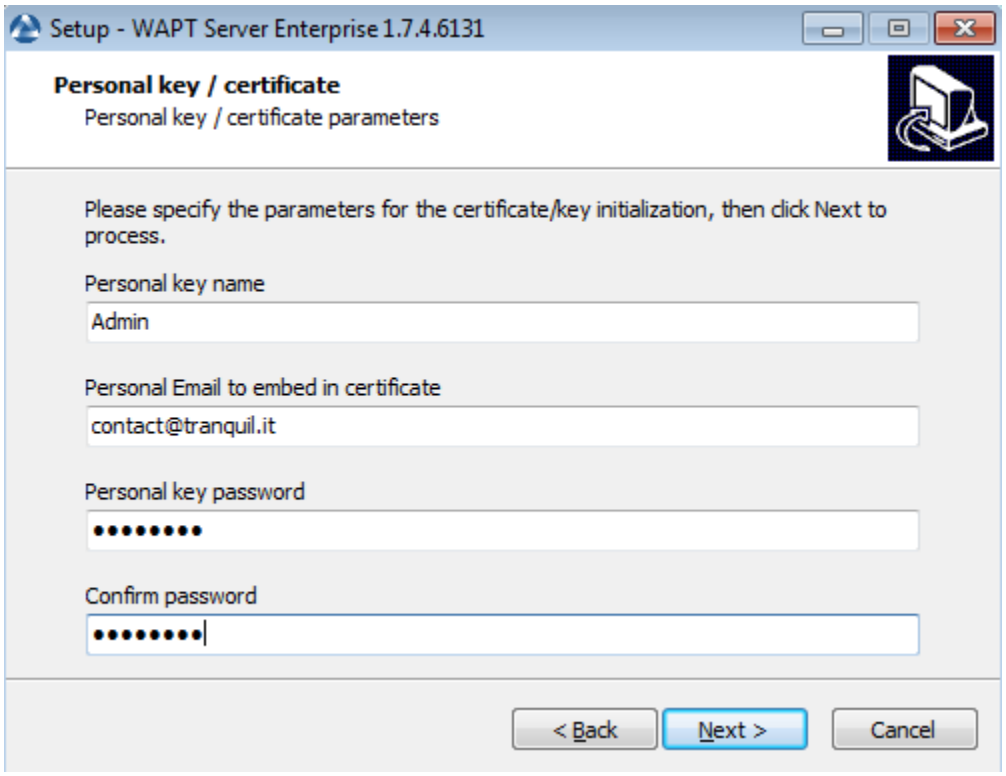


Figure21: Choisir le mot de passe de la clé privée

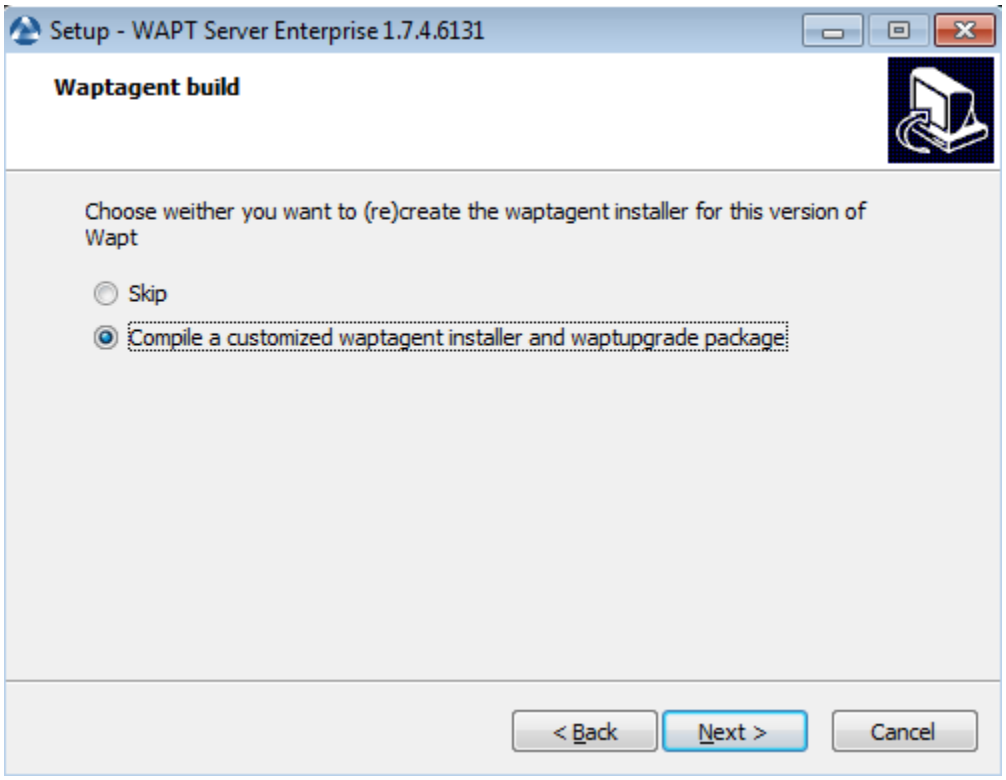


Figure22: Construire l'agent WAPT

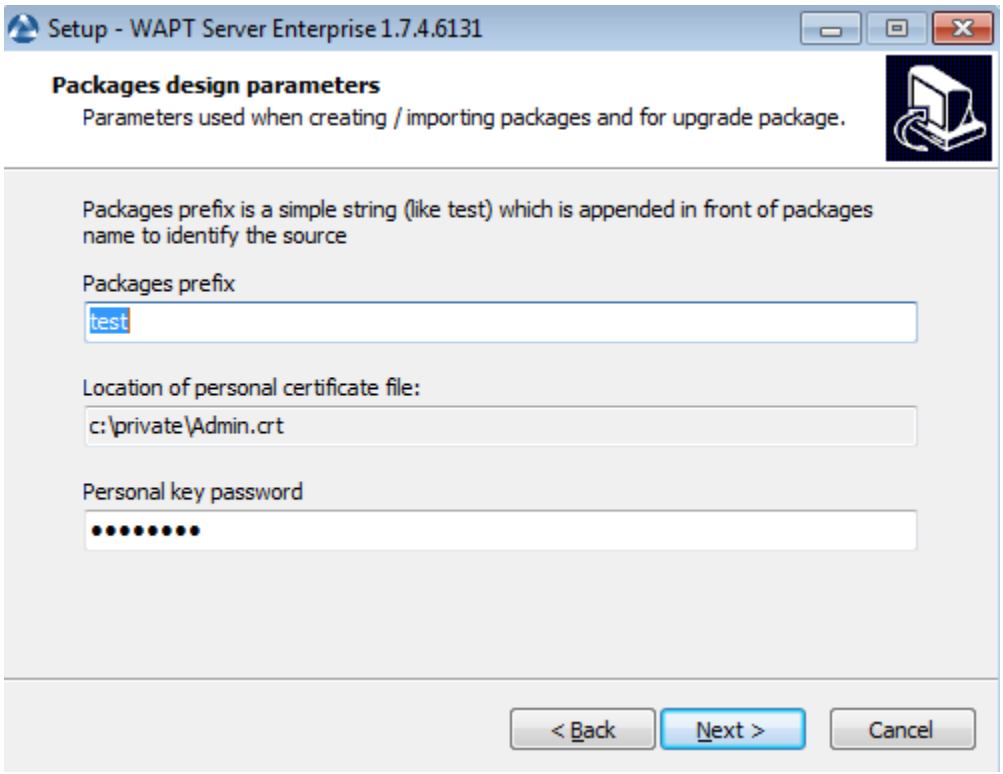


Figure23: Choisir le préfixe utilisé pour nommer vos paquets WAPT

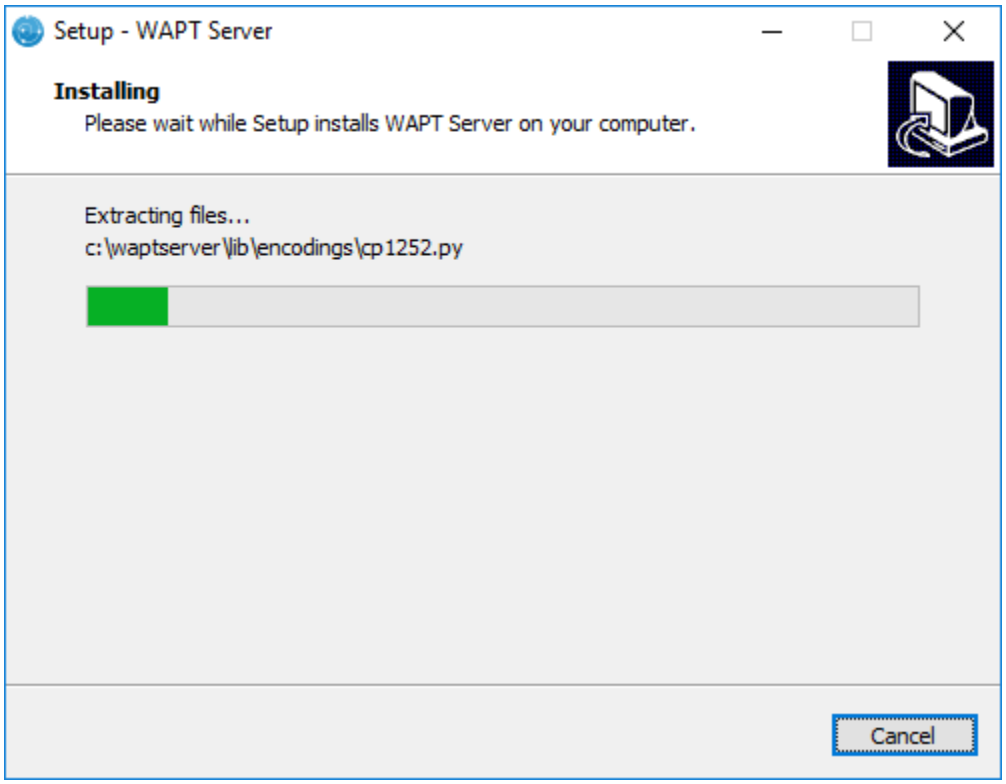


Figure24: Progression de l'installation

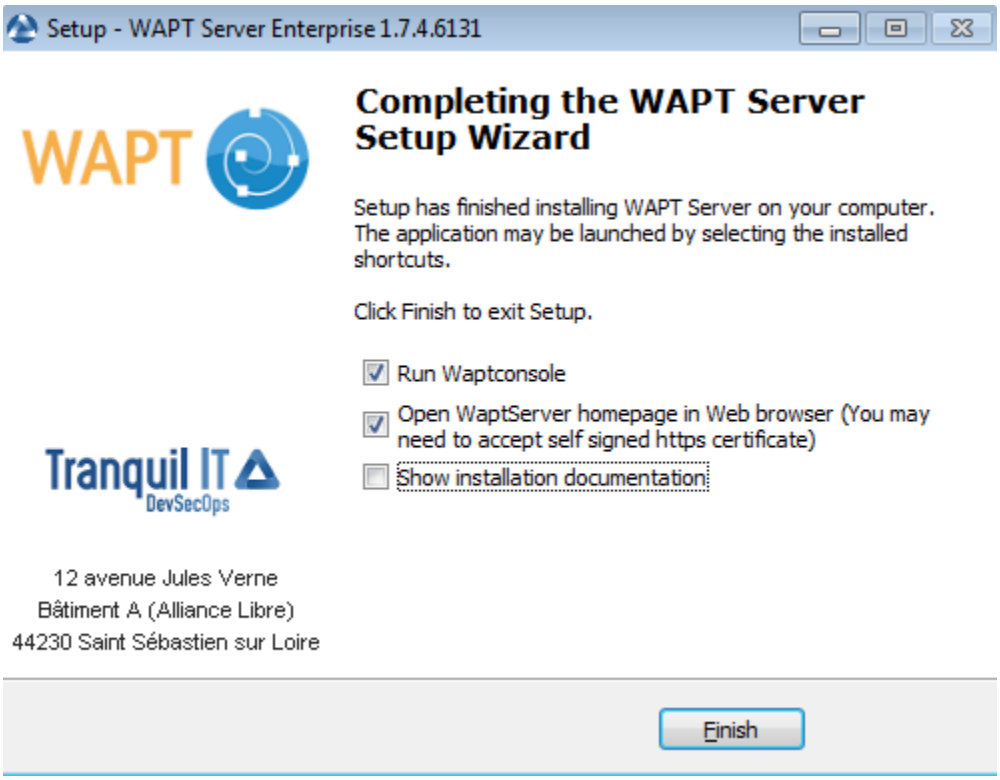


Figure25: Installation terminée

Modifier le port dans la configuration Nginx

- C:\Program Files (x86)\wapt\
- remplacer les lignes :

```
listen       80;
listen       443 ssl;
```

par :

```
listen       8000;
listen       8443 ssl;
```

- relancer **Nginx** avec `net start WAPTNginx` :
- C:\Program Files (x86)\wapt\
- ajouter le port a l'url indiquée, exemple :

```
repo_url=https://wapt.mydomain.lan:8443/wapt
wapt_server=https://wapt.mydomain.lan:8443
```

- redémarrer ensuite le service WAPT avec `net start waptservice` ;

Modifier les règles de pare-feu Windows

Vous devez maintenant modifier les deux règles waptserver créées pendant l'installation : waptserver *waptserver 80* et *waptserver 443*.

- changer les ports :

Passer à l'étape suivante pour *lancer la console WAPT*.

Indication: Si vous aviez déjà démarré la console WAPT, ne pas oublier de modifier aussi le port dans la configuration de la console en cliquant sur la *clé à molette* sur la mire de connexion.

6.5.7 Installer le serveur WAPT avec Ansible

Installer le serveur WAPT avec Ansible

Pour éviter les erreurs et automatiser le déploiement de votre serveur WAPT, nous fournissons des rôles Ansible pour installer le serveur WAPT.

Vous pouvez explorer le code source du rôle en visitant le dépôt Tranquil IT sur Github.

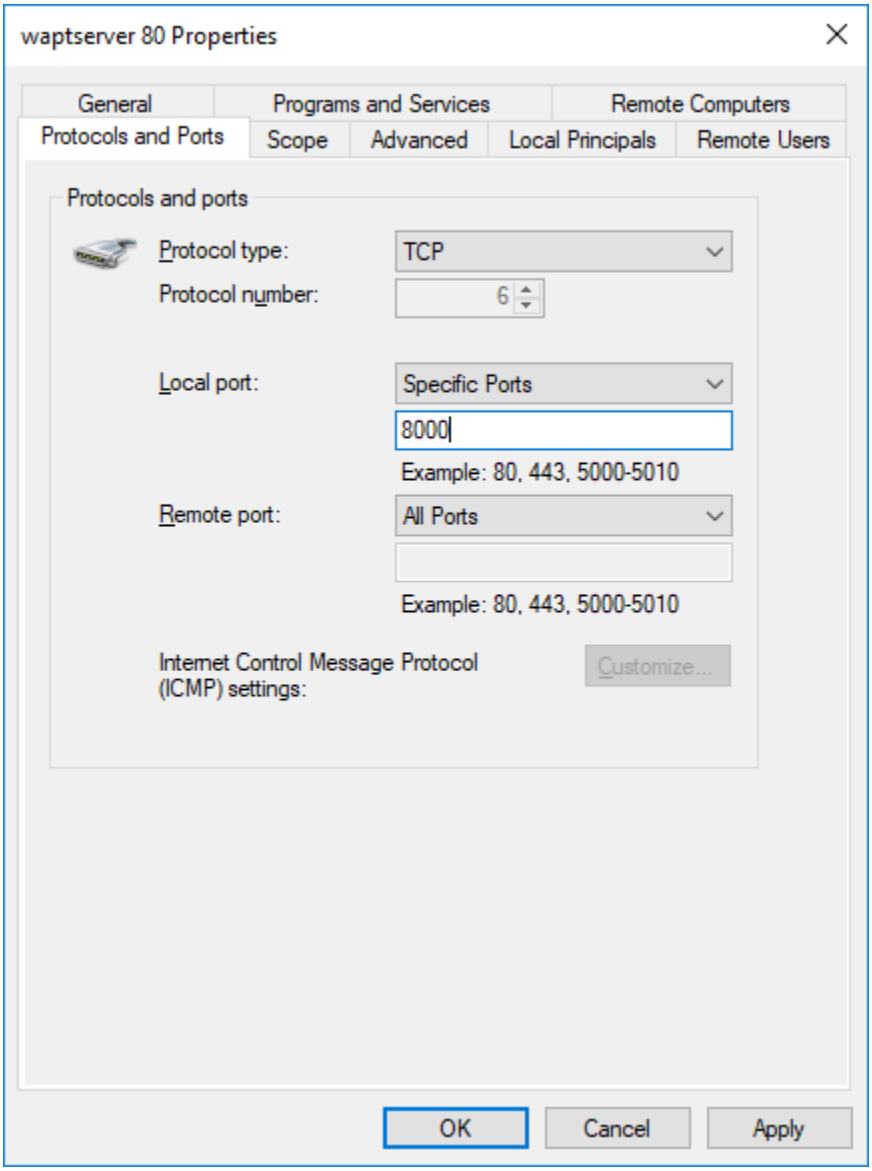


Figure26: Modification du port de 80 a 8000

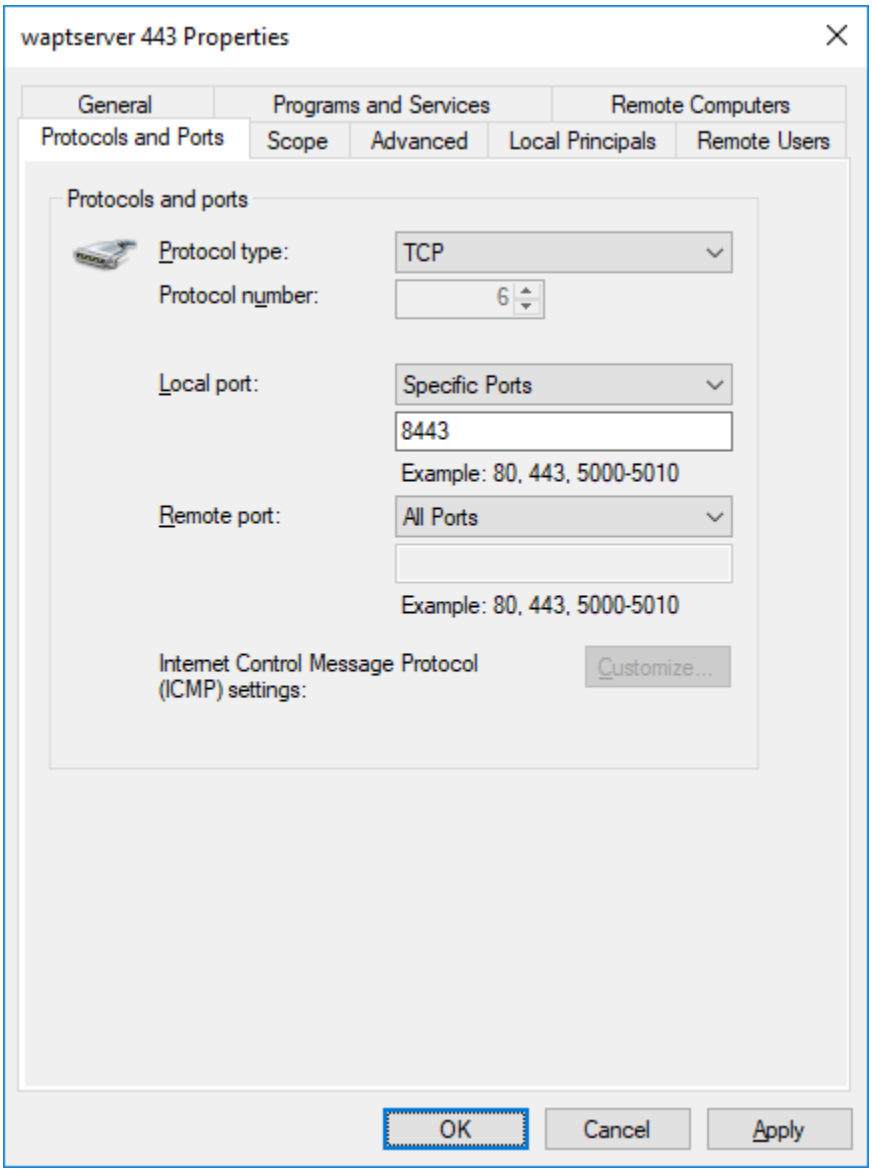


Figure27: Modification du port de 443 a 8443

Pré-requis

- Postes Debian Linux ou CentOS;
- un compte utilisateur sudoers sur ces postes;
- Ansible 2.8;

Installer le rôle Ansible

- installer le rôle Ansible `tranquilit.waptagent` ;

```
ansible-galaxy install tranquilit.waptserver
```

- pour installer le rôle autre part, utiliser la commande *-p* comme ceci;

```
ansible-galaxy install tranquilit.waptserver -p /path/to/role/directory/
```

Utiliser le rôle Ansible

- assurez vous d'avoir une clé ssh déployée sur vos postes, sinon vous pouvez en générer une et la copier comme ci dessous;

```
ssh-keygen -t ed25519
ssh-copy-id -i id_ed25519.pub user@srvwapt.mydomain.lan
ssh user@srvwapt.mydomain.lan -i id_ed25519.pub
```

- éditer le fichier d'inventaire Ansible (`./hosts`) et ajouter les hôtes Linux ;

```
[srvwapt]
srvwapt.mydomain.lan ansible_host=192.168.1.40
```

- créer un playbook avec le contenu suivant dans `./playbooks/deploywaptagent.yml` ;

```
- hosts: srvwapt
  roles:
    - { role: tranquilit.waptserver }
```

- lancez votre playbook avec la commande suivante ;

```
ansible-playbook -i ./hosts ./playbooks/wapt.yml -u user --become --become-method=sudo -K
```

Félicitations, vous avez installé votre serveur WAPT sur votre serveur Linux!

Variables du rôle

Les variables disponibles sont listées ci-dessous, avec les valeurs par défaut (voir `defaults/main.yml`) :

- version de WAPT qui sera installée depuis les dépôts WAPT Deb/RPM ;

```
wapt_version: "1.8"
```

- version de PostgreSQL qui sera installée depuis les dépôts WAPT Deb/RPM ;

```
pgsql_version: "9.6"
```

- version CentOS utilisée pour l'adresse du dépôt RPM ;

```
centos_version: "centos7"
```

- La valeur par défaut de `launch_postconf` est *True*, elle lance le script de post-configuration du serveur WAPT en mode silencieux ;

```
launch_postconf: True
```

Exemple de playbook

Voici un exemple de playbook Ansible.

```
- hosts: srvwapt
  vars_files:
    - vars/main.yml
  roles:
    - tranquilit.waptserver
```

6.6 Configurer WAPT

Le serveur WAPT est installé, les étapes suivantes sont :

6.6.1 Installer la console de management WAPT

Note:

- l'administration du serveur WAPT se fait grâce à une console installée sur la station de travail de l'*Administrateur* ;
- la machine d'administration doit être membre du domaine Active Directory de l'*Organisation* ;
- le nom de la machine d'administration ne doit pas dépasser 15 caractères (limite liée au sAMAccountName dans Active Directory) ;
- le poste de l'Administrateur sera critique pour l'administration de WAPT et les tests de paquets WAPT ;
- si vos DNS sont correctement configurés, vous pourrez télécharger la console depuis le serveur WAPT en visitant : https://srvwapt.mydomain.lan ;

Télécharger et lancer l'installation de la console sur le poste Administrateur

Attention: Attention, la console **NE DOIT PAS** être installée sur le serveur WAPT Windows.

La console doit être installée sur le poste à partir duquel vous faites l'administration de votre réseau.

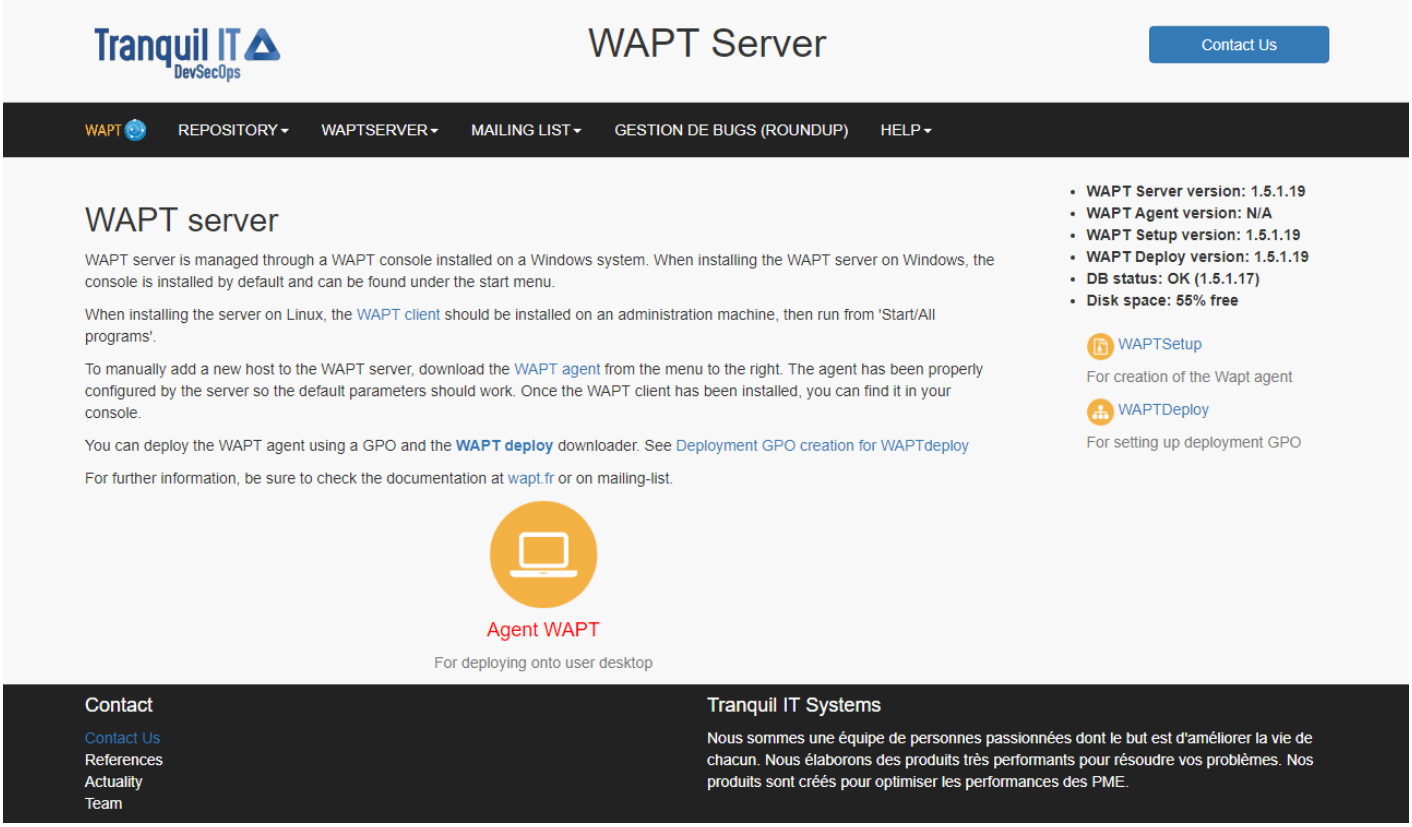


Figure28: Console d'administration Web du serveur WAPT

- si vos DNS sont correctement configurés, vous pourrez télécharger la console depuis le serveur WAPT en visitant : https://srvwapt.mydomain.lan ;
- cliquer sur *WAPTSetup* disponible sur la partie droite de la page web de votre serveur WAPT ;
- lancer l'exécutable en tant qu'*Administrateur Local* de la machine de l'*Administrateur* ;
- choisir la langue et cliquer sur *OK* pour lancer l'installation ;
- cliquer sur *OK* pour passer à l'étape suivante ;
- cliquer sur *Suivant* pour passer à l'étape suivante et accepter la licence ;
- choisir le dossier d'installation (`C:\Program Files (x86)\wapt` par défaut) ;
- cliquer sur *Suivant* et choisir les options d'installation (les valeurs par défaut sont bonnes) ;

Note:

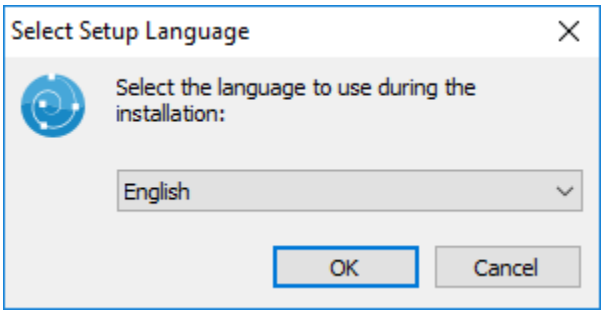


Figure29: Choisir la langue pour WAPT

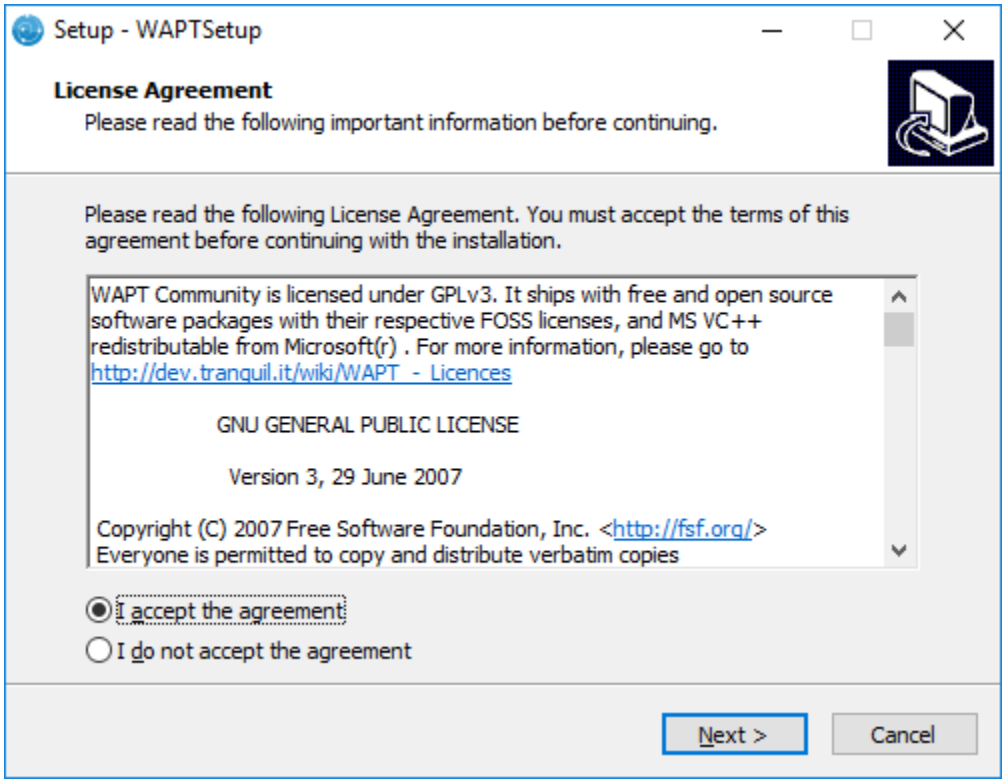


Figure30: Accepter la licence pour le logiciel WAPT

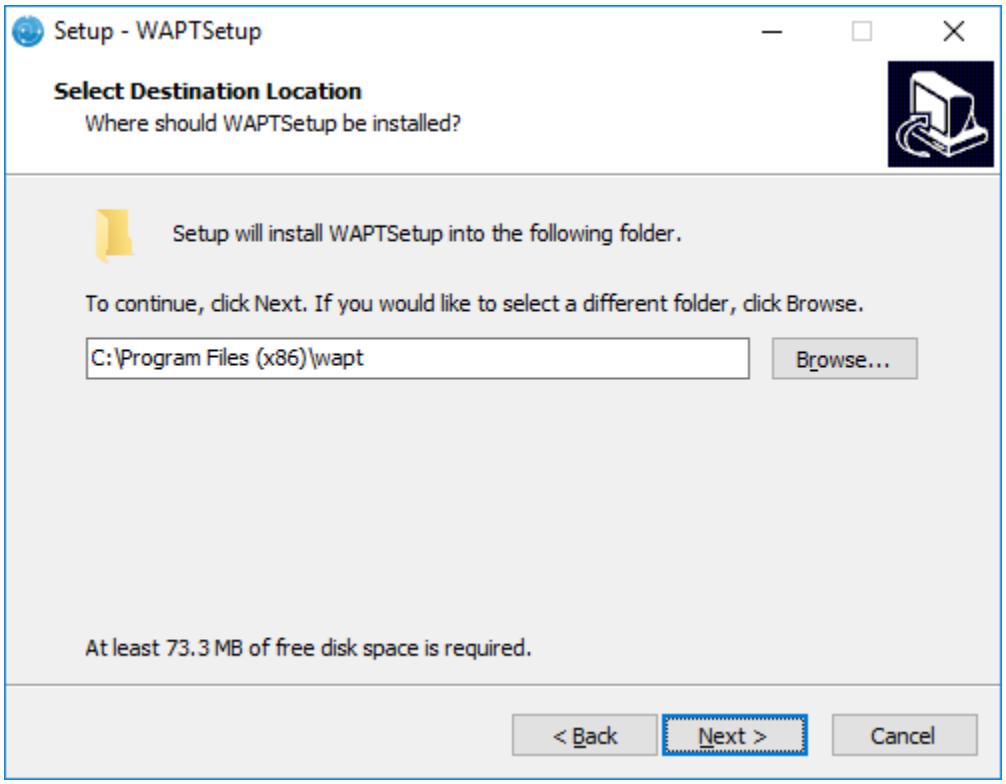


Figure31: Choisir le dossier d'installation de WAPT

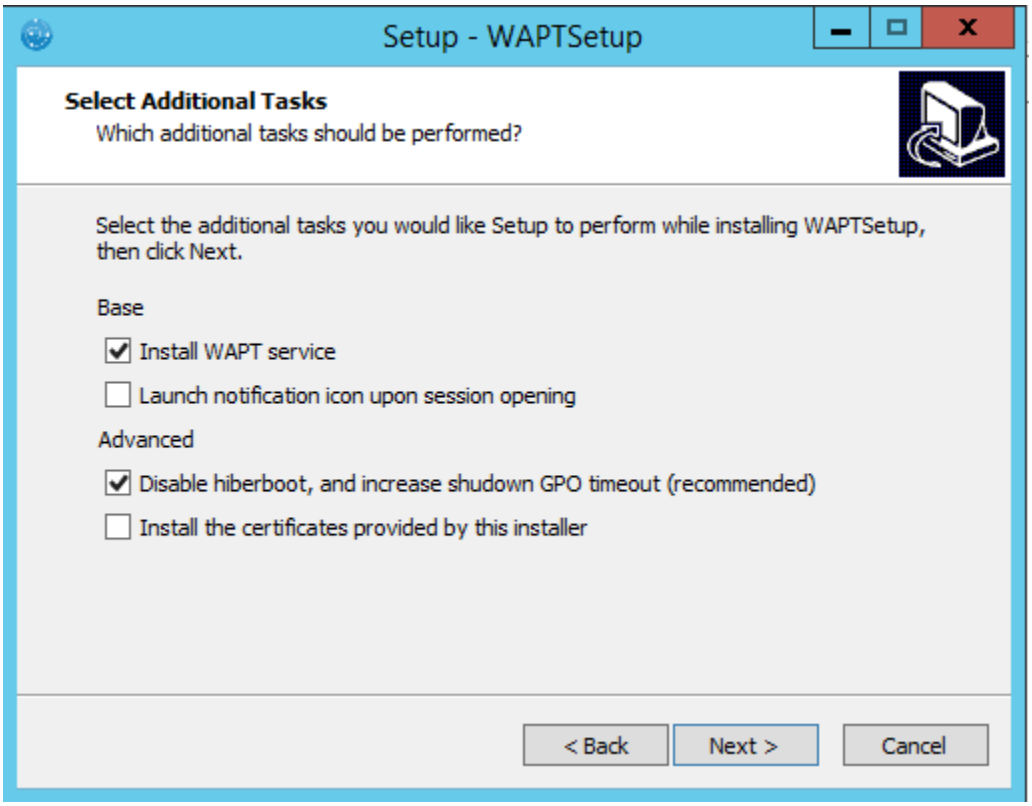


Figure32: Choisir les options d'installation

- cocher *Installer le service WAPT* pour lancer l'agent WAPT sur votre poste d'*Administrateur* ;
- cocher *Lancer l'icône de notification lors de l'ouverture de session* pour que l'icône WAPT soit accessible dans la barre des tâches ;

Indiquer l'adresse du serveur WAPT

Indication: Ici deux choix s'offrent à vous, vous pouvez vous laisser guider par l'assistant de configuration et passer directement à *Utiliser la console WAPT*, ou vous pouvez continuer la configuration manuellement.

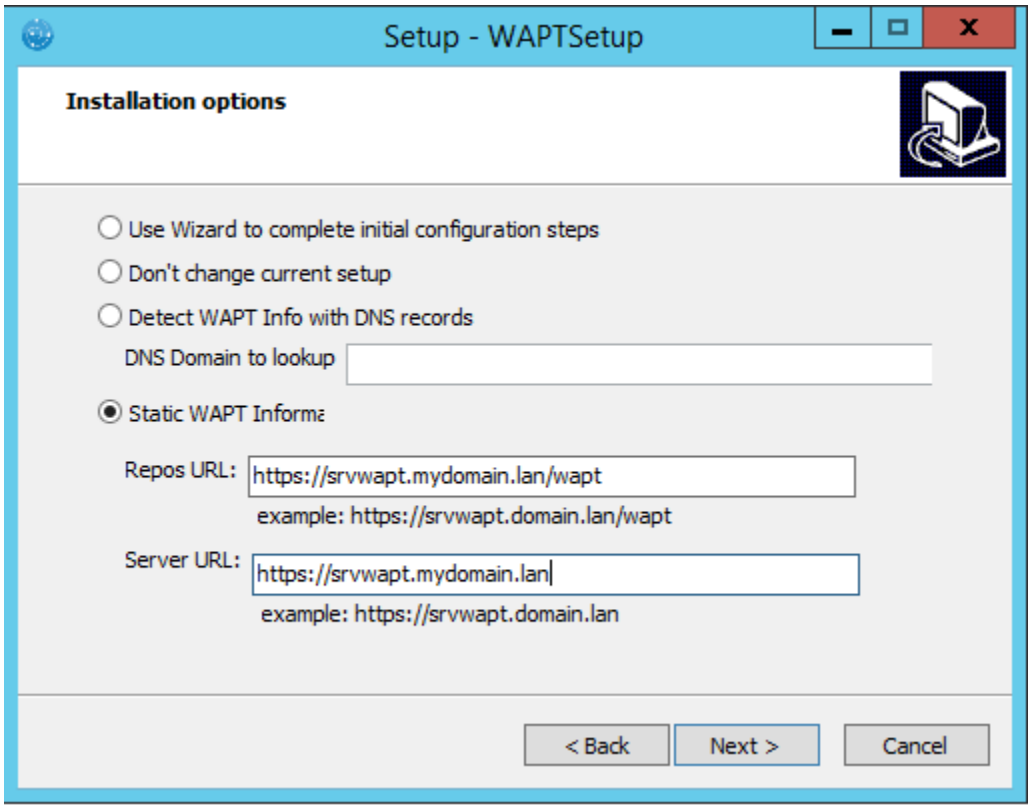


Figure33: Choisir le dépôt et le serveur WAPT

Note: Exemple :

- Repos url : https://srvwapt.mydomain.lan/wapt
- Server url : https://srvwapt.mydomain.lan

- choisir la langue et cliquer sur *OK* pour lancer l'installation ;
- cliquer sur *Installer* pour passer à l'étape suivante, attendre la fin de l'installation puis cliquer sur *Fin* (laisser les options par défaut) ;

Figure34: Assistant d'installation terminé

Note:

- cocher *Enregistrer cet ordinateur dans le serveur WAPT* pour enregistrer l'ordinateur dans l'inventaire central WAPT ;
- cocher *Mettre à jour la liste des paquets disponibles sur le dépôt principal* pour télécharger la liste des paquets disponibles ;

Lancer la console WAPT

- Lancer la console WAPT avec le binaire `C:\Program Files (x86)\wapt\waptconsole.exe`
- au premier lancement, lancer la console avec élévation de privilège : *Clic-droit sur waptconsole.exe → Exécuter en tant qu'Administrateur Local* ;
- s'identifier sur la console de management avec l'identifiant et le mot de passe de *SuperAdmin* ;

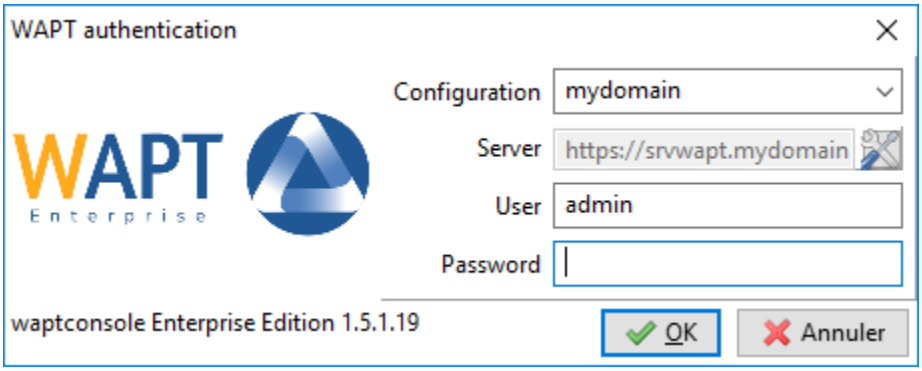


Figure35: Mire de connexion au serveur WAPT

Si vous rencontrez des problèmes pour vous connecter à la console WAPT, visitez la section des problèmes fréquents *Message d'erreur à l'ouverture de la console*;

Note: Un message peut apparaître indiquant que l'agent WAPT est introuvable.

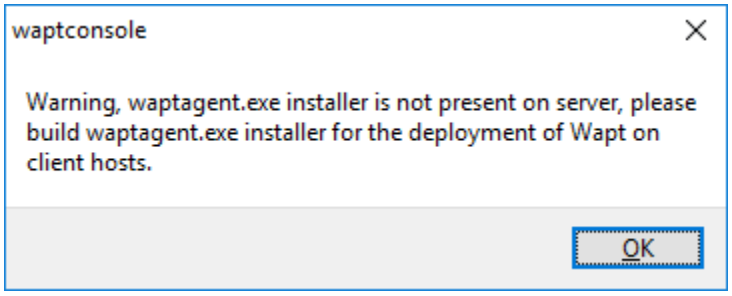


Figure36: Erreur de disparité entre la version de la console et de l'agent

Passez à l'étape suivante pour *créer votre certificat*.

6.6.2 Générer le certificat de l'Administrateur pour signer les paquets

Introduction

Hypothèse de cette documentation

- le nom de la clé privée est `wapt-private.pem` ;
- certificat public signé avec le fichier de la clé privée `wapt-private.crt` ;

La clé privée *wapt-private.pem*

Attention: Le fichier `wapt-private.pem` que nous allons générer est d'une importance **fondamentale**. Il doit être conservé en lieu sûr et correctement protégé.

Le fichier `wapt-private.pem` est la clé privée, elle est générée puis copiée par défaut dans le dossier `C:private` du poste de l'*Administrateur*.

Pour plus de sûreté, la clé privée peut être transférée sur un support externe sécurisé.

Elle servira, en pointant sur le certificat, à signer les paquets WAPT avant de les envoyer sur le dépôt.

certificat public signé avec la clé privée : `wapt-private.crt`

Le fichier `wapt-private.crt` est le certificat public qui est utilisé avec la clé privée. Il est créé par défaut dans le dossier `C:\private`, copié dans le dossier `C:\Program Files (x86)\wapt\ssl` de l'Administrateur et déployé sur les postes de travail gérés par l'Administrateur via un paquet WAPT ou une GPO.

Ce certificat sert à valider la signature des paquets avant leur installation.

Générer le certificat

Dans la console WAPT aller dans *Outils* → *Créer un certificat* ;

Important: A partir d'ici, deux cas possibles :

- *créer un certificat pour la version WAPT Community* ;
- *créer des certificats pour la version WAPT Enterprise* ;

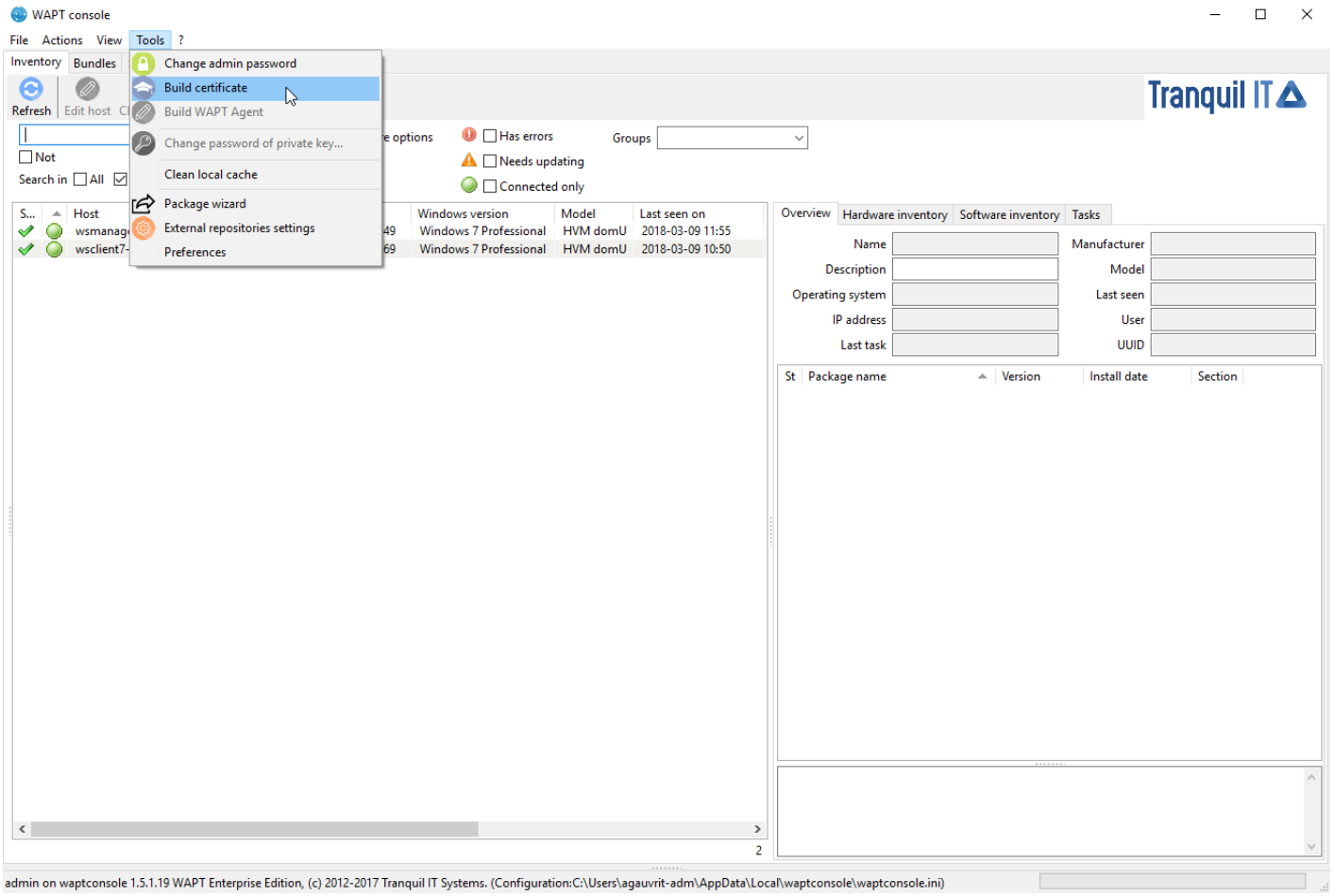


Figure37: Générer un certificat auto-signé

Générer le certificat - WAPT Community

- renseigner les champs suivants :

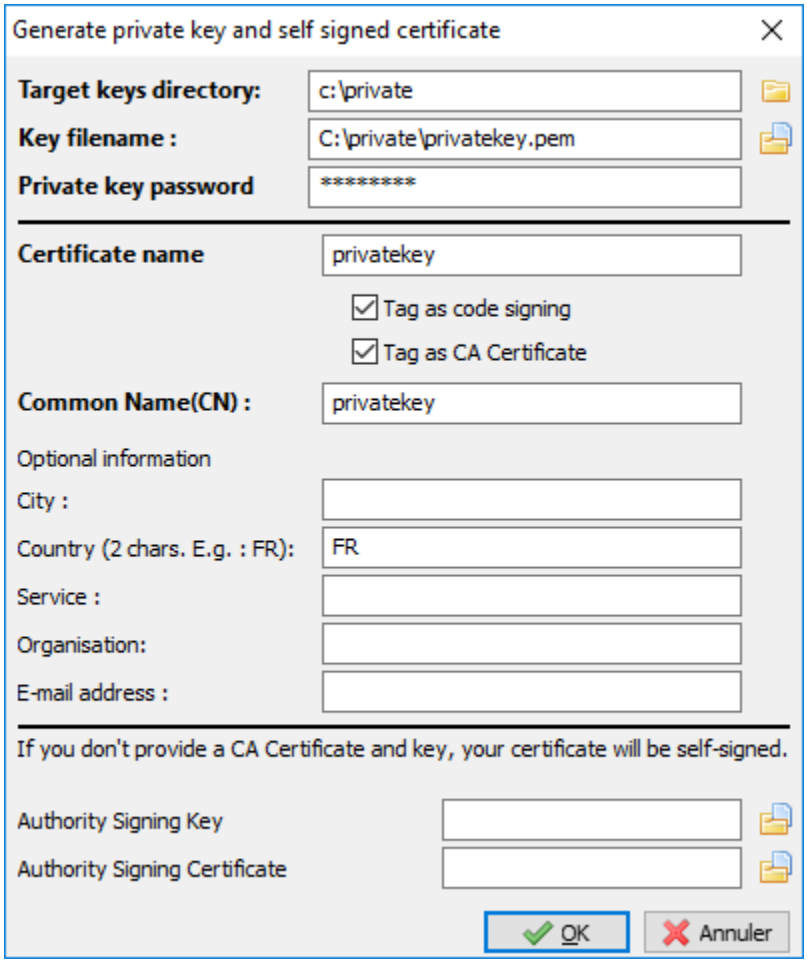


Figure38: Générer un certificat auto-signé

- cliquer sur *OK* pour passer à l'étape suivante ;

Les informations à renseigner sont :

- *Répertoire de destination* où seront sauvegardés la clé privée et le certificat public : **obligatoire** ;
- *Nom de la clé* donné aux fichiers `.pem` et `.crt` ;
- *Mot de passe de la clé* pour vérouiller la clé : **obligatoire** ;
- *Mot de passe de la clé* pour vérouiller la clé : **obligatoire** ;
- *Common Name (CN)* pour identifier formellement le signataire : **obligatoire** ;
- *Certificate Name* donné au certificat `.crt` : **obligatoire** ;
- *Informations optionnelles* : détails supplémentaires stockées dans la clé privée. Ces informations permettront aux destinataires d'un paquet WAPT de connaître son origine ;

Pour une première installation, vous pouvez suivre l'exemple de la capture d'écran.

Indication: Le mot de passe de la clé privée doit être choisi en conformité avec les préconisations en vigueur dans votre *Organisation* (par exemple les recommandations *ANSSI*).

Danger:

- le répertoire de destination de la clé privée ne doit pas être le dossier d'installation de WAPT (`C:\Program Files (x86)\wapt`);
- si votre clé est stockée dans `C:\Program Files (x86)\wapt`, votre clé privée d'*Administrateur* sera déployée sur vos clients, **absolument à ne pas faire !**

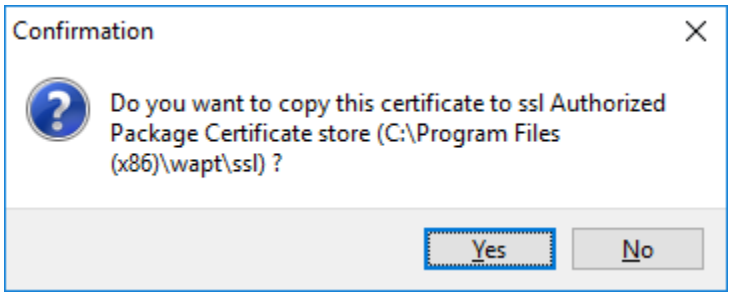


Figure39: Confirmation de copie dans le répertoire ssl

- cliquer sur *Oui* pour copier le certificat nouvellement généré dans le dossier `C:\Program Files (x86)\wapt\ssl`. Ce certificat sera récupéré lors de la compilation de l'agent WAPT et déployé sur les ordinateurs clients ;

Lorsque tout s'est bien déroulé le message suivant apparaît :

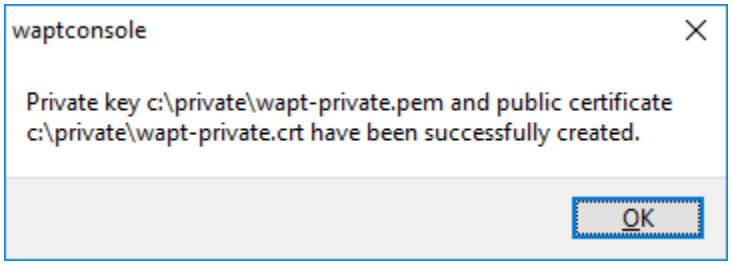


Figure40: Réussite de la génération du certificat

- cliquer sur *OK* pour passer à l'étape suivante ;

Vous pouvez maintenant passer à l'étape suivante et *configurer votre console WAPT* !!

Générer le certificat - WAPT Enterprise

En version WAPT Enterprise, créer une clé « Master » capable de signer des paquets logiciels et également de signer d'autres certificats (CA).

Indication: Vous pourrez par la suite suivre la documentation pour *Différencier les niveaux de rôle dans WAPT*.

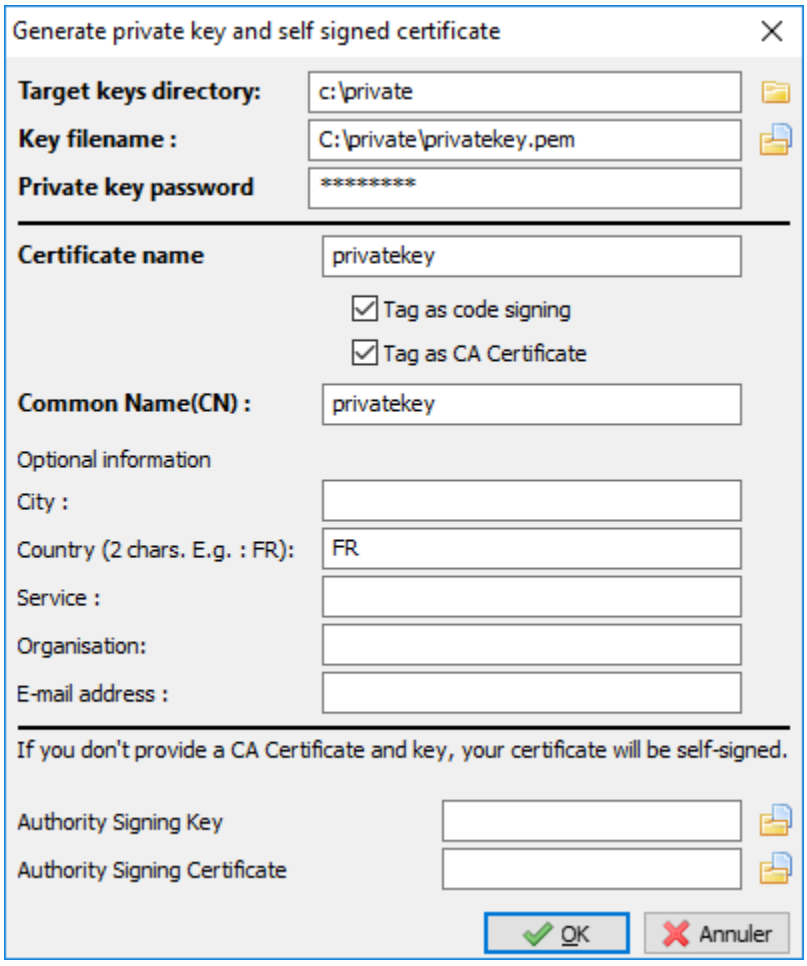


Figure41: Générer un certificat auto-signé

Les informations à renseigner sont :

- *Répertoire de destination* où seront sauvegardés la clé privée et le certificat public : **obligatoire** ;
- *Nom de la clé* donné aux fichiers `.pem` et `.crt` : **obligatoire** ;
- *Mot de passe de la clé* pour vérouiller la clé : **obligatoire** ;
- *Common Name (CN)* pour identifier formellement le signataire : **obligatoire** ;
- *Certificate Name* donné au certificat `.crt` : **obligatoire** ;
- *Code Signing* si le certificat sera utilisé pour signer des paquets logiciels : **obligatoire** ;

- *Certificat CA* si la clé associée au certificat pourra signer d'autres certificats ;
- *Informations optionnelles* : détails supplémentaires stockées dans la clé privée. Ces informations permettront aux destinataires d'un paquet WAPT de connaître son origine ;

Indication: Le mot de passe de la clé privée doit être choisi en conformité avec les préconisations en vigueur dans votre *Organisation* (par exemple les recommandations *ANSSI*).

Note: Si votre Organisation est déjà équipée d'une *Autorité de Certification* (CA), il faudra renseigner le certificat ainsi que la clé dans les champs *Certificat de la CA* et *Clé de la CA*.

Cette étape sera utile lorsque vous souhaiterez générer une nouvelle paire de clé de type *Code Signing* ou de type *Non Code Signing*.

Danger:

- le répertoire de destination de la clé privée ne doit pas être le dossier d'installation de WAPT (`C:\Program Files (x86)\wapt`);
- si votre clé est stockée dans `C:\Program Files (x86)\wapt`, vous risqueriez de déployer votre clé privée sur le réseau lors de la création de l'agent WAPT, **absolument à ne pas faire!**

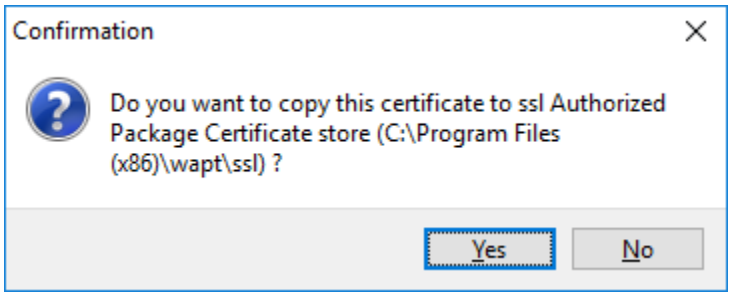


Figure42: Confirmation de copie dans le répertoire ssl

- cliquer sur *Oui* pour copier le certificat nouvellement généré dans le dossier `C:\Program Files (x86)\wapt\ssl`. Ce certificat sera récupéré lors de la compilation de l'agent WAPT et déployé sur les ordinateurs clients ;

Lorsque tout s'est bien déroulé le message suivant apparaît :

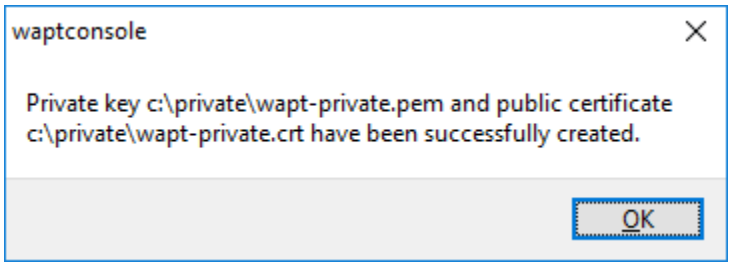


Figure43: Réussite de la génération du certificat

- cliquer sur *OK* pour passer à l'étape suivante ;

Vous pouvez maintenant passer à l'étape suivante et *configurer votre console WAPT*.

6.6.3 Configurer la console WAPT

Nous allons maintenant configurer la console WAPT.

Accédez aux paramètres de configuration de la console en faisant *Outils* → *Préférences*.

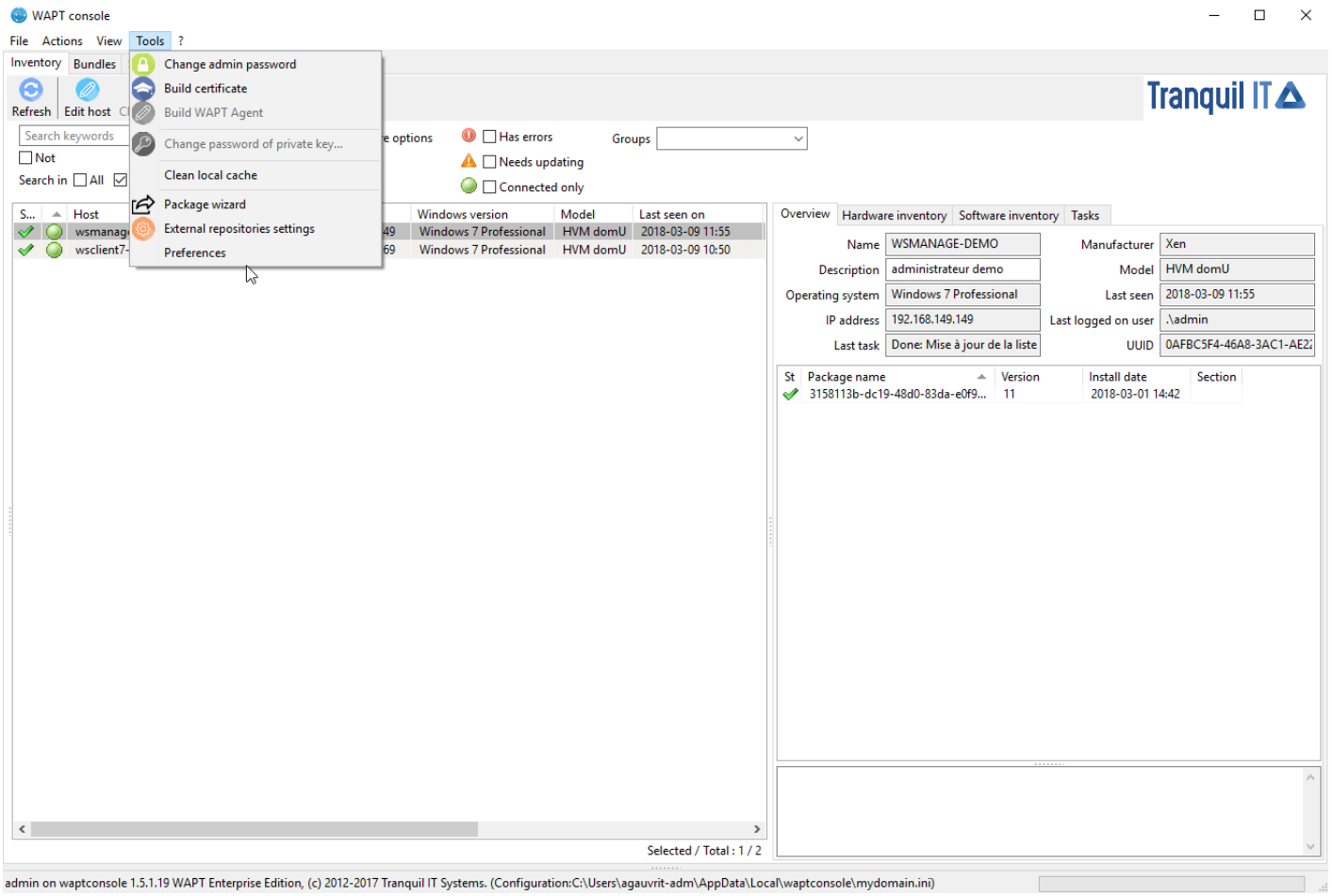


Figure44: Le menu Préférences de la console WAPT

Préférences de la console WAPT

Nous allons paramétrer maintenant le préfixe.

Attention: Il est difficile de changer de préfixe par la suite, donc choisissez-le bien !

Le chemin vers la clé privée a été automatiquement renseigné lors de la création de la clé privée.

Si une clé existe déjà (liée à une installation précédente), il faudra la renseigner ici.

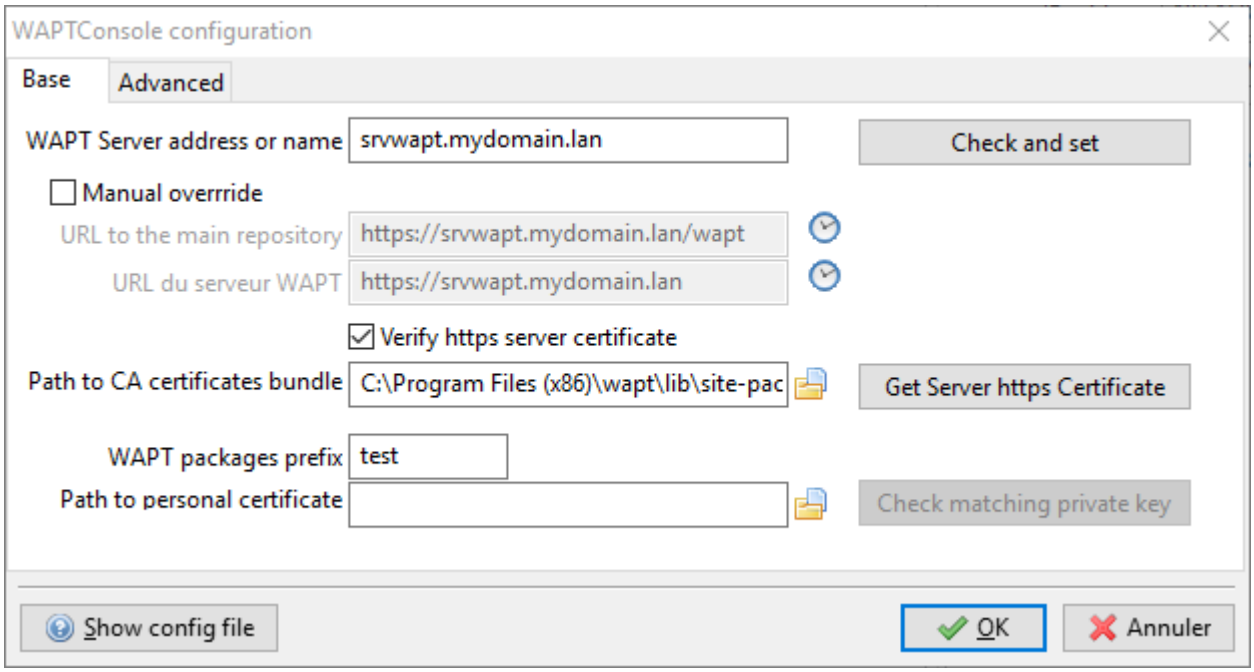


Figure45: Configurer la console WAPT

Si vous souhaitez en savoir plus sur la configuration de la console WAPT, rendez-vous sur : *Configurer la console WAPT*.

Vous pouvez maintenant *créer l'agent WAPT* !!

6.6.4 Générer l'installeur de l'agent WAPT

Choisir le mode d'identification unique des agents WAPT

Dans WAPT, vous pouvez choisir le mode d'identification unique des agents.

Lorsqu'un agent WAPT s'enregistre, le serveur doit savoir s'il s'agit d'une nouvelle machine ou s'il s'agit d'une machine déjà enregistrée.

Pour cela, le serveur WAPT examine le numéro unique « UUID » dans l'inventaire.

Le WAPT propose 3 modes de fonctionnement pour vous aider à distinguer vos machines, c'est à vous de choisir le mode qui vous convient le mieux.

Attention: Lorsqu'on choisit un mode de fonctionnement, il est difficile de le changer, donc réfléchissez le bien !

Identification des agents WAPT par leur BIOS UUID (numéro de série)

Ce mode de fonctionnement permet d'identifier les machines de la console de manière physique.

Si vous remplacez un ordinateur et vous donnez au nouvel ordinateur le même nom que le précédent, vous aurez deux ordinateurs qui apparaîtront dans la console WAPT puisque vous aurez physiquement deux ordinateurs différents.

Note: Certains fabricants ne font pas correctement leur travail et attribuent les mêmes UUID BIOS à des lots entiers d'ordinateurs. Dans ce cas, le WAPT ne verra qu'un seul ordinateur ...

Identification de l'agent WAPT par son nom d'hôte

Ce mode de fonctionnement est similaire à celui d'Active Directory. Les machines sont identifiées par leur nom d'hôte.

Note: Ce mode ne fonctionne pas si plusieurs machines de votre parc portent le même nom. Nous savons tous que cela ne devrait pas arriver !!

Identifier les agents WAPT avec un UUID généré de manière aléatoire

Ce mode de fonctionnement permet d'identifier les PC par leur installation WAPT. Chaque installation de WAPT génère un numéro aléatoire unique. Si vous désinstallez WAPT de l'ordinateur puis vous réinstallez WAPT, alors vous verrez apparaître un nouveau PC dans votre console.

Générer l'installeur de l'agent WAPT

L'installeur **`waptagent`** est un installeur InnoSetup.

Une fois la console WAPT installée sur le poste de l'*Administrateur*, on dispose de tous les fichiers nécessaires pour compiler l'installeur WAPT :

- les fichiers qui seront utilisés pour créer l'installeur se trouvent dans le répertoire d'installation `C:\Program Files (x86)\wapt` ;
- les fichiers sources de l'installeur (fichiers `*.iss`) se trouvent dans `C:\Program Files (x86)\wapt\waptsetup` ;

Indication: Avant de lancer la création de l'installeur pour l'agent WAPT, vérifier la présence du certificat public dans le dossier `C:\Program Files (x86)\wapt\ssl`.

Si vous souhaitez déployer d'autres certificats publics sur les ordinateurs de l'*Organisation* équipés de WAPT, il faudra les copier dans ce dossier.

Danger: **NE PAS COPIER votre clé privée** d'*Administrateur* dans `C:\Program Files (x86)\wapt`.

Ce dossier est utilisé pour créer l'agent WAPT et votre clé privée se retrouverait déployée sur l'ensemble du parc.

- dans la console WAPT, ouvrir *Outils* → *Créer un agent WAPT* ;
- renseigner les informations nécessaires à l'installeur :
  - le champ **obligatoire** *Certificat public* ;

    exemple : `C:\private\mydomain.crt`
  - le champ *Adresse du dépôt WAPT* : **obligatoire** ;

    exemple : https://srvwapt.mydomain.lan/wapt
  - le champ *Adresse du serveur WAPT* : **obligatoire** ;

    exemple : https://srvwapt.mydomain.lan
  - la case à cocher *Vérifier le certificat HTTPS du serveur* ;
  - le champ *Chemin du bundle de certificats* pour vérifier le certificat HTTPS du serveur ;
  - la case à cocher *Utiliser Kerberos pour l'enregistrement* initial des agents WAPT avec le serveur ;
  - le champ *Organisation* pour identifier l'origine des paquets ;
  - le champ *Signer waptupgrade en sha256 et sha1* peut maintenant être ignoré car il n'est utile que lors d'une mise à jour depuis une version 1.3 vers 1.5 ;
  - le champ *Utilise le FQDN de l'ordinateur comme UUID* et *Utilise un UUID aléatoire (for buggy BIOS)* (voir explication dans le paragraphe précédent de cette documentation) ;

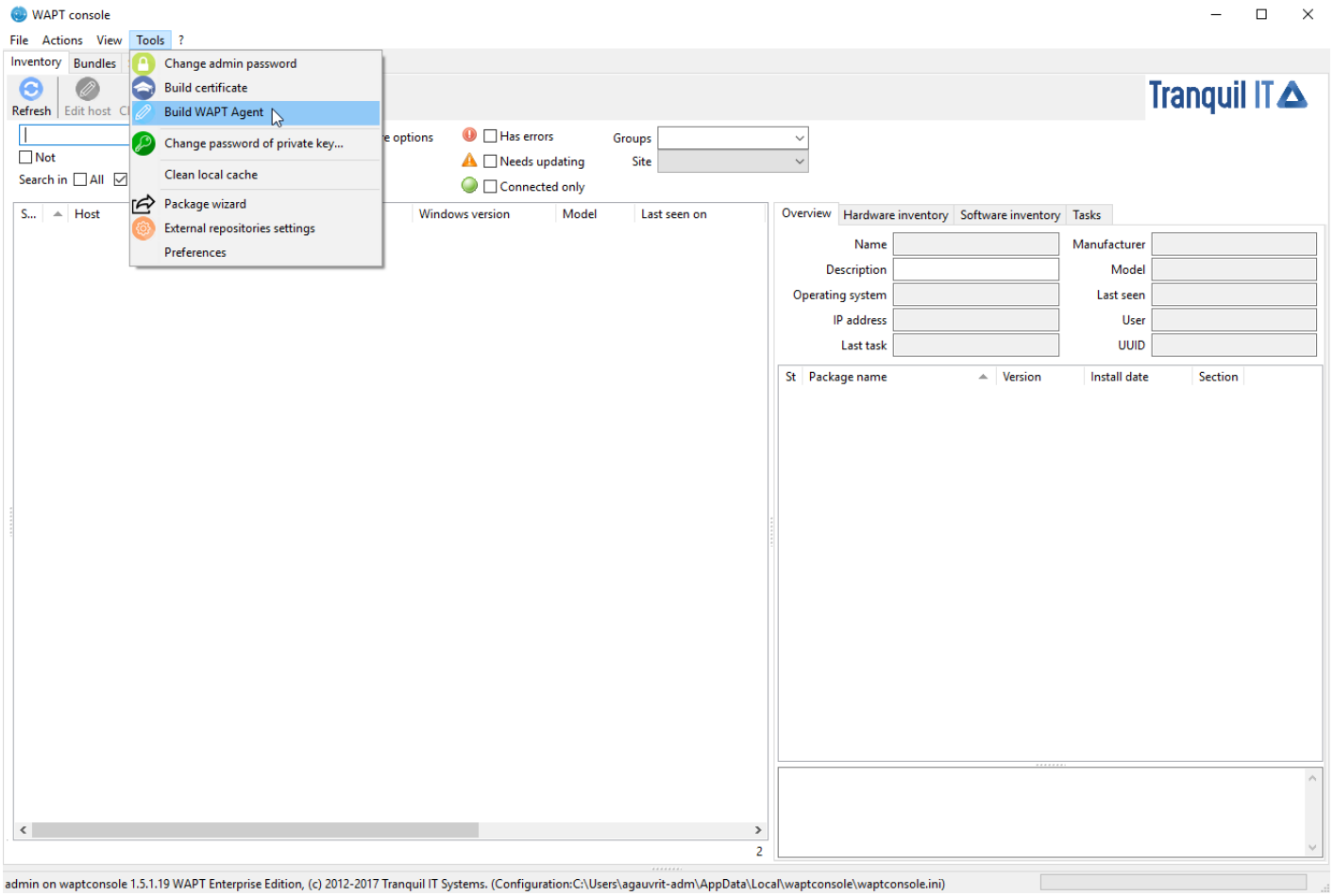


Figure46: Générateur l'agent WAPT à partir de la console

- le champ *Activer les groupes AD* permet l'installation de paquets de profil basés sur les groupes Active Directory dont la machine est membre. **Cette fonctionnalité peut dégrader les performances du WAPT** ;
- le champ *Ajouter au profil de machine* permet de définir une liste de paquets WAPT à installer obligatoirement ;
- le champ *Programmation des audits de paquet* définit la fréquence à laquelle l'agent WAPT vérifie s'il a des audits à effectuer ;
- Pour les mises à jour Windows, voir *cet article sur la configuration de WAPTWUA sur l'agent WAPT* ;

Danger:

- La case **Utiliser l'enregistrement kerberos pour l'enregistrement initial** doit être activé **UNIQUEMENT** si vous avez suivi la documentation **Configurer l'authentification Kerberos**.
- La case **Utiliser l'enregistrement kerberos pour l'enregistrement initial** doit être activé **UNIQUEMENT** si vous avez suivi la documentation **Configurer l'authentification Kerberos**.

- renseignez le mot de passe de la clé privée :

Une fois la compilation terminée, une confirmation s'affiche informant que le **`waptagent`** a bien été chargé sur https://srvwapt.mydomain.lan/wapt/.

Note: Un avertissement apparaît pour rappeler de changer la valeur du hash dans la GPO de déploiement de l'agent WAPT.

Mettre à jour les agents WAPT

Le paquet `test-waptupgrade` a également été chargé sur le dépôt.

Ce paquet WAPT contient l'agent WAPT avec les paramètres spécifiés lors de l'installation.

Note: Ce paquet permettra de mettre à jour les agents WAPT du parc lorsqu'une mise à jour de l'agent WAPT sera rendue disponible.

Mettre à niveau les agents WAPT à l'aide du paquet *xxx-waptupgrade* est un processus en deux étapes :

- le paquet copie d'abord le nouveau fichier `waptagent.exe` sur l'ordinateur client et crée une nouvelle tâche programmée qui exécutera **`waptagent.exe`** deux minutes après la création de la tâche programmée. À ce moment, le paquet lui-même sera installé et l'inventaire sur le serveur indiquera que l'installation du paquet est *OK*, avec la version correcte installée, mais l'inventaire affichera toujours l'ancienne version car l'agent n'aura pas encore été mis à jour.
- après deux minutes, la tâche programmée démarre et exécute **`waptagent.exe`**. **`waptagent.exe`** arrête le service WAPT local, met à niveau l'installation WAPT locale, puis redémarre le service. La tâche programmée est alors automatiquement supprimée et l'agent WAPT renvoie son inventaire au serveur WAPT. L'inventaire sur le serveur affichera alors la nouvelle version de l'agent.

Du point de vue d'un administrateur, en regardant la console, vous verrez les étapes suivantes :

- le paquet *xxx-waptupgrade* commence à être installé ;
- *xxx-waptupgrade* est installé, la machine est à jour du point de vue de la liste des paquets, mais la version dans l'inventaire est toujours l'ancienne version de l'agent WAPT ;
- après deux minutes, l'état de la connectivité de l'ordinateur passe à *déconnecté* pendant que l'agent WAPT est mis à jour ;

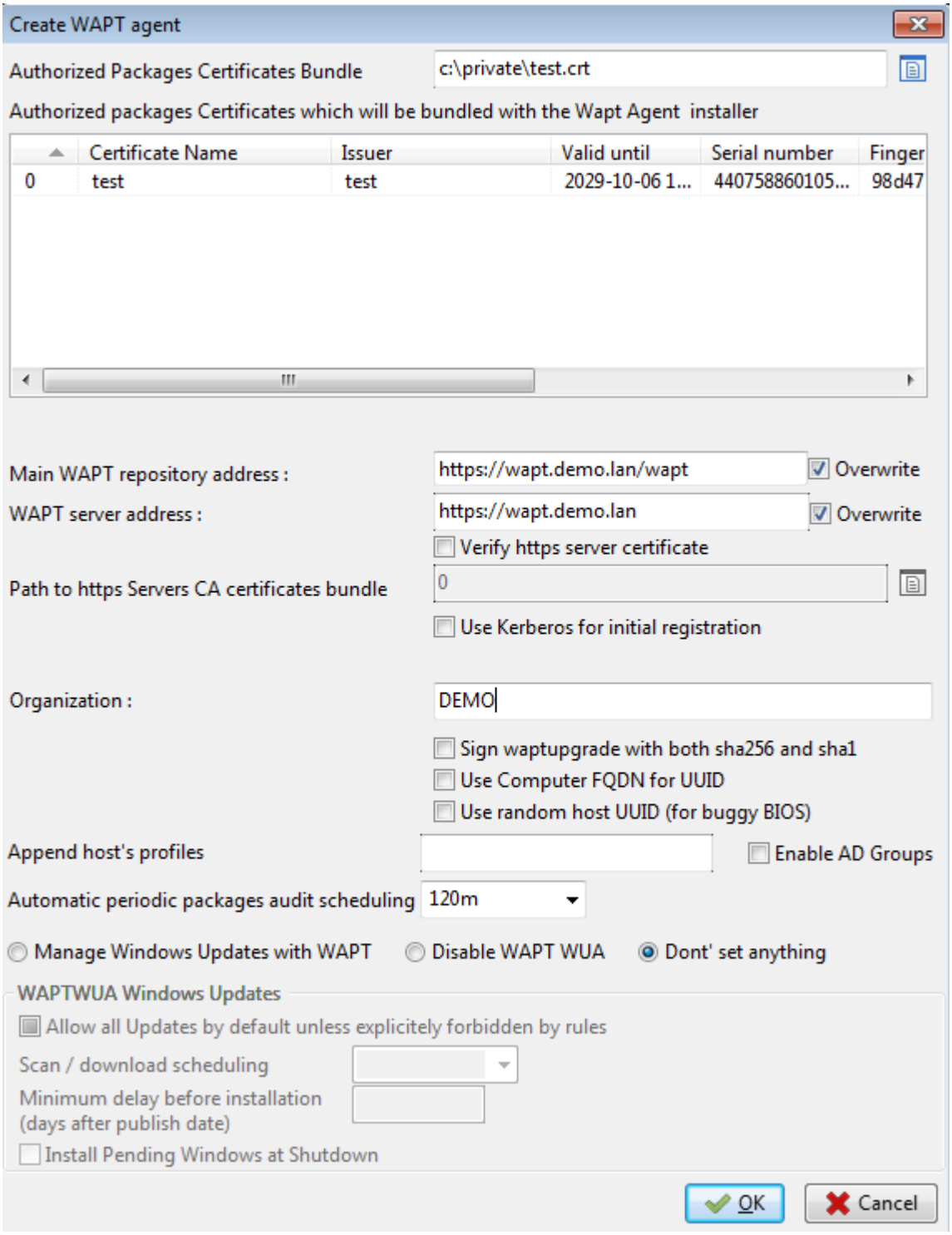


Figure47: Renseigner les informations sur l'Organisation

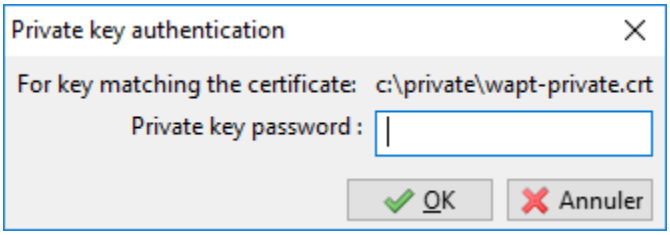


Figure48: Entrer le mot de passe de la clé privée

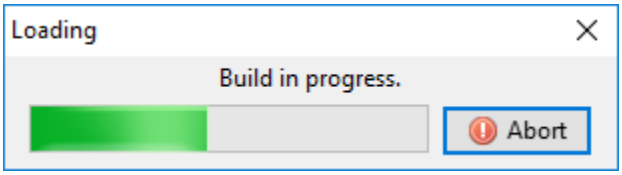


Figure49: Progression de la création de l'installeur de l'agent WAPT

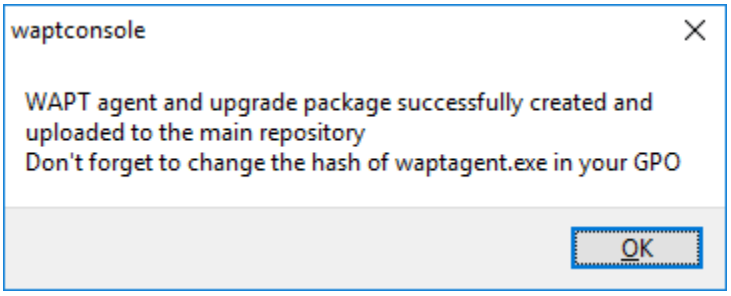


Figure50: Confirmation du chargement de l'agent WAPT sur le serveur WAPT

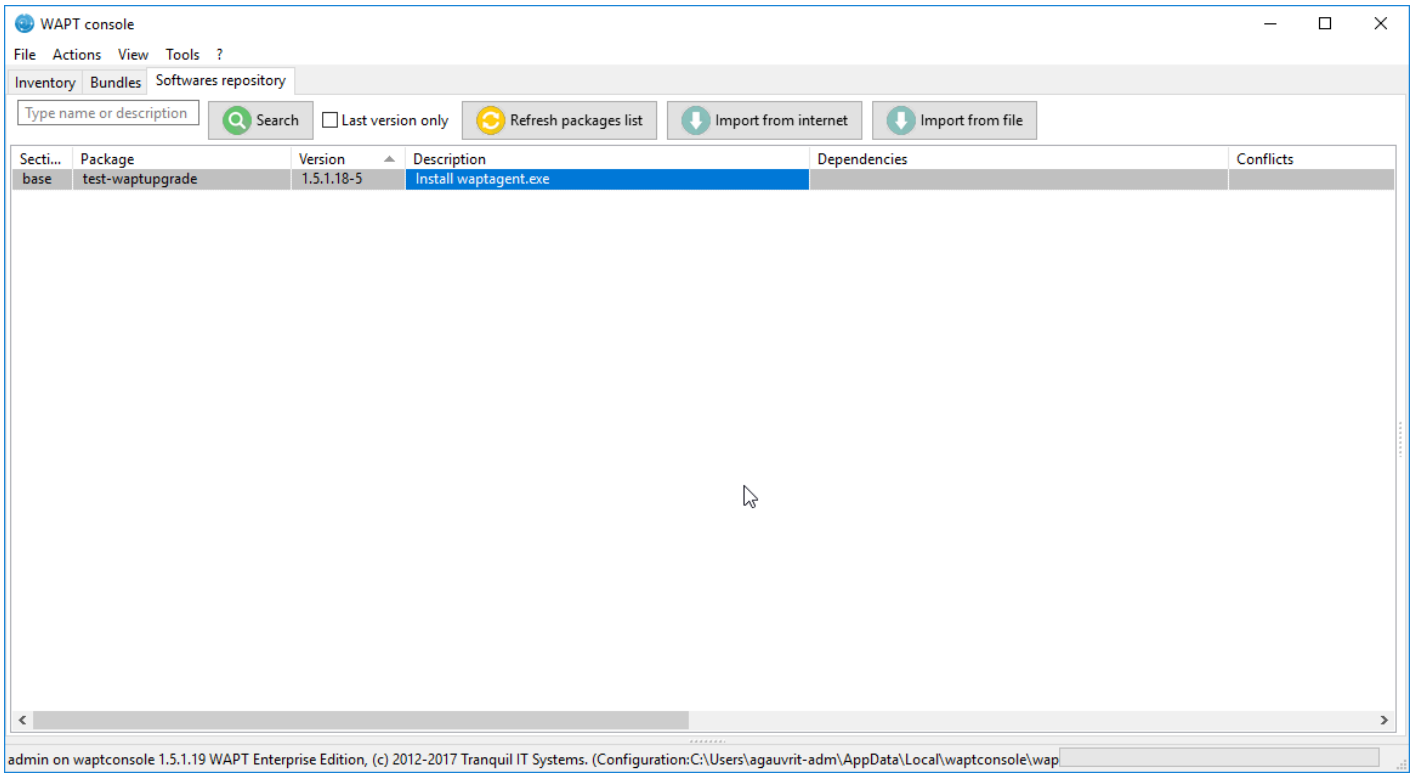


Figure51: Nouvel agent dans le dépôt WAPT

- après environ deux minutes, l'ordinateur client se remet en ligne dans la console, met à jour son inventaire et affiche la nouvelle version ;

Ce qui peut mal se passer lors des mises à niveau

- le problème le plus courant lors du processus de mise à niveau est le blocage de l'installation par un antivirus local (le WAPT est un installateur de logiciels qui maintient une fenêtre Web ouverte sur un serveur de gestion central, ce comportement peut donc être signalé comme suspect par un antivirus, même si cette méthode est à la base de la gestion des points finaux...). **Si vous avez un problème lors du déploiement de la mise à jour, veuillez vérifier votre console antivirus et mettre en liste blanche le waptagent.exe**. Une autre option est de re-signer le binaire **`waptagent.exe`** si votre organisation possède un certificat de signature de code interne ;
- le deuxième problème le plus courant est que, pour certaines raisons, un autre programme verrouille un *DLL* qui est livré avec le WAPT. Cela peut arriver avec des installateurs de logiciels mal conçus qui commencent par prendre la variable locale %PATH% et qui trouvent ensuite des DLL openssl ou python propres embarquées avec WAPT ;
- le troisième problème le plus courant est une installation de Windows défectueuse qui n'exécute pas correctement les tâches programmées, et oui nous avons déjà vu ça !!

6.6.5 Déployer l'agent WAPT

Deux méthodes sont possibles pour déployer l'utilitaire **`waptagent.exe`**.

La première est manuelle et la procédure doit être appliquée sur chaque machine.

La deuxième est automatique et s'appuie sur les GPO.

Note: L'installeur **`waptsetup.exe`** est disponible à l'adresse suivante : https://srvwapt.mydomain.lan/wapt/waptagent.exe.

Si vous ne signez pas le fichier `waptagent.exe` avec un certificat `Code Signing` commercial ou bien issu par l'*Autorité de Certification* de votre Organisation après l'avoir généré, les navigateurs web provoqueront un message d'alerte lors du téléchargement. Pour supprimer ce message d'alerte, il faut signer le fichier `.exe` avec un certificat `Code Signing` reconnu par le bundle de CA installé dans la magasin de certificats de la machine.

Indication: Dans quelles situations déployer WAPT manuellement ?

La méthode manuelle de déploiement est efficace dans les situations suivantes :

- tester WAPT ;
- maintenir à jour un petit parc de PC dans une petite entreprise, une association, etc ;

Déployer manuellement l'agent WAPT

Attention: Cette manipulation requiert les droits d'*Administrateur Local* sur la machine.

Installer *waptagent.exe*

- choisir la langue pour l'installation puis cliquer sur *Suivant* pour passer à l'étape suivante ;

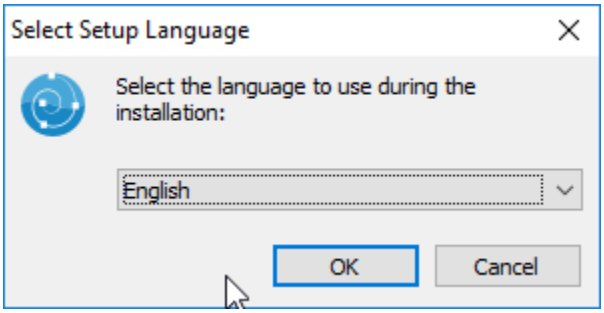


Figure52: Choisir la langue pour l'installation

- accepter la licence puis cliquer sur *Suivant* pour passer à l'étape suivante ;

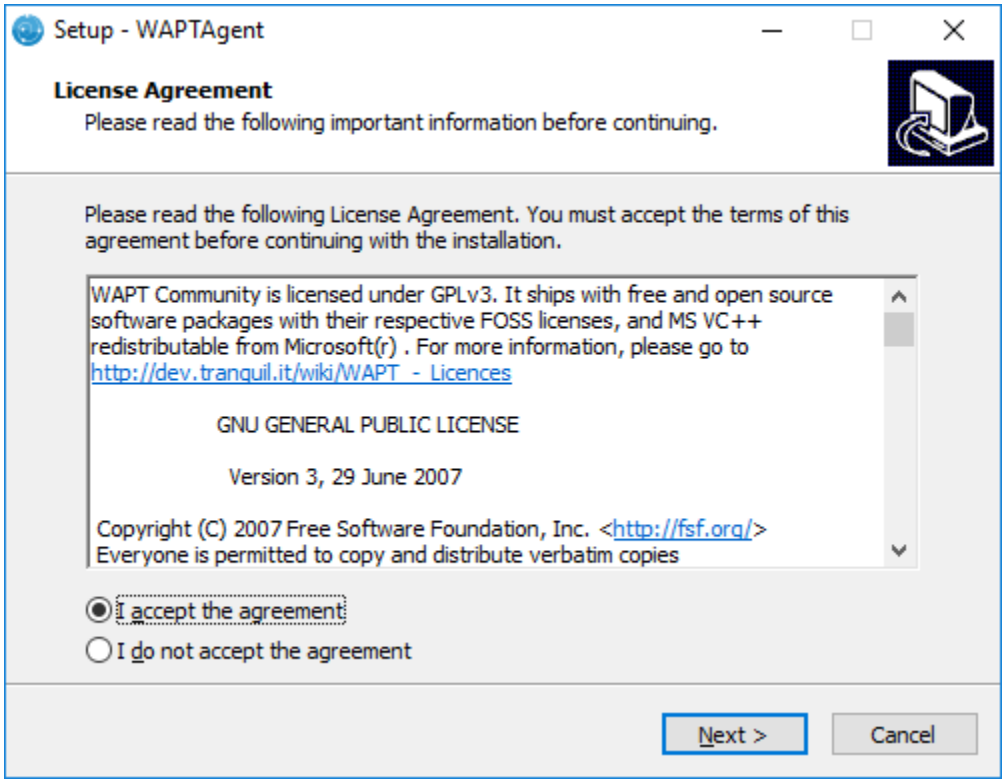


Figure53: Accepter la licence

- choisir le répértoire d'installation puis cliquer sur *Suivant* pour passer à l'étape suivante ;

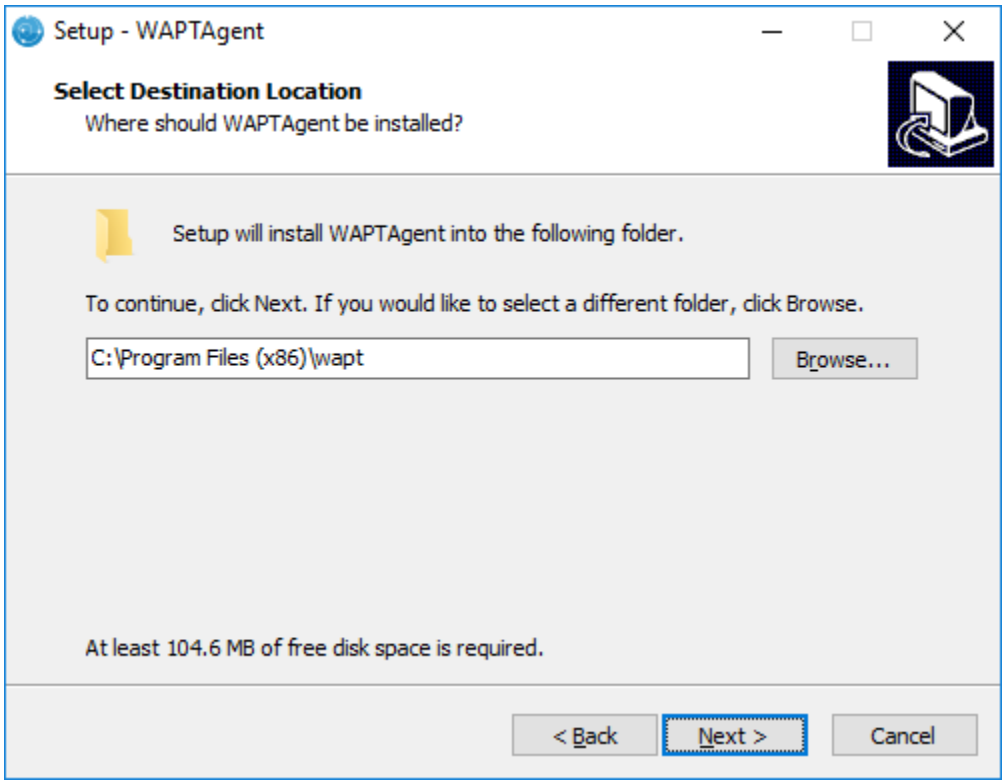


Figure54: Sélectionner le répertoire d'installation de l'agent WAPT

- choisir les tâches complémentaires puis cliquer sur *Suivant* pour passer à l'étape suivante ;

Indication: laisser coché *Force-reinstall VC++*. Si l'option est cochée, c'est que l'installation du composant est nécessaire.

- choisir le dépôt et le serveur puis cliquer sur *Suivant* pour passer à l'étape suivante ;
- installer l'agent WAPT en cliquant sur *Installer* ;
- attendre la fin de l'installation de l'agent WAPT puis cliquer sur *Fin* pour terminer ;

L'installation est terminée, avec **`cmd.exe`**, lancer un **`register`** pour enregistrer la machine sur le serveur WAPT et un **`update`** pour afficher la liste des paquets WAPT disponibles.

Note:

- cocher *Enregistrer cet ordinateur dans le serveur WAPT* pour enregistrer l'ordinateur dans l'inventaire central WAPT ;
- cocher *« Mettre à jour la liste des paquets disponibles sur le dépôt principal* pour télécharger la liste des paquets disponibles ;

Pour gérer les client WAPT, voir *Comment utiliser la console WAPT*.

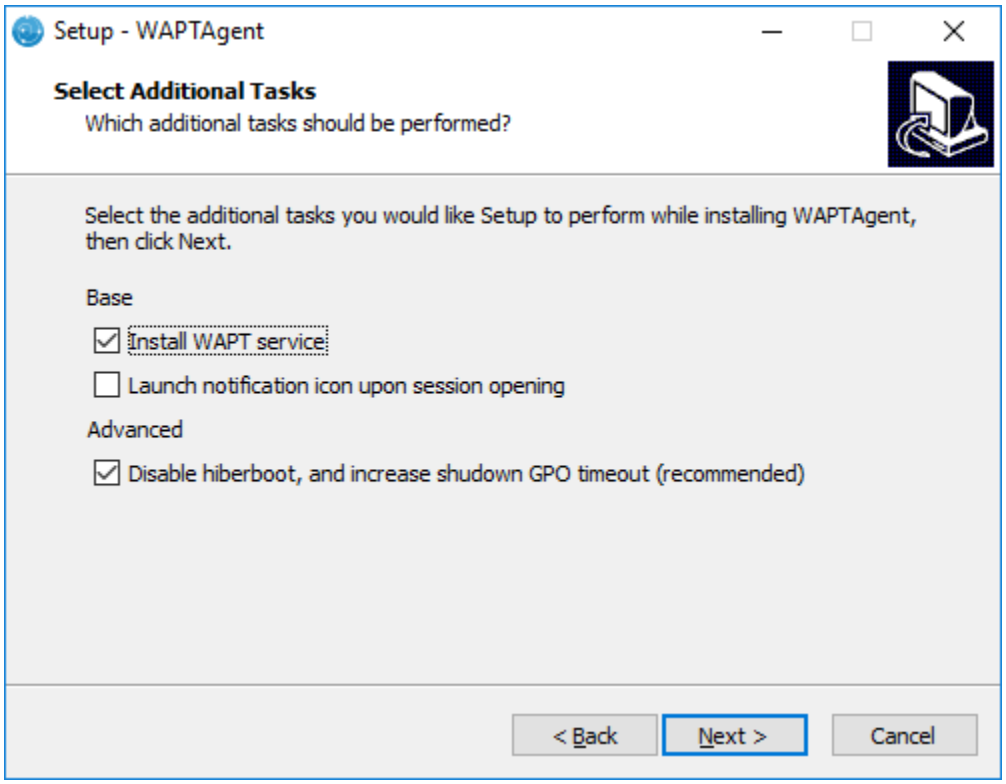


Figure55: Choisir les options d'installation

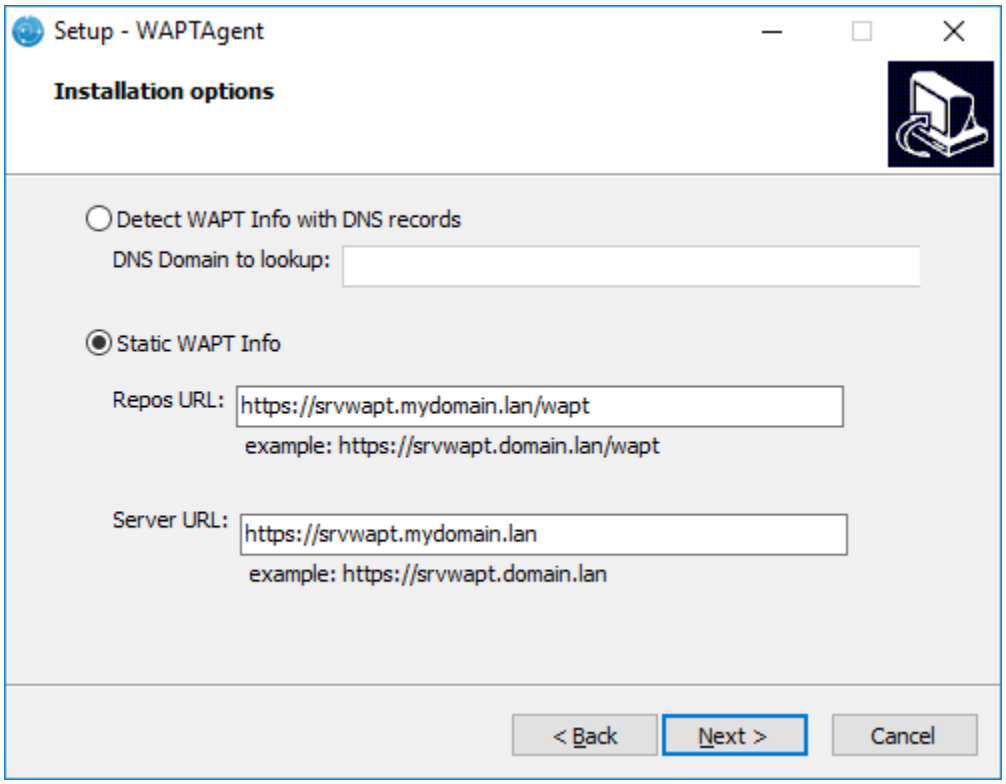


Figure56: Choisir le dépôt et le serveur WAPT

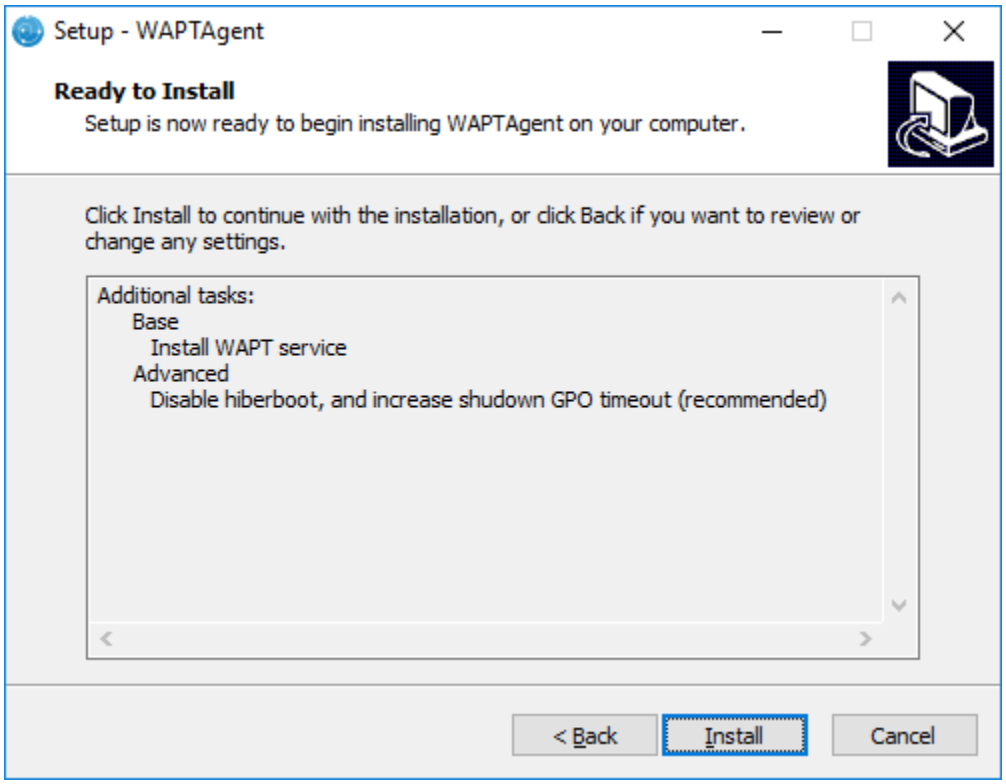


Figure57: Fenêtre récapitulative des options d'installation

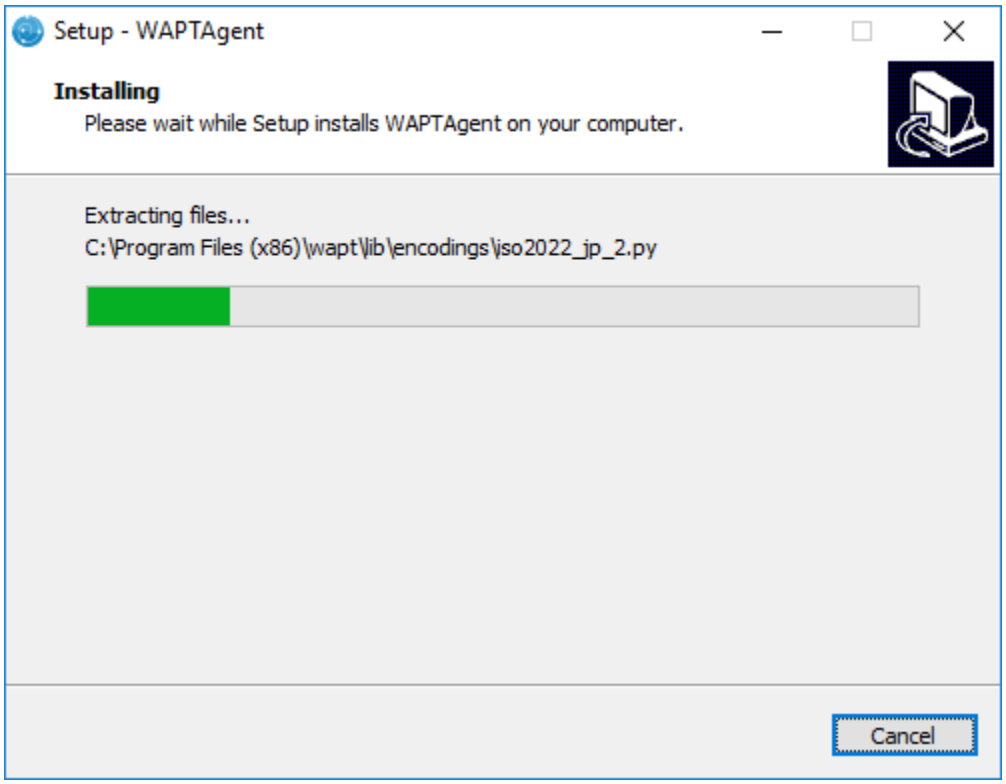


Figure58: Progression de l'installation

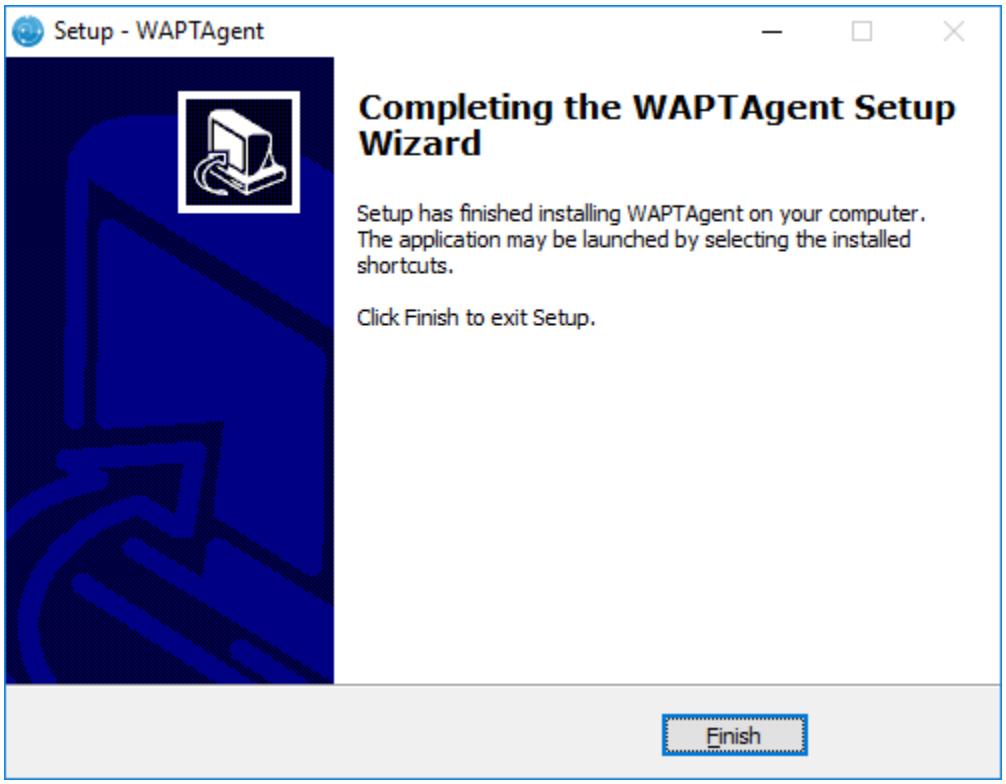


Figure59: Fin de l'installation de l'agent WAPT

Déployer automatiquement les agents WAPT

Important: Pré-requis techniques

Des connaissances avancées en administration de réseau et de système sont nécessaires pour réaliser cette procédure. Un réseau correctement configuré assurera son succès.

Indication: La méthode décrite dans cette section est particulièrement efficace dans les situations suivantes :

- vous gérez un grand parc de machines ;
- vous disposez d'un domaine SaMBa Active Directory ou Microsoft Active Directory pour lequel vous avez des droits d'administration suffisants ;
- la sécurité et la traçabilité des actions est importante pour vous ou pour votre *Organisation* ;
- ou bien tout simplement, vous préférez agir avec votre tête plutôt qu'avec vos pieds ;) ;

Déployer l'agent WAPT silencieusement

Sans waptdeploy

Le **waptagent** est un installeur *InnoSetup*, il peut donc être déployé silencieusement avec les arguments suivants :

```
waptagent.exe /VERYSILENT
```

- Arguments supplémentaires disponibles pour waptdeploy

Table2: Description des options disponibles pour déployer l'agent WAPT silencieusement

Options	Description
`/dnsdomain` = mydomain.lan	Domaine dans le fichier `wapt-get.ini` rempli lors de l'installation.
`/wapt_server` = https://srvwapt.mydomain.lan	URL du serveur WAPT dans `wapt-get.ini` complétée lors de l'installation
`/repo_url` = https://repo1.mydomain.lan/wapt	URL du dépôt WAPT dans `wapt-get.ini` complétée lors de l'installation.
`/StartPackages` = basic-group	Groupe de paquets WAPT à installer par défaut.
`/verify_cert=` = 1 or relative path `ssl\server\srvwapt.mydomain.lan.crt`	Valeur de `verify_cert` entrée lors de l'installation
`/CopyServersTrustedCA` = path to a bundle to copy to `ssl\server`.	Ensemble de certificats pour les connexions https (à définir par `verify_cert`)
`/CopypackagesTrustedCA` = path to a certificate bundle to copy into `ssl`	Bundle de certificats pour la vérification des signatures de paquets

Indication: Le fichier `iss` pour l'installeur InnoSetup est situé ici C:\Program Files (x86)\wapt\ .

Vous pouvez l'adapter à vos besoins spécifiques. Une fois modifié il suffira de relancer la création d'un **`waptagent`**.

Pour voir les nombreuses options disponibles avec *InnoSetup*, rendez-vous sur la documentation InnoSetup.

Avec waptdeploy

Le **`waptdeploy`** est un petit exécutable qui :

- teste la version installée de WAPT ;
- télécharge par https l'installeur **`waptagent.exe`** ;
- lance l'installeur en mode silencieux avec les options sélectionnées à la création du waptagent ;

```
/VERYSILENT /MERGETASKS= ""useWaptServer""
```

- mettre à jour le serveur avec le statut de la machine (version WAPT, situation des packages) ;

Note: **`waptdeploy`** doit être lancé en tant qu'*Administrateur Local*, d'où l'utilisation d'une GPO.

Créer la GPO de déploiement avec WAPTdeploy

Télécharger **`waptdeploy.exe`** ici : https://wapt.tranquil.it/wapt/releases/latest/waptdeploy.exe.

Créer la GPO

- créer une nouvelle stratégie de groupe **install_wapt** sur le serveur Active Directory (Microsoft ou SaMBa-AD) ;
- ajouter la stratégie : *Configuration Ordinateur* → *Stratégies* → *Paramètres Windows* → *Scripts* → *Démarrage* → *Ajouter* ;

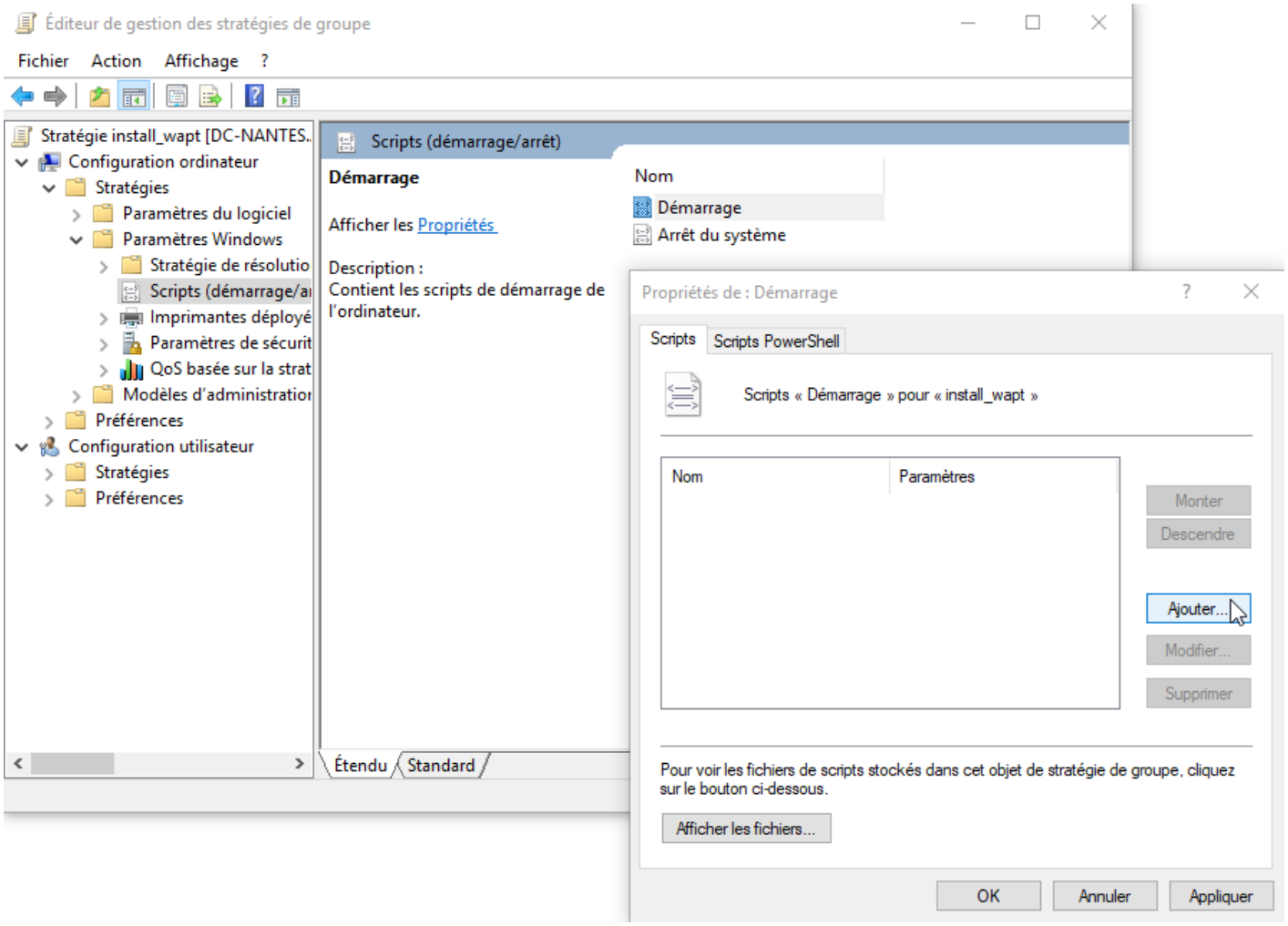


Figure60: Créer une stratégie de groupe pour déployer l'agent WAPT

- cliquer sur *Parcourir* pour sélectionner le script **`waptdeploy.exe`** ;
- copier **`waptdeploy`** dans le dossier ;
- cliquer sur *Ouvrir* pour importer le script :program`waptdeploy.exe` ;
- cliquer sur *Ouvrir* pour confirmer l'importation du script **`waptdeploy`** ;

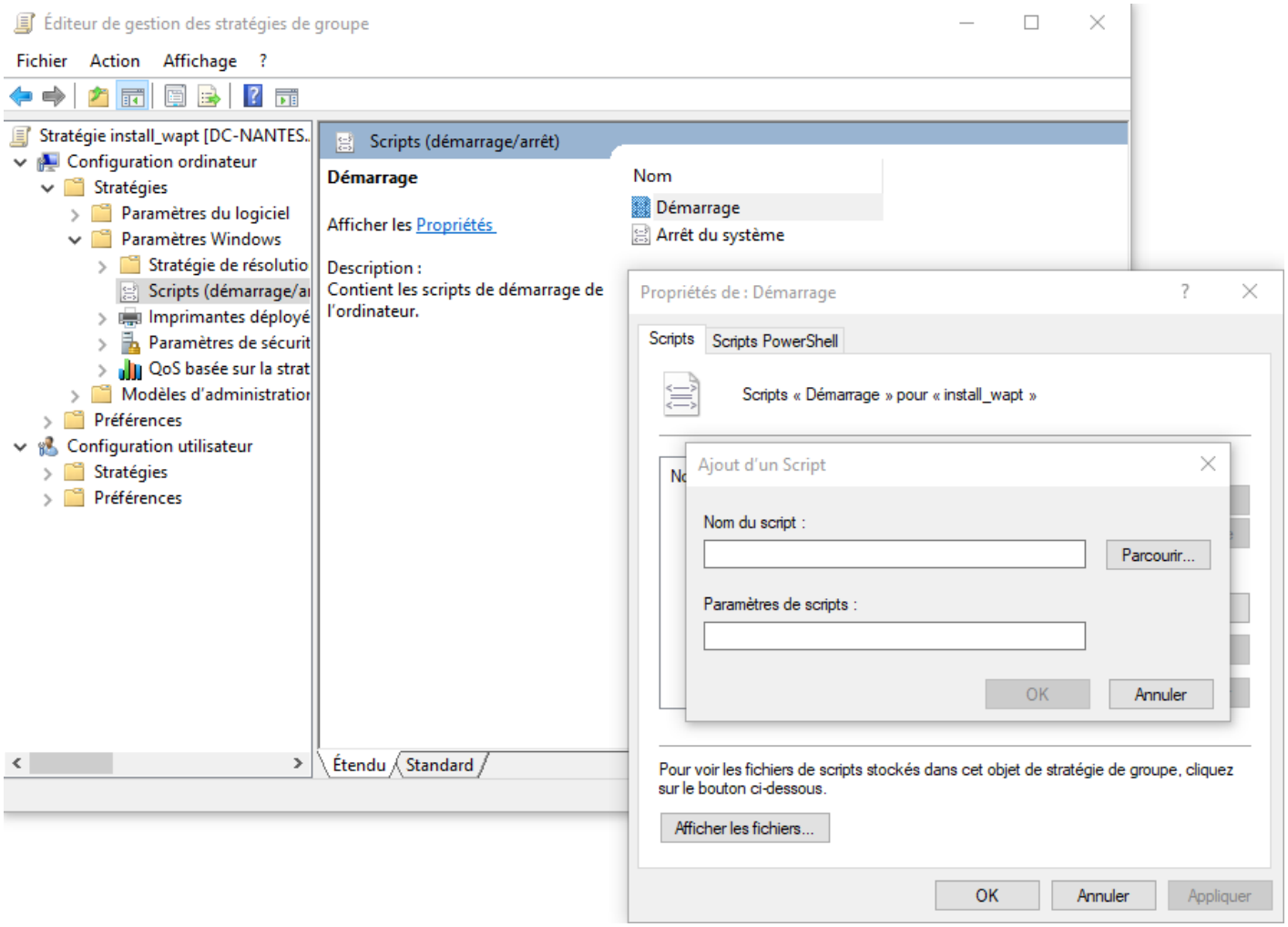


Figure61: Rechercher le script waptdeploy.exe

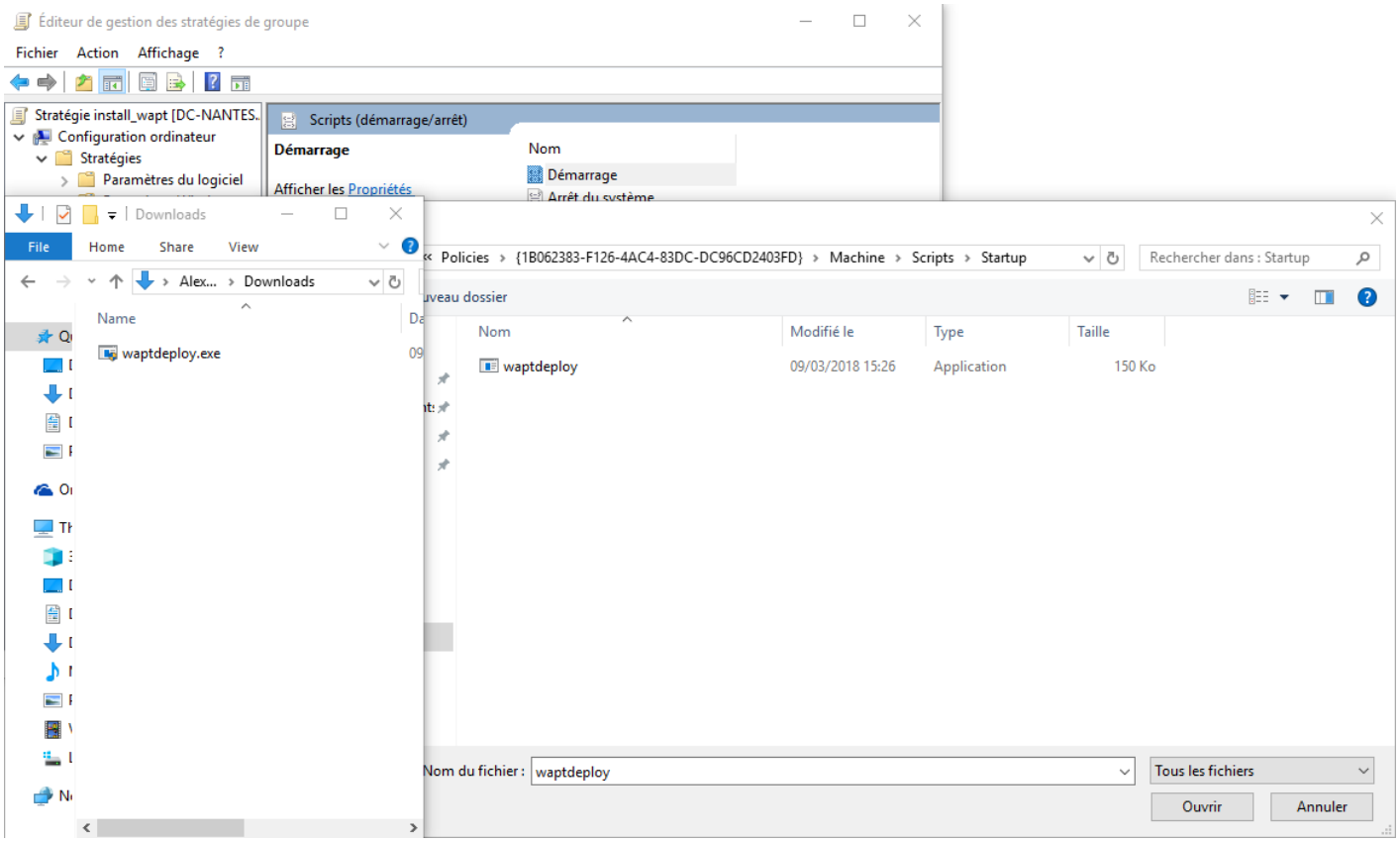


Figure62: Sélectionner le script waptdeploy.exe

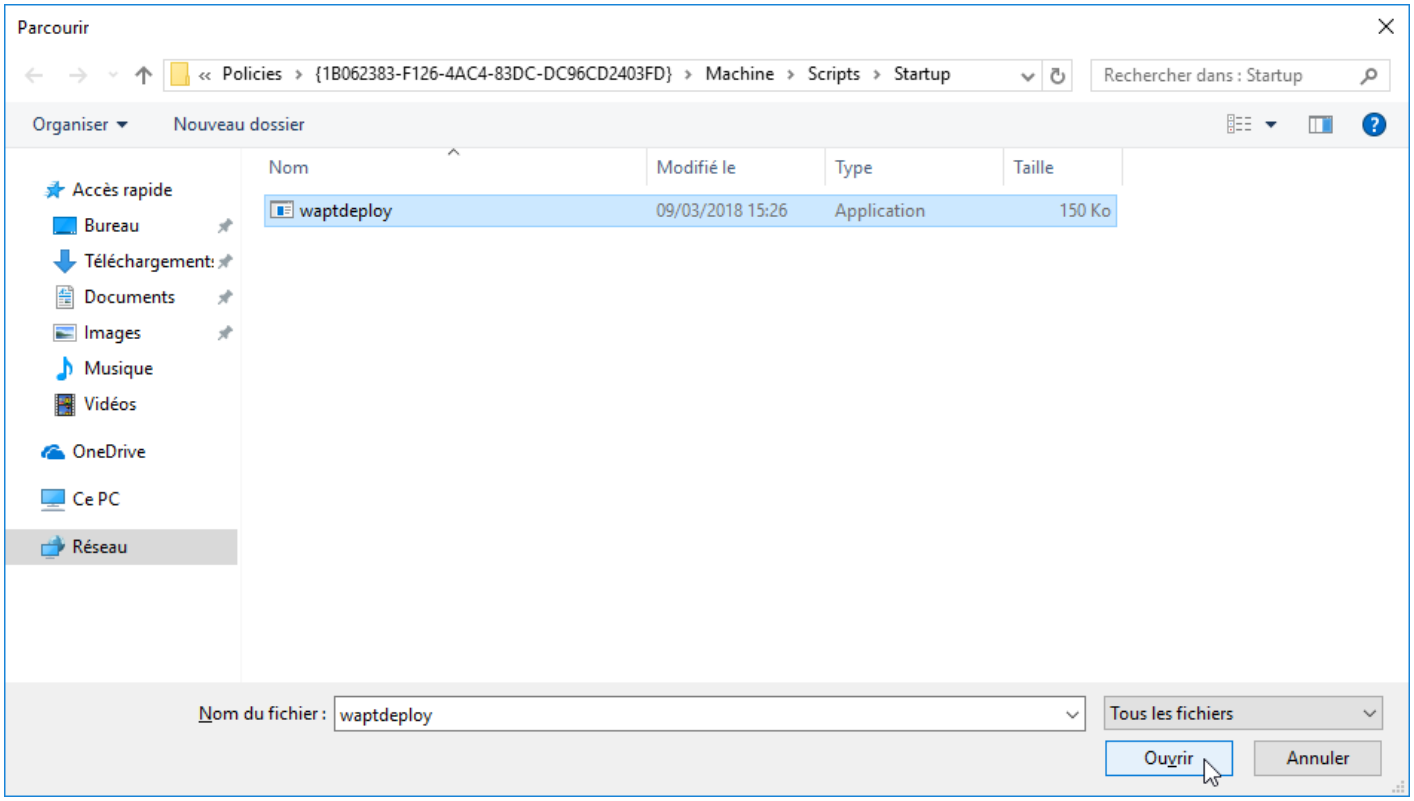


Figure63: Sélectionner le script waptdeploy.exe

Renseigner les arguments

Indication: À partir de la version 1.3.7, il est nécessaire de placer un checksum dans les paramètres de scripts.

Ceci permet de s'assurer que le **waptdeploy** ne lancera pas un installeur falsifié ou corrompu.

```
--hash="checksum du WaptAgent"--minversion=1.5.1.23 --wait=15
```

Note: Les paramètres complets à utiliser for the *waptdeploy* ainsi que le checksum du **waptagent.exe** sont désormais disponibles sur la page d'accueil du serveur WAPT local à l'adresse https://srvwapt.mydomain.lan.

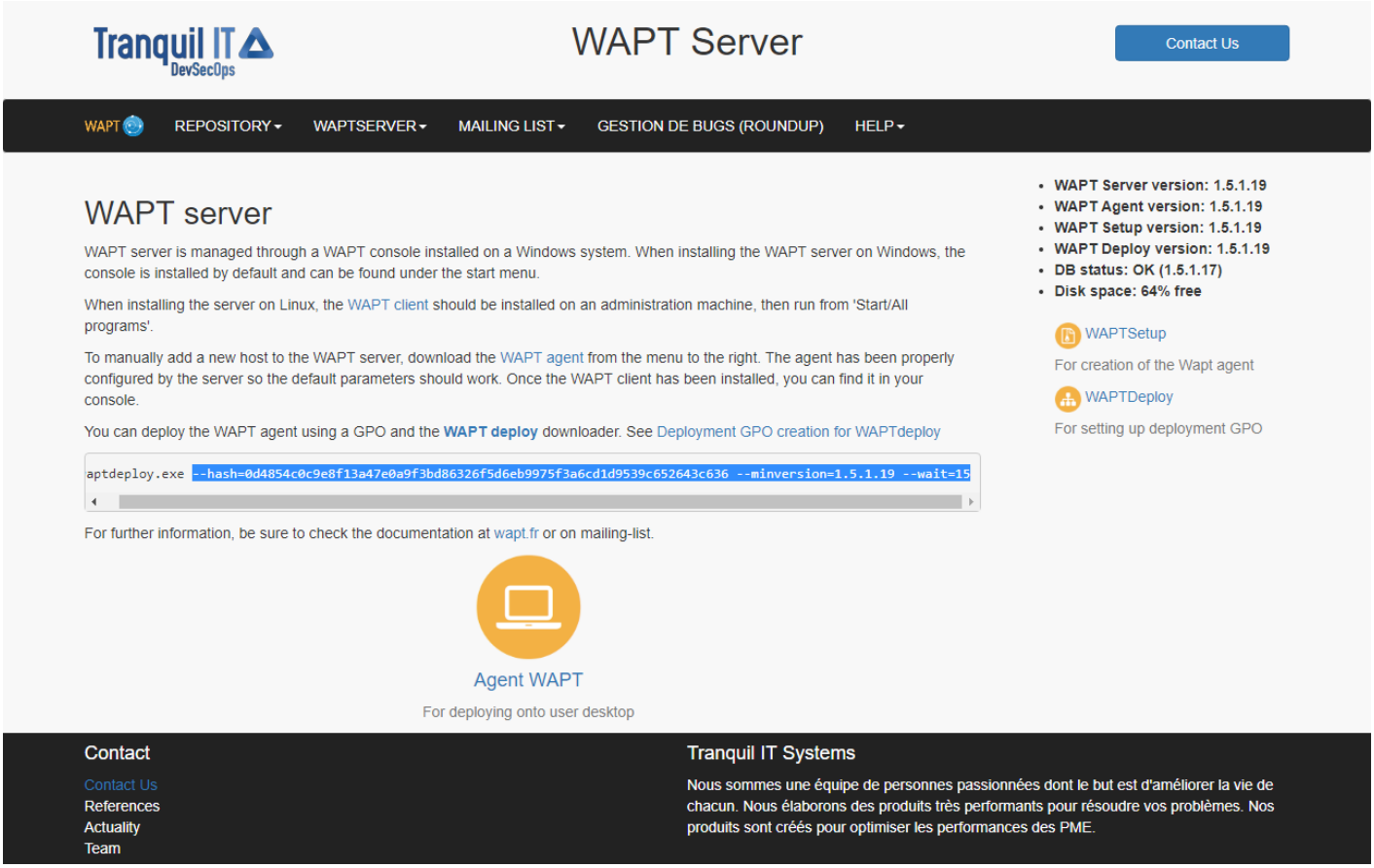


Figure64: Console Web du serveur WAPT

- copier les paramètres indiqués ;
- cliquer sur *OK* pour passer à l'étape suivante ;
- cliquer sur *OK* pour passer à l'étape suivante ;

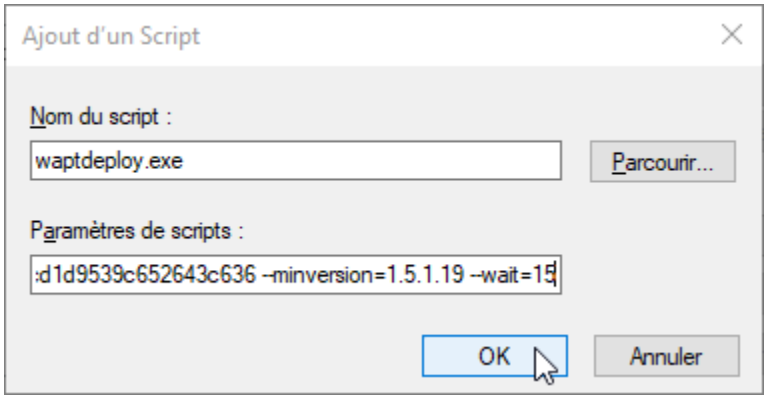


Figure65: ajouter le script *waptdeploy* à la GPO de démarrage

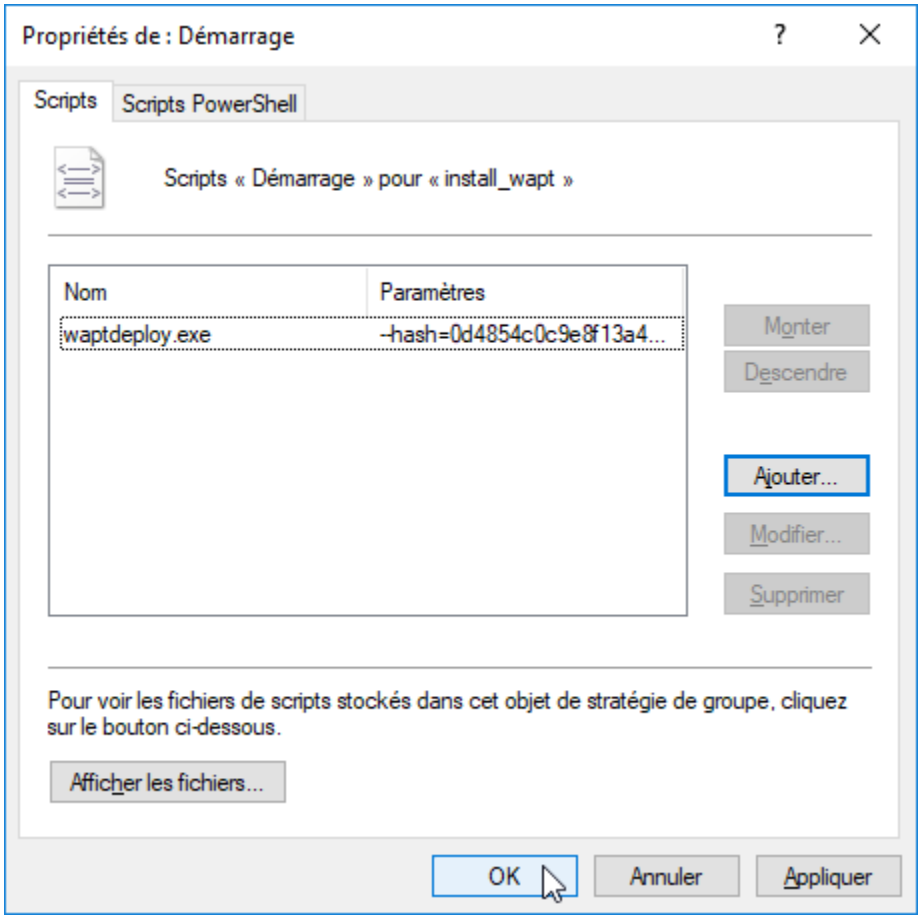


Figure66: GPO WAPTdeploy to be deployed on next startup

- appliquer la stratégie sur l'OU qui contient les ordinateurs de l''UO (:TERM:`UNITÉ ORGANISATIONNELLE (Group Policy Objects / Stratégies de Groupes))` ;

Arguments supplémentaires disponibles pour waptdeploy

Table3: Arguments supplémentaires disponibles pour waptdeploy

Options	Valeur	Description
`--force`		Force l'installation de **waptagent.exe** même si l'agent WAPT est déjà installé.
`--waptsetupurl`	https://srvwapt.mydomain.lan/wapt/waptagent.exe	Donne explicitement le chemin / l'url de téléchargement de l'agent à installer
`--tasks`	autorunTray,installService,installredist2008,autoUpgradePolicy	Indique les tâches pour le programme d'installation **waptagent**. Par défaut, toutes les tâches sont activées
`--wait`	10	Timeout pour l'installation de l'agent WAPT.
`--setupargs=`	/dnsdomain=mydomain.lan /wapt_server= /repo_url=	Passer des arguments supplémentaires à **waptagent**

```
--hash="43254648348435423486"--minversion=1.8.1 --waptsetupurl=http://srvwapt.mydomain.lan/wapt/
↪waptagent.exe --wait=10
```

Lancer waptdeploy avec une tâche planifiée

Pour un bon fonctionnement de **waptdeploy**, vous pouvez aussi exécuter la même GPO à l'arrêt de l'ordinateur ;

Vous pouvez également lancer **waptdeploy** avec une tâche planifiée installée par GPO.

Indication: Ce mode de fonctionnement est particulièrement efficace pour déployer WAPT sur des Postes Client où le réseau n'est ni disponible au démarrage ni disponible à l'arrêt.

La méthode consiste à utiliser une GPO « Copie de fichier », copiez les fichiers `waptdeploy.exe` et `waptagent.exe` :

- Source : `\mydomain.lan\netlogon\waptagent.exe` Destination : `C:\windows\temp\waptagent.exe`
- Source : `\mydomain.lan\netlogon\waptagent.exe` Destination : `C:\windows\temp\waptagent.exe`
- copier `waptdeploy.exe` et `waptagent.exe` dans le partage netlogon sur votre serveur Active Directory.
- créer ensuite une GPO de tâche planifiée qui lancera **waptdeploy** :

  ```
  C:\windows\temp\waptdeploy.exe
  ```

  Arguments :

  ```
  --hash="43254648348435423486"--minversion=1.5.1.23 --waptsetupurl=C:\windows\temp\waptagent.
  ↪exe --wait=10
  ```

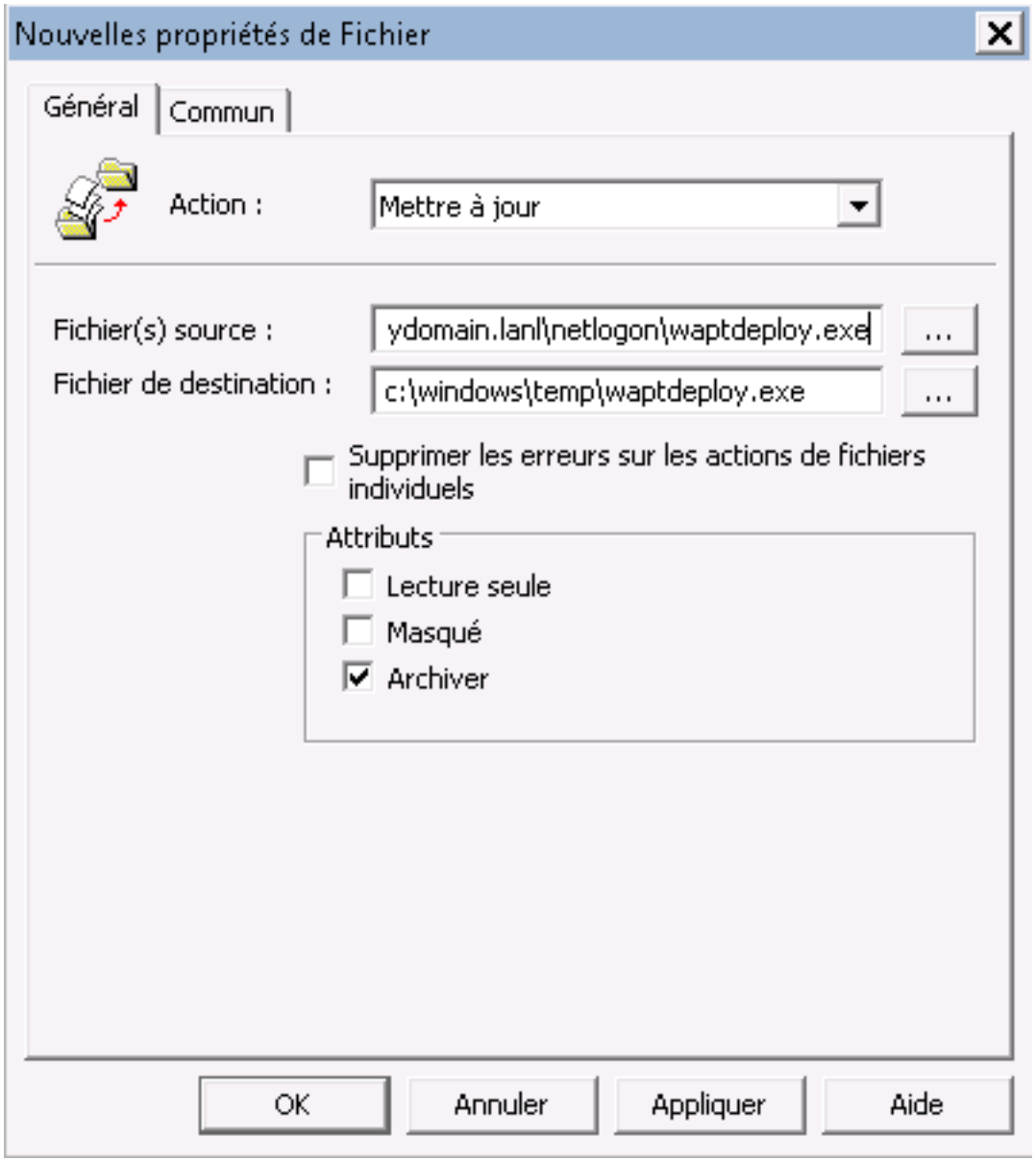


Figure67: Propriétés affichant la copie des fichiers

Attention: Les arguments « hash » et « min_version » changeront en réalité par rapport à cette documentation, à mesure que WAPT continuera de s'améliorer.

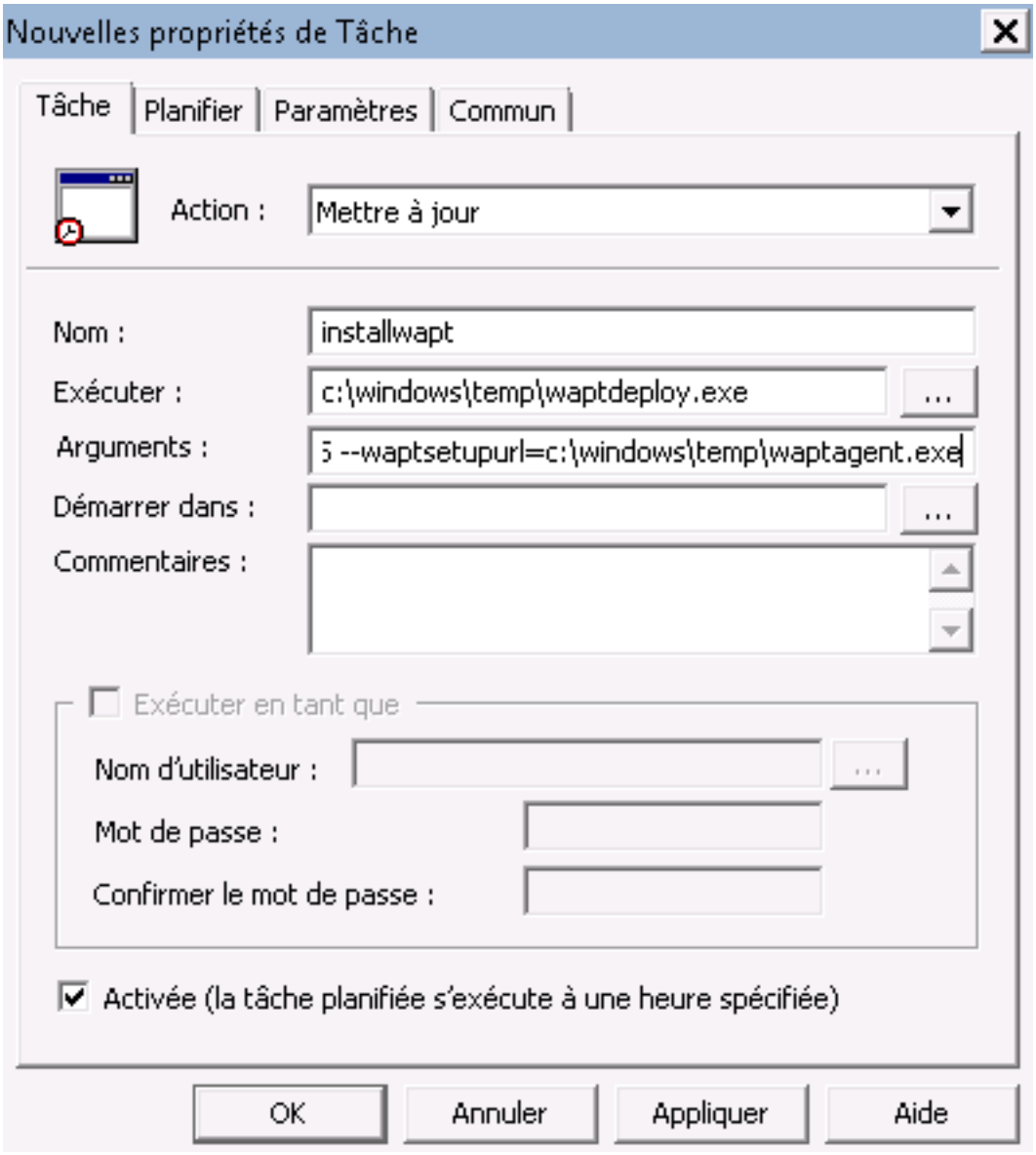


Figure68: Propriétés de la tâche d'installation

- préciser ensuite une heure de lancement et préciser que la tâche doit être répétée toutes les 30 minutes ou jusqu'à sa réussite :
- autorise que la tâche puisse démarrer même sur batterie :

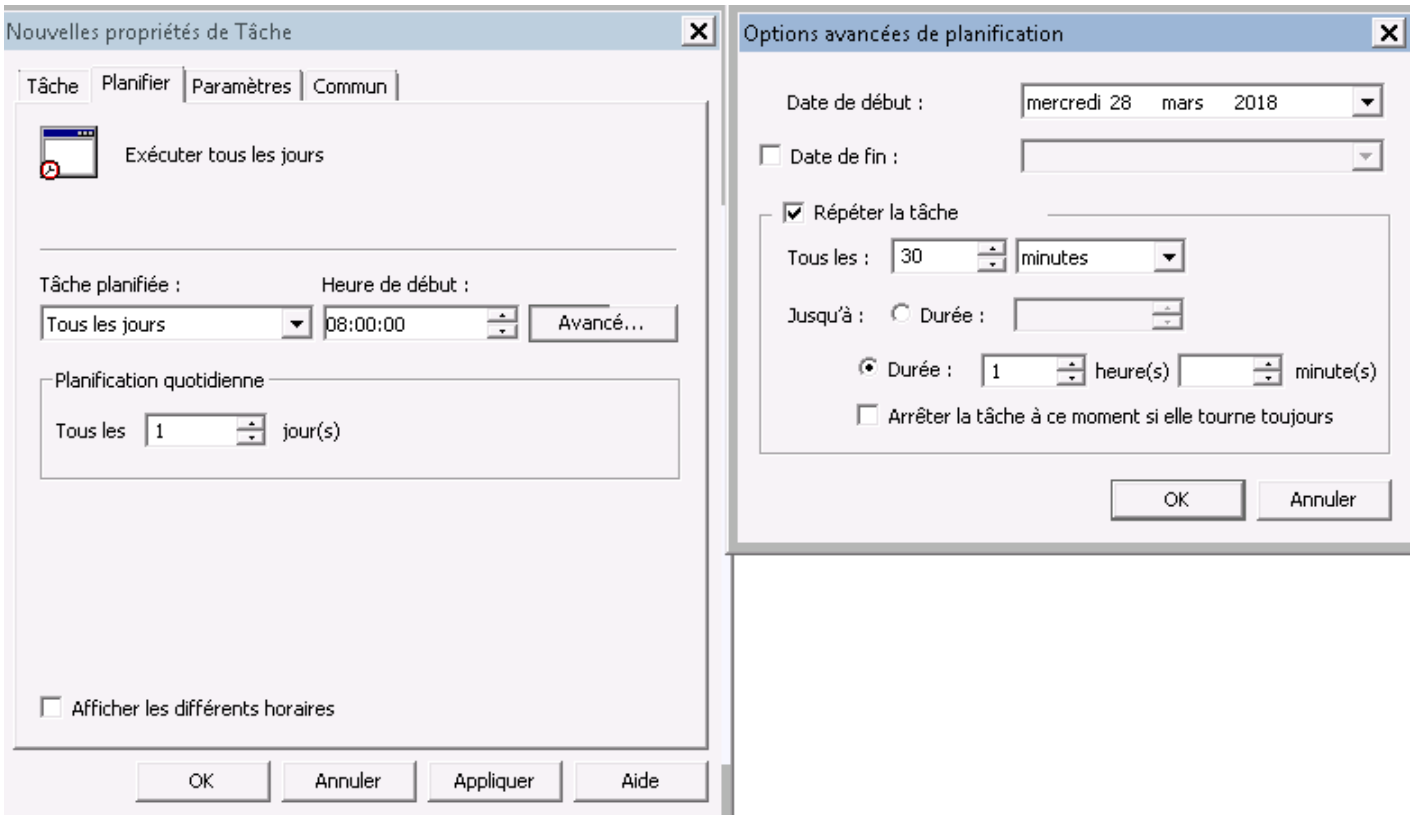


Figure69: Propriétés avancées de la tâche d'installation

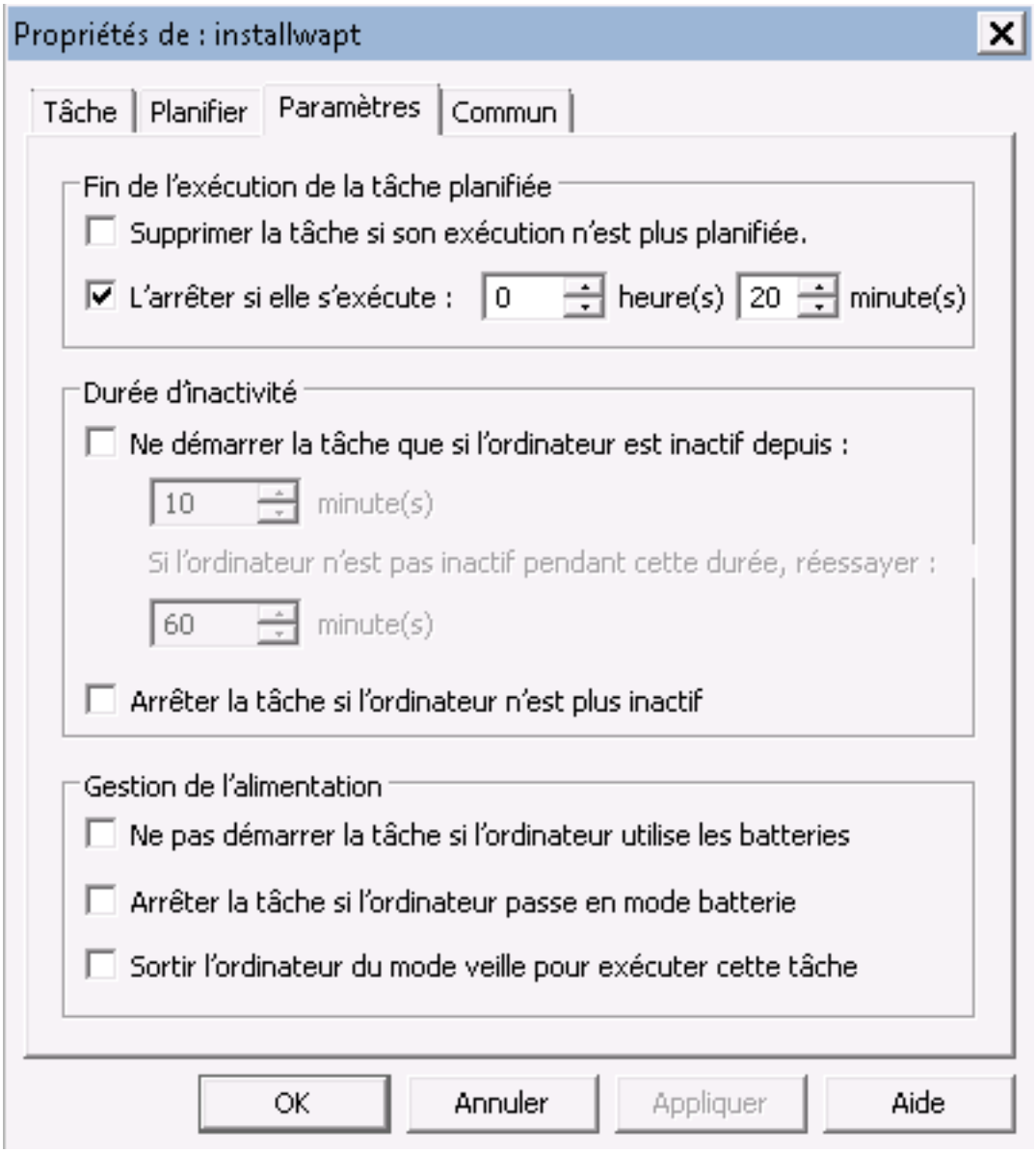


Figure70: Configuration de la gestion de l'alimentation

6.6.6 Déployer l'agent WAPT sur Linux

Nouveau dans la version 1.8.

À partir de WAPT 1.8, un agent Linux est disponible pour  /  and  .

Note:

- la procédure suivante installe l'agent WAPT en utilisant les dépôts Tranquil IT pour Debian / CentOS ;
- si vous souhaitez l'installer manuellement, vous pouvez rechercher votre version correspondante ;
- copier le lien du binaire dont vous avez besoin, le télécharger et l'installer avec dpkg / rpm ;

Installer l'agent WAPT sur Debian

La façon la plus sûre et la plus fiable d'installer le dernier agent WAPT sur Linux Debian est d'utiliser le dépôt public Tranquil IT.

- ajouter le dépôt Tranquil IT dans les listes apt :

Important: Suivez cette procédure pour obtenir les bons paquets pour l'édition WAPT Enterprise. Pour l'édition Community de WAPT, veuillez vous référer au bloc suivant.

Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **deb** pour accéder au dépôt WAPT Enterprise.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo "deb https://user:password@srvwapt-pro.tranquil.it/entreprise/debian/wapt-1.8/
↪$(lsb_release -c -s) main" > /etc/apt/sources.list.d/wapt.list
```

Important: Suivez cette procédure pour obtenir les bons paquets pour l'édition Community de WAPT. Pour l'édition WAPT Enterprise, veuillez vous référer au bloc précédent.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo "deb https://wapt.tranquil.it/debian/wapt-1.8/ $(lsb_release -c -s) main" > /etc/apt/
↪sources.list.d/wapt.list
```

- installer l'agent WAPT en utilisant apt :

```
apt update
apt install tis-waptagent
```

Installer l'agent WAPT sur CentOS

La façon la plus sûre et la plus fiable d'installer le dernier agent WAPT sur Linux CentOS est d'utiliser le dépôt public Tranquil IT.

- ajouter le dépôt Tranquil IT dans les listes yum :

Important: Suivez cette procédure pour obtenir les bons paquets pour l'édition WAPT Enterprise. Pour l'édition Community de WAPT, veuillez vous référer au bloc suivant.

Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **baseurl** pour accéder au dépôt WAPT Enterprise.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://user:password@srvwapt-pro.tranquil.it/entreprise/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF
```

Important: Suivez cette procédure pour obtenir les bons paquets pour l'édition Community de WAPT. Pour l'édition WAPT Enterprise, veuillez vous référer au bloc précédent.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF
```

- installer l'agent WAPT en utilisant yum :

```
yum install tis-waptagent
```

Créer le fichier de configuration de l'agent

Les conditions requises pour que votre agent WAPT puisse travailler sont les suivantes :

- le fichier de configuration `wapt-get.ini` se trouve dans le fichier `/opt/wapt/` ;
- un certificat public de l'autorité de signature des paquets dans `/opt/wapt/ssl/` ;

Vous devez créer et configurer le fichier `wapt-get.ini` dans `/opt/wapt` (*Configurer l'agent WAPT*).

Un exemple de ce à quoi il devrait ressembler est présenté plus bas sur cette page. Vous pouvez l'utiliser après avoir modifié les paramètres en fonction de vos besoins.

```
vim /opt/wapt/wapt-get.ini
```

```
[global]
repo_url=https://srvwapt.mydomain.lan/wapt
wapt_server=https://srvwapt.mydomain.lan/
use_hostpackages=1
use_kerberos=0
verify_cert=0
```

Copier le certificat Code-Signing

Vous devez copier manuellement, ou par script, le certificat public de votre autorité de certification de signature de paquets.

Le certificat devrait être situé sur votre machine Windows dans `C:\Program Files (x86)\wapt\ssl\`.

Copier les certificat(s) dans `/opt/wapt/ssl` en utilisant **`WinSCP`** ou **`rsync`**.

Copier le certificat SSL/TLS

Si vous avez déjà configuré votre serveur WAPT pour utiliser les bons certificats *Nginx SSL/TLS*, vous devez copier le certificat dans votre agent WAPT Linux.

Le certificat devrait être situé sur votre machine Windows dans `C:\Program Files (x86)\wapt\ssl\server\`.

Copier le(s) certificat(s) dans `/opt/wapt/ssl/server/` en utilisant **`WinSCP`** ou **`rsync`**.

Ensuite, modifier le ficher de configuration en donnant le chemin de votre certificat.

```
vim /opt/wapt/wapt-get.ini
```

Et donner le chemin absolu de votre certificat.

```
verify_cert=/opt/wapt/ssl/server/YOURCERT.crt
```

Attention: Si vous n'utilisez pas de certificat SSL/TLS avec votre serveur WAPT, vous devez changer dans `/opt/wapt/wapt-get.ini` les lignes suivantes à *0* :

```
verify_cert=0
```

Enregistrer votre agent Linux avec le serveur WAPT

Attention:

- attention, par défaut, WAPT prend la langue du système par défaut pour les paquets, vous devrez peut-être définir la langue dans `wapt-get.ini` avec `locales=`.

- redémarrer le serveur WAPT :

```
systemctl restart waptservice.service
```

- enfin, exécuter la commande suivante pour enregistrer votre hôte Linux auprès du serveur WAPT :

```
wapt-get register
wapt-get update
```

Félicitations,votre agent Linux est maintenant installé et configuré et il apparaîtra dans votre console WAPT avec une icône

!!

Fonctionnalités supportées

La plupart des fonctionnalités sont désormais prises en charge dans la version 1.8.2 de WAPT.

Fonctionnalités non supportées

- installer les mises à jour à l'extinction de la machine 

;
- La console WAPT n'est actuellement pas disponible sur Linux 

;
- toute fonctionnalité spécifique à Windows ;

Particularités du fonctionnement en domaine

- les tests ont été effectués avec sssd avec un domaine Active Directory et une authentification kerberos ;
- pour intégrer une machine dans le domaine Active Directory, vous pouvez choisir de suivre cette documentation
- pour forcer la mise à jour des Unités Organisationnelles sur la machine, vous pouvez appliquer une commande **`gpupdate`** depuis la console WAPT ;
- pour que les groupes Active Directory fonctionnent correctement, vous devez vérifier que la commande **`id hostname$`** renvoie la liste des groupes dont la machine est membre ;

Attention: Nous avons remarqué que la requête LDAP Kerberos ne fonctionne pas si l'enregistrement DNS inverse n'est pas configuré correctement pour vos contrôleurs de domaine. Ces enregistrements doivent donc être créés s'ils n'existent pas.

6.6.7 Déployer l'agent WAPT sur MacOS

Nouveau dans la version 1.8.

Attention: Actuellement, l'agent WAPT pour MacOS n'a été testé que sur High Sierra (version 10.13) et Mojave (10.14) alors que la dernière version de MacOS est Catalina (10.15). Catalina a peut-être introduit des changements qui pourraient empêcher l'agent WAPT de fonctionner.

Installer le paquet agent WAPT pour MacOS à partir du dépôt public de Tranquil IT

- download WAPT agent for Apple Mac OSX : Copy link from Tranquil IT's public repository and paste it into a terminal

```
sudo curl <PastedLink> tis-waptagent.pkg
```

- installer le paquet téléchargé :

```
sudo installer -pkg tis-waptagent.pkg -target /
```

Créer le fichier de configuration de l'agent WAPT pour MacOS

Les conditions requises pour que votre agent WAPT puisse fonctionner sont les suivantes :

- Le fichier de configuration `wapt-get.ini` se trouve dans le fichier `/opt/wapt/` ;
- un certificat public de l'autorité de signature des paquets dans `/opt/wapt/ssl/` ;

Vous devez créer et configurer le fichier `wapt-get.ini` dans `/opt/wapt` (*Configurer l'agent WAPT*).

Un exemple de ce à quoi il devrait ressembler est présenté plus bas sur cette page. Vous pouvez l'utiliser après avoir modifié les paramètres en fonction de vos besoins.

```
sudo vim /opt/wapt/wapt-get.ini
```

```
[global]
repo_url=https://srvwapt.mydomain.lan/wapt
wapt_server=https://srvwapt.mydomain.lan/
use_hostpackages=1
use_kerberos=0
verify_cert=0
```

Copier le certificat Code-Signing

Vous devez copier manuellement, ou par script, le certificat public de votre autorité de certification de signature de paquets.

Le certificat devrait être situé sur votre machine Windows dans `C:\Program Files (x86)\wapt\ssl\`.

Copier les certificat(s) dans `/opt/wapt/ssl` en utilisant **`WinSCP`** ou **`rsync`**.

Copier le certificat SSL/TLS

If you already have configured your WAPT server to use correct *Nginx SSL/TLS certificates*, you must copy the certificate in your WAPT Mac agent.

Le certificat devrait être situé sur votre machine Windows dans `C:\Program Files (x86)\wapt\ssl\server\`.

Copier le(s) certificat(s) dans `/opt/wapt/ssl/server/` en utilisant **`WinSCP`** ou **`rsync`**.

Ensuite, modifier le ficher de configuration `wapt-get.ini` en fournissant le chemin de votre certificat.

```
sudo vim /opt/wapt/wapt-get.ini
```

Et donner le chemin absolu de votre certificat.

```
verify_cert=/opt/wapt/ssl/server/YOURCERT.crt
```

Attention: Si vous n'utilisez pas de certificat SSL/TLS avec votre serveur WAPT, vous devez changer dans `/opt/wapt/wapt-get.ini` les lignes suivantes à *0* :

```
verify_cert=0
```

Enregistrer votre agent MacOS avec le serveur WAPT

Attention:

- attention, par défaut, WAPT prend la langue du système par défaut pour les paquets, vous devrez peut-être définir la langue dans `wapt-get.ini` avec `locales=`.

- redémarrer le serveur WAPT :

```
sudo launchctl unload /Library/LaunchDaemons/com.tranquilit.tis-waptagent.plist
sudo launchctl load /Library/LaunchDaemons/com.tranquilit.tis-waptagent.plist
```

- enfin, exécuter la commande suivante pour enregistrer votre hôte MacOS auprès du serveur WAPT :
- you must logon as root to run :

```
wapt-get register
```

- then switch back to normal user for the following :

```
sudo wapt-get update
```

Félicitations, votre agent MacOS est maintenant installé et configuré et il apparaîtra dans votre console WAPT avec une icône  !!

Fonctionnalités supportées

La plupart des fonctionnalités sont désormais prises en charge dans la version 1.8.2 de WAPT.

Fonctionnalités non supportées

- installer les mises à jour à l'extinction de la machine 

;
- La console WAPT n'est actuellement pas disponible sur Linux 

;
- toute fonctionnalité spécifique à Windows ;

Particularités du fonctionnement en domaine

- les tests ont été effectués avec sssd avec un domaine Active Directory et une authentification kerberos ;
- pour intégrer une machine dans le domaine Active Directory, vous pouvez choisir de suivre cette documentation
- pour forcer la mise à jour des Unités Organisationnelles sur la machine, vous pouvez appliquer une commande **`gpupdate`** depuis la console WAPT ;
- pour que les groupes Active Directory fonctionnent correctement, vous devez vérifier que la commande **`id hostname$`** renvoie la liste des groupes dont la machine est membre ;

Attention: Nous avons remarqué que la requête LDAP Kerberos ne fonctionne pas si l'enregistrement DNS inverse n'est pas configuré correctement pour vos contrôleurs de domaine. Ces enregistrements doivent donc être créés s'ils n'existent pas.

6.6.8 Déployer l'agent WAPT Linux avec Ansible

Pour éviter les erreurs et automatiser le déploiement de vos agents WAPT sous Linux, nous vous proposons des rôles Ansible pour installer des agents WAPT sur :

- 
- 
- 

Vous pouvez explorer le code source du rôle en visitant ce lien sur Github.

Pré-requis

- Postes Debian Linux ou CentOS;
- un compte utilisateur sudoers sur ces postes;
- Ansible 2.8;

Installer le rôle Ansible

- installer le rôle Ansible `tranquilit.waptagent` ;

```
ansible-galaxy install tranquilit.waptagent
```

- pour installer le rôle autre part, utiliser la commande *-p* comme ceci;

```
ansible-galaxy install tranquilit.waptagent -p /path/to/role/directory/
```

Utiliser le rôle Ansible

- assurez vous d'avoir une clé ssh déployée sur vos postes, sinon vous pouvez en générer une et la copier comme ci dessous;

```
ssh-keygen -t ed25519
ssh-copy-id -i id_ed25519.pub user@computer1.mydomain.lan
ssh user@computer1.mydomain.lan -i id_ed25519.pub
```

- éditer le fichier d'inventaire Ansible (`./hosts`) et ajouter les hôtes Linux ;

```
[computers]
computer1.mydomain.lan ansible_host=192.168.1.50
computer1.mydomain.lan ansible_host=192.168.1.60
```

- créer un playbook avec le contenu suivant dans `./playbooks/deploywaptagent.yml` ;

```
- hosts: computers
  roles:
    - { role: tranquilit.waptagent }
```

- assurez-vous que les variables sont correctement renseignées (voir *Variables wapt-get.ini*) ;
  - `wapt_server_url`;
  - `wapt_repo_url`;
  - `wapt_crt`;

Important: La configuration des variables est importante car elles configurent le comportement de l'agent WAPT.

Vous **devez** remplacer le certificat par défaut avec votre certificat public de signature de vos paquets WAPT.

- lancez votre playbook avec la commande suivante ;

```
ansible-playbook -i ./hosts ./playbooks/deploywaptagent.yml -u user --become --become-
↪method=sudo -K
```

Félicitations, vous avez installé votre agent WAPT sur vos hôtes Linux!

Variables du rôle

Les variables disponibles sont listées ci-dessous, avec les valeurs par défaut (voir `defaults/main.yml`).

Variables des agents WAPT

- version de WAPT qui sera installée depuis les dépôts WAPT Deb/RPM ;

```
wapt_version: "1.8"
```

- version CentOS utilisée pour l'adresse du dépôt RPM ;

```
centos_version: "centos7"
```

Variables wapt-get.ini

Le paramètre `wapt_server_url` pointe vers votre serveur WAPT et est utilisé par défaut pour `wapt_repo_url`.

```
wapt_server_url: "https://srvwapt.mydomain.lan"
wapt_repo_url: "{{ wapt_server_url }}/wapt/"
```

Vous pouvez le surcharger comme ceci :

```
wapt_server_url: "https://wapt.landomain.lan"
wapt_repo_url: "https://wapt.otherdomain.com/wapt/"
```

Nom du certificat situé dans le sous-dossier `files/` du rôle :

```
wapt_crt: "wapt_ca.crt"
```

Exemple de playbook

Voici un exemple de playbook Ansible.

```
- hosts: hosts
  vars_files:
    - vars/main.yml
  roles:
    - tranquilit.waptagent
```

Pour aller plus loin, les différentes possibilités de configuration sont détaillées dans :

6.6.9 Configurer l'agent WAPT

Le fichier de configuration `C:\Program Files (x86)\wapt\wapt-get.ini` contrôle le comportement de l'agent WAPT.

La section `[global]` est obligatoire :

```
[global]
```

Description des options possibles pour l'agent WAPT

Note:

- si les champs `repo_url` et `wapt_server` sont vides, le client WAPT ira chercher son dépôt en utilisant l'enregistrement SRV dans la zone `dnsdomain` ;
- s'il n'y a pas de clé `wapt_server` dans la section `[global]`, aucun serveur WAPT ne sera contacté ;
- s'il n'y a pas d'attribut `repo_url` dans la section `[global]`, il sera nécessaire de définir explicitement un dépôt dans une section `[wapt]`, et de l'activer en l'ajoutant à la clé `repositories` de la section `[global]` ;
- les dépôts activés sont listés dans le paramètres `repositories` de la section `[global]` ;

Table4: Description des options possibles pour l'agent WAPT

Options	Description
use_hostpackages = 1	Utiliser du paquet machine ou non (défaut 1).
waptupdate_task_period = 120	Fréquence d'update (120 minutes par défaut).
waptupgrade_task_period = 360	Fréquence d'upgrade (inactif par défaut).
waptservice_port = 8088	Port local HTTPS d'écoute du service WAPT.
dbpath = C:\Program Files(x86)\wapt\db\waptdb.sqlite	Path to the local database file.
loglevel = warning	Niveau de log de l'agent WAPT. Valeurs possibles : debug, info, warning, critical.
maturities = PROD	Liste des maturités de paquets qui peuvent être vus et installés par l'agent WAPT. La valeur par défaut est PROD. N'importe quelle valeur peut être utilisée.
use_fqdn_as_uuid = 1	Permet d'utiliser le nom FQDN plutôt que l'UUID BIOS comme identifiant unique de la machine dans WAPT.
waptupdate_task_period = 120	Définit la fréquence à laquelle l'agent vérifie s'il a des audits à faire.
loglevel = warning	Permet de définir la liste des langues de l'agent WAPT pour modifier la liste des paquets visibles par WAPT (pour le filtrage des paquets). Vous pouvez ajouter plusieurs langues (par exemple locales=fr,en) par ordre de préférence.
host_profiles = tis-firefox,tis-java	Définit une liste de paquets WAPT que l'agent WAPT doit installer.
language = en	Forcer la langue par défaut de la console graphique (ne s'applique pas au filtrage des paquets)
host_organizational_unit_dn = OU=TOTO,OU=TEST,DC=DEMO,DC=LAN	Allows you to force an Organizational Unit on the WAPT agent. (Convenient to assign a fake OU for out-of-domain PC) Make sure it respects a consistent case (don't mix « dc »s and « DC »s, for example), which you can find in the console (in the DN/computer_ad_dn fields for each machine)
download_after_update_with_waptupdate_task_period = True	Définit si un téléchargement de paquets en attente doit être lancé après une mise à jour avec waptupdate_task_period
log_to_windows_events = False	Envoyer les logs wapt dans la fenêtre événements
service_auth_type = system	How the self service authentication works. Possible values are: system, waptserver-ldap or waptagent-ldap
uninstall_allowed = 1	Whether or not it is possible for the user to uninstall applications via the self-service.

Paramètres d'accès au serveur WAPT

Ces options définissent le comportement de l'agent WAPT lors de sa connexion avec le serveur WAPT.

Table5: Description des options possibles pour le serveur WAPT

Options	Description
`wapt_server` =	URL du serveur WAPT. Si pas présent, aucun serveur n'est contacté. Si vide, une recherche DNS est effectuée sur la zone `dnsdomain`.
`dnsdomain` =	Zone sur laquelle la recherche DNS SRV `_waptserver._tcp` est effectuée.
`wapt_server_timeout` = 10	Timeout en secondes lors de la tentative de connexion https vers le serveur
`use_kerberos` = 1	Utilisation de l'authentification Kerberos pour l'enregistrement sécurisé avec le serveur WAPT.
`verify_cert` = C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt	Voir la documentation pour activer la *vérification des certificats HTTPS*
`public_certs_dir` = `C:\Program Files (x86)\wapt\ssl`	Répertoire des certificats autorisés pour la vérification de _signature_ des paquets. Par défaut : `<wapt_base_dir>\\ssl`. Seuls les fichiers dans ce répertoire avec l'extension `.crt` ou `.pem` sont pris en compte. Il peut y avoir plusieurs certificats X509 dans chaque fichier. Les certificats de paquets autorisés sont ceux dont la signature peut être vérifiée par l'un des certificats contenus dans les fichiers PEM de ce répertoire. Chaque dépôt peut avoir son propre répertoire de certificats autorisés.

Utiliser plusieurs dépôts

D'autres sections peuvent être définies dans le `wapt-get.ini` pour définir d'autres dépôts.

- `[wapt]`: dépôt principal. Options pertinentes: `repo_url`, `verify_cert`, `dnsdomain`, `http_proxy`, `use_http_proxy_for_repo`, `timeout`. Si cette section n'existe pas, les paramètres sont récupérés depuis la section `[global]` ;
- `[wapt-templates]` : dépôt externe par défaut dans la console WAPT pour importer des nouveaux paquets ou des paquets mis à jour ;
- `[wapt-host]`: dépôt pour les paquets machines. Si cette section n'existe pas, elle est extrapolée des paramètres du dépôt principal ;

Plus d'informations sur cet usage sont à découvrir dans cet *article sur comment fonctionner avec de multiples dépôts publics ou privés*.

Note: Les dépôts activés sont listés dans le paramètres `repositories` de la section `[global]`.

Table6: Description des options possibles pour les dépôts

Options	Description
`repositories` = depot1,depot2	Liste des dépôts activés. Séparateur : virgule. Les valeurs désignent le nom de la section du fichier `wapt-get.ini`. Dans chaque section, on peut préciser `repo_url`, `dnsdomain`, `public_certs_dir`, `http_proxy`.

Note: Ce paramètre est également à définir dans la configuration de la console WAPT située dans `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

Pour plus d'informations, visitez *la documentation sur la configuration de la console WAPT*.

Paramètres pour waptexit

Table7: Description des options possibles pour WAPTexit

Options	Description
`allow_cancel_upgrade` = 1	Interdire à l'utilisateur d'annuler la mise à jour lors de l'extinction de sa machine.
`pre_shutdown_timeout` = 180	Temps d'attente maximum pour l'exécution des scripts à l'extinction.
`max_gpo_script_wait` = 180	Temps d'attente maximum pour l'application des GPO à l'extinction.
`hiberboot_enabled` = 0	Désactive Hiberboot pour utiliser **`waptexit`** sur Windows 10.

Paramètres pour le libre-service WAPT et l'authentification Waptservice

Table8: Description des options disponibles pour le libre-service WAPT et l'authentification Waptservice

Options	Description
`waptservice_admin_filter` = True	Appliquer le filtrage des paquets *selfservice* aux Administrateurs Locaux.
`service_auth_type` = system	Définit le système d'authentification du service wapt, les valeurs disponibles sont *system*, *waptserver-ldap*, *waptagent-ldap*.
`ldap_auth_ssl_enabled` = False	Utile avec *waptagent-ldap*, définit si la demande LDAP doit être chiffrée.
`verify_cert_ldap` = True	Utile avec *waptagent-ldap*, définit si le certificat doit être vérifié.
`ldap_auth_base_dn` = dc=domain,dc=lan	Utile avec *waptagent-ldap*, définit le dn de base pour la requête LDAP.
`ldap_auth_server` = srvads.domain.lan	Utile avec *waptagent-ldap*, définit le serveur LDAP à contacter.
`waptservice_user` = admin	Oblige un utilisateur à s'authentifier sur le service WAPT.
`waptservice_password` = 5e884898da	Mot de passe hashé sha256 hex lors de l'appel d'une fonction par un *waptservice_user* (la valeur *NOPASSWORD* permet de ne pas mettre de mot de passe).

Paramètres pour wapttray

Table9: Description des options possibles pour le WAPTtray

Options	Description
`notify_user` = 0	Empêche `waptray` d'envoyer des notifications (popup).

Paramètres proxy

Table10: Description des options possibles pour le serveur WAPT

Options	Description
`http_proxy` = http://user:pwd@host_fqdn:port	Adresse du proxy HTTP à utiliser
`use_http_proxy_for_repo` = 0	Utiliser un proxy pour accéder aux dépôts.
`use_http_proxy_for_server` = 0	Utiliser un proxy pour accéder au serveur WAPT.
`use_http_proxy_for_templates` = 0	Utiliser un proxy pour accéder au serveur de modèles de paquets.

Paramètres pour la création de paquets

Table11: Description des options possibles pour créer des paquets WAPT

Options	Description
`personal_certificate_path` = C:\private\org-codeur.crt	Chemin de la clé privée de l'Administrateur.
`default_sources_root` = C:\waptdev	Répertoire de développement des paquets.
`default_sources_root_host` = C:\waptdev\hosts	Répertoire de développement des paquets hosts.
`default_package_prefix` = tis	Préfixe des paquets par défaut ou des paquets importés.
`default_sources_suffix` = wapt	Préfixe des paquets par défaut ou des paquets importés.

Paramètres pour les `WAPT Windows Updates`

Se référer à *cet article sur la configuration de WAPTWUA pour l'agent WAPT*.

Paramètres de surcharge des commandes *upload*

Il est possible de surcharger les commandes **upload** pour définir un comportement particulier lors de l'envoi des paquets. On pourrait par exemple envoyer le paquet sur plusieurs dépôts, via un autre protocole, etc.

Pour téléverser le paquet hôte sur le serveur (**upload-package** ou **build-upload** d'un paquet machine), utiliser :

```
upload_cmd="C:\\Program Files (x86)\\WinSCP\\WinSCP.exe" admin@srvwapt.mydomain.lan /upload
↪%(waptfile)s
```

Pour téléverser le paquet hôte sur le serveur (**upload-package** ou **build-upload** d'un paquet machine), utiliser :

```
upload_cmd_host="C:\\Program Files (x86)"\\putty\\pscp -v -l admin %(waptfile)s srvwapt.mydomain.
↪lan:/var/www/wapt-host/
```

Pour à exécuter après l'**upload** :

```
after_upload="C:\\Program Files (x86)"\\putty\\plink -v -l admin srvwapt.mydomain.lan "python /
↪var/www/wapt/wapt-scanpackages.py /var/www/%(waptdir)s/"
```

Configurer les agents WAPT

Lors de l'installation les paramètres par défaut sont :

```
[global]
waptupdate_task_period=120
waptserver=https://srvwapt.mydomain.lan
repo_url=https://srvwapt.mydomain.lan/wapt/
use_hostpackages=1
```

Modifier le `wapt-get.ini` et regénérer un agent ne suffit pas à pousser une nouvelle configuration.

Vous pouvez en revanche faire un paquet WAPT qui va pousser la nouvelle configuration.

Le paquet exemple est présent dans le dépôt Tranquil IT : https://store.wapt.fr/wapt/tis-wapt-conf-policy_6_f913e7abc2f223c3e243cc7b7f95caa5.wapt :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():

  print('Modify max_gpo_script_wait')
  inifile_writestring(WAPT.config_filename,'global','max_gpo_script_wait',180)

  print('Modify Preshutdowntimeout')
  inifile_writestring(WAPT.config_filename,'global','pre_shutdown_timeout',180)

  print('Disable Hyberboot')
  inifile_writestring(WAPT.config_filename,'global','hiberboot_enabled',0)

  print('Disable Notify User')
  inifile_writestring(WAPT.config_filename,'global','notify_user',0)
```

6.6.10 Configurer la console WAPT

Indication: La configuration de la console WAPT est stockée dans le fichier `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`. Ce fichier est généré automatiquement lors du premier lancement de **waptconsole** à partir du fichier `wapt-get.ini` de l'agent de l'*Administrateur*.

On peut retrouver plusieurs options dans la section `[global]` du fichier `waptconsole.ini` :

Table12: Description des options possibles pour la console WAPT

Options	Description
`wapt_server = https://srvwapt.mydomain.lan`	Adresse du serveur WAPT.
`repo_url = https://srvwapt.mydomain.lan/wapt`	Adresse du dépôt WAPT principal.
`last_usage_report = 03/01/2017 18:45:51`	Date du dernier usage de la console.
`http_proxy =`	Adresse du proxy dans la console.
`use_http_proxy_for_server = 0`	Utiliser un proxy pour contacter le serveur WAPT depuis la console.
`use_http_proxy_for_repo = 0`	Utiliser un proxy pour contacter le dépôt WAPT depuis la console.
`default_package_prefix = tis`	Préfix pour personnaliser le nommage des paquets.
`default_sources_root = c:\\waptdev`	Répertoire de développement des paquets WAPT.
`personal_certificate_path =` C:\private\mykey.crt	Chemin du certificat associé à la clé privée de l'*Administrateur*.
`send_usage_report = 1`	Autoriser la console à envoyer des statistiques anonymes à Tranquil IT.
`language = en`	Langue de la console.
`advanced_mode = 0`	Lancer la console en debug.
C:\Program Files (x86)\wapt\	Pour *les options possibles, voir* pour vérifier les certificats HTTPS.
`waptservice_timeout = 2`	Timeout lors d'actions sur les agent WAPT. Ex = **`update`**.
`enable_external_tools = 0`	Afficher les actions faisant appel à des applications externes (RDP, VNC etc...).
`enable_management_features = 0`	Afficher le bouton de création de certificat et d'agent.
`hide_unavailable = 0`	Cacher les actions indisponibles pour l'agent.
`check_certificates_validity = 1`	Vérifier la date et la CRL du certificat du paquet.
`sign_digests = sha256,sha1`	Liste les algorithmes de signature autorisés pour les paquets.

Configurer des dépôts externes

Vous pouvez ajouter plusieurs dépôts externes en ajoutant des `[sections]` dans le fichier `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

Exemple :

```
[store.wapt.fr]
repo_url=https://store.wapt.fr/waptdev
verify_cert=1
http_proxy=http://proxy.mydomain.lan:8080
public_certs_dir=
timeout=2

[otherwapt.tranquil.it]
repo_url=https://otherwapt.tranquil.it/waptdev
verify_cert=0
http_proxy=
public_certs_dir=c:\Users\admin\Documents\ssl\otherwapt\
timeout=2
```

Table13: Description des options possibles pour les dépôts extérieurs

Options	Description
repo_url = https://srvwapt.mydomain.lan/wapt	Adresse du dépôt WAPT externe.
wapt_server = https://srvwapt.mydomain.lan	Adresse du proxy à utiliser pour accéder au dépôt externe référencé dans la « [section] ».
verify_cert = 1	Pour *les options possibles, voir* pour vérifier les certificats HTTPS.
public_certs_dir =	Dossier qui contient les certificats utilisés pour authentifier les paquets externes téléchargés.
timeout = 2	Délai d'attente pour le dépôt externe référencé dans la [section]. S'il est laissé vide, aucune vérification n'est effectuée.

Paramètres pour la création de paquets WAPT

Table14: Description des options possibles pour créer des paquets WAPT

Options	Description
personal_certificate_path = C:\private\coder.crt	Chemin de la clé privée pour signer les paquets.
default_sources_root = c:\\waptdev	Répertoire de développement des paquets WAPT.
default_sources_root_host = C:\waptdev\hosts	Répertoire de développement des paquets machine.
default_package_prefix = tis	Préfixe des paquets par défaut.
default_sources_suffix = wapt	Suffixe des paquets par défaut.

6.6.11 Configurer le serveur WAPT

La configuration du serveur WAPT sous Linux est située dans /opt/wapt/waptserver/waptserver.ini.

Le fichier de configuration du serveur WAPT sur les systèmes Windows se trouve dans C:\wapt\conf\waptserver.ini.

Attention: La modification de ces fichiers est à réserver aux utilisateurs avertis !!

Partie [option]

On peut lui passer plusieurs options dans la section :

```
[options]
```

Table15: Paramètres disponibles pour la section [option] de waptserver.ini

Options	Description
allow_unauthenticated_connect = False	Définit si les requêtes websocket doivent être authentifiées
allow_unauthenticated_registration = True	Permet une authentification Identifiant / Mot de passe pour l'enregistrement initial
allow_unsigned_status_data = False	Debug - Autorise la réception de données d'agent non signées

suite sur la page suivante

Table 15 – suite de la page précédente

Options	Description
application_root = ""	Défini le chemin d'application de WAPT Server (ex: wapt)
auto_create_ldap_users = True	Relatif aux ACLs utilisateurs
client_certificate_lifetime = 3650	Durée maximale du certificat client
clients_read_timeout = 5	Durée de délai d'expiration Websocket
clients_signing_certificate =	Certificat de signature des certificats machines
clients_signing_crl_days =	Expiration de la CRL des certificats machines
clients_signing_crl =	CRL des certificats machines
clients_signing_crl_url =	URL de la CRL des certificats machines
clients_signing_key =	Clé de signature des certificats machines
client_tasks_timeout = 1	Délai d'attente maximum pour la réponse de l'agent WAPT
db_connect_timeout = 10	Délai d'attente maximum pour les requête de PostgreSQL
db_host =	Adresse du serveur PostgreSQL (par défaut vide car la connexion par socket UNIX est par défaut)
db_max_connections = 100	Nombre de connexions simultanés pour les requêtes de PostgreSQL
db_name = wapt	Nom de la base PostgreSQL à laquelle le serveur WAPT va se connecter.
db_password =	Mot de passe pour la base PostgreSQL (par défaut vide car la connexion par socket UNIX est activée par défaut)
db_port = 5432	Port du serveur PostgreSQL
db_stale_timeout = 300	Délai d'expiration de la connexion à la base de donnée
db_user =	Nom d'utilisateur pour la base PostgreSQL (par défaut vide car la connexion par socket UNIX est activée par défaut)
enable_store = False	Enables WAPT Store Webui (WAPT Enterprise only)
encrypt_host_packages = False	Chiffre le paquet machine avec le certificat machine client
htpasswd_path = None	Ajoute l'autentifcation basic à WAPT Server
http_proxy = http://srvproxy.mydomain.lan:3128	Définit le serveur proxy du serveur WAPT (pour récupérer des CRL (Certificate Revocation List))
known_certificates_folder = /opt/wapt/ssl/	Ajoute un répertoire additionnel des CA pour la vérification des certificats
ldap_auth_base_dn = None	Défini le DN de base LDAP
ldap_auth_server = None	Défini le serveur d'authentification LDAP
ldap_auth_ssl_enabled = True	Active la connexion SSL pour LDAP
loglevel = debug	Niveau de débuggage. Le niveau par défaut est warning
max_clients = 4096	Défini le nombre de connexion simultanées maximum de clients WAPT
min_password_length = 10	Définit la taille minimale de mot de passe
nginx_http = 80	Défini le port Nginx http (Windows seulement)
nginx_https = 443	Défini le port Nginx https (Windows seulement)
remote_repo_diff = False	Active la diff pour la réplication dépôt distant
remote_repo_support = True	Active la fonctionnalité de réplication de dépôts sur WAPT Server
remote_repo_websockets = True	Permet la communication par websocket pour la réplication dépôt distant
secret_key = FKjfzjfkF687fjrkeznfkj7678jknk78687	Chaîne d'initialisation du moteur de gestion de session Flask. Il est généré lors de la première installation du serveur WAPT et est unique pour chaque serveur WAPT.
server_uuid = 76efezfa6-b309-1fez5-92cd-8ea48fc122dc	*UUID* du server (pour les statistiques anonymes WAPT)

suite sur la page suivante

Table 15 – suite de la page précédente

Options	Description
signature_clockskew = 72000	Décalage de temps autorisé pour les timestamps lors d'une connexion websocket
token_lifetime = 43200	Durée d'un jeton d'authentification
trusted_signers_certificates_folder = None	Chemin du dossier des certificats de signature de confiance
trusted_users_certificates_folder = None	Chemin du dossier des CA utilisateur de confiance
use_kerberos = True	Exige une authentification Kerberos pour l'enregistrement initial d'une machine sur le serveur WAPT.
use_ssl_client_auth = False	Active l'authentification par certiticat client
wapt_admin_group_dn = CN=waptadmins,OU=groups,DC=ad,DC=mydomain,DC=lan	LDAP DN du groupe d'utilisateurs Active Directory autorisé à se connecter à la console WAPT
wapt_admin_group = None	LDAP CN du groupe d'utilisateurs Active Directory autorisé à se connecter à la console WAPT
wapt_folder = /var/www/wapt	Répertoire du dépôt WAPT.
wapt_huey_db = C:\Program Files(x86)\wapt\db\waptservertasks.sqlite	Path to database that handles tasks
wapt_password = 46642dd2b1dfezfezgfezgadf0ezgeezgezf53d	Mot de passe du *SuperAdmin* pour la console WAPT.
waptserver_port = 8080	Défini le port de WAPT Server python, par defaut 8080
wapt_user = admin	Identifiant de connexion du *SuperAdmin* pour la console WAPT.
waptwua_folder = /var/www/waptwua	Emplacement du dossier WAPT WUA
wol_port = 9,123,4000	Liste des ports UDP WakeOnLAN auxquels envoyer des paquets magiques
wapt_bind_interface = 127.0.0.1	Définit comment écouter au service waptserver

Configurer Nginx

La configuration nginx par défaut est la suivante :

```
server {
  listen                      80;
  listen                      443 ssl;
  server_name                 _;
  ssl_certificate             "/opt/wapt/waptserver/ssl/cert.pem";
  ssl_certificate_key         "/opt/wapt/waptserver/ssl/key.pem";
  ssl_protocols               TLSv1.2;
  ssl_dhparam                 /etc/ssl/certs/dhparam.pem;
  ssl_prefer_server_ciphers   on;
  ssl_ciphers                 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
  ssl_stapling                on;
  ssl_stapling_verify         on;
  ssl_session_cache           none;
  ssl_session_tickets         off;
  index index.html;

  location ~ ^/wapt.* {
    proxy_set_header Cache-Control "store, no-cache, must-revalidate, post-check=0, pre-check=0";
    proxy_set_header Pragma "no-cache";
    proxy_set_header Expires "Sun, 19 Nov 1978 05:00:00 GMT";
    root "/var/www";
    }
```

(suite sur la page suivante)

(suite de la page précédente)

```
  location / {
    proxy_set_header X-Real-IP  $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;

  location  ~ ^/(api/v3/upload_packages|api/v3/upload_hosts/|upload_waptsetup)  {
    proxy_pass http://127.0.0.1:8080;
    client_max_body_size 4096m;
    client_body_timeout 1800;
    }

  location /wapt-host/Packages {
    return 403;
    }

  location /wapt-host/add_host_kerberos {
    return 403;
    }

  location / {
    proxy_pass http://127.0.0.1:8080;
    }

  location /socket.io {
    proxy_http_version 1.1;
    proxy_buffering off;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "Upgrade";
    proxy_pass http://127.0.0.1:8080/socket.io;
    }
  }
}
```

6.6.12 Configurer le serveur WAPT pour les grands déploiements

Les paramètre par défaut pour le système d'exploitation, le service web Nginx et la base de données Postgresql sont adaptés pour environ 400 agents WAPT. Si vous avez plus de 400 clients, il est nécessaire de modifier quelques paramètres au niveau du système, au niveau de la base de données PostgreSQL, du service web Nginx et du Serveur WAPT.

À l'avenir, le script **`postconf.sh`** pourrait prendre en charge cette configuration en fonction du nombre prévu d'ordinateurs clients.

Avec les paramètres suivants, un serveur WAPT devrait pouvoir accueillir jusqu'à environ 5 000 clients actifs simultanés. Vous pouvez avoir plus de clients dans la base de données s'ils ne fonctionnent pas tous en même temps. Si vous avez plus de 5000 clients, il est recommandé d'avoir plus d'un serveur WAPT en parallèle.

La limite du nombre de clients finaux est due au goulot d'étranglement dans le code python et le moteur PostgreSQL. Les performances du WAPT s'améliorent avec le temps et, à l'avenir, le serveur WAPT pourrait prendre en charge une large base de clients WAPT sur un seul serveur. Cependant, la partie Nginx est très bien dimensionnée et elle peut tirer pleinement partie d'une connexion 10 Gbps pour les gros déploiements de paquets.

Changements de configuration pour une performance renforcée

Note: Les paramètres à modifier ci-dessous sont liés entre eux et doivent être modifiés globalement et non individuellement.

Configurer Nginx

In the `/etc/nginx/nginx.conf` file (for Windows `C:\wapt\waptserver\nginx\conf\nginx.conf`), modify `worker_connections` parameter. The value should be around 2.5 times the number of WAPT clients (n connections for websockets and n connections for package downloads and inventory upload + some margin).

```
events {
    worker_connections 4096;
}
```

Then upgrade the number of *filedescriptors* in the `/etc/nginx/nginx.conf` file (for Windows `C:\wapt\waptserver\nginx\conf\nginx.conf`):

```
worker_rlimit_nofile 32768;
```

Configurer Linux

Augmenter le nombre de *filedescriptors*. Le fichier systemd demande une augmentation du nombre autorisé de *filedescriptors* (LimitNOFILE=32768). Nous devrions avoir la même chose pour Nginx. Il y a quelques limites à augmenter.

Tout d'abord, nous modifions au niveau du système le nombre de *filedescriptors* autorisés pour Nginx et WAPT.

- créer le fichier `/etc/security/limits.d/wapt.conf` :

```
cat > /etc/security/limits.d/wapt.conf <<EOF
wapt         hard    nofile      32768
wapt         soft    nofile      32768
www-data     hard    nofile      32768
www-data     soft    nofile      32768
EOF
```

Nginx sert de proxy inverse et établit un grand nombre de connexions. Chaque client WAPT maintient une connexion *websocket* en permanence afin d'échanger avec le serveur WAPT.

Le noyau Linux a une protection contre l'ouverture simultanée de trop de connexions TCP et on peut obtenir le message *SYN flooding on port* dans le journal d'historique Nginx. Afin d'éviter ces messages, il est nécessaire de modifier les deux paramètres suivants. Ils doivent être environ 1,5 fois supérieur au nombre de clients WAPT.

```
cat > /etc/sysctl.d/wapt.conf <<EOF
net.ipv4.tcp_max_syn_backlog=4096
net.core.somaxconn=4096
EOF

sysctl --system
```

Configurer PostgreSQL

A higher number of clients need a higher number of connections to the PostgreSQL database. In the `postgresql.conf` file (file:*/etc/postgresql/11/main/postgresql.conf* on debian 10 for example or for Windows `C:\wapt\waptserver\pgsql9.6_data\postgresql.conf`), you need to increase the following parameter to approximately 1/4 the number of active WAPT agents.

```
max_connections = 1000
```

Configurer le serveur WAPT

In `/opt/wapt/conf/waptserver.ini` file (for Windows `C:\wapt\conf\waptserver.ini`, `db_max_connections` should be equal to PostgreSQL `max_connections` minus 10 (PostgreSQL needs to keep some connections for its housekeeping stuff). The `max_clients` parameter should be set around 1.2 times the number of WAPT agents:

```
[options]
...
max_clients = 4096
db_max_connections =  990
```

Configuration pour le téléchargement de gros paquets

En fonction du partitionnement de votre serveur WAPT, vous devrez peut-être faire attention au répertoire de téléchargement de fichiers temporaires Nginx. Nginx agit comme un proxy inversé pour le moteur Python du serveur WAPT et il effectue une mise en cache des paquets téléchargés lors du téléchargement d'un nouveau paquet depuis la console.

Les paquets sont stockés dans le répertoire `/var/lib/nginx/proxy`. Vous devez vous assurer que la partition qui héberge ce répertoire est suffisamment grande. Vous pouvez changer l'emplacement de ce répertoire en utilisant le paramètre de configuration Nginx suivant.

```
$client_body_temp_path
```

Pour désinstaller proprement l'agent WAPT

6.6.13 Désinstaller l'agent WAPT sur les clients

Désinstaller l'agent WAPT sur Windows

Si vous devez désinstaller les agents WAPT des clients, le désinstalleur est automatiquement créé dans l'emplacement d'installation WAPT, par défaut il s'agit de `C:\Program Files (x86)\wapt\unins000.exe`.

- la désinstallation silencieuse par défaut de l'agent WAPT peut être réalisée avec la commande suivante :

```
unins000.exe /VERYSILENT
```

- un argument supplémentaire peut être passé à **`unins000.exe`** pour tout nettoyer :

```
unins000.exe /VERYSILENT /purge_wapt_dir=1
```

Liste complète des arguments de la ligne de commande pour **unins000.exe** :

Paramètres	Description
/VERYSILENT	Lance unins000.exe en mode silencieux
/purge_wapt_dir=1	Purge le répertoire WAPT (supprime tous les dossiers et fichiers)

Désinstaller l'agent WAPT sous Linux

- la désinstallation par défaut de l'agent WAPT peut être réalisée avec la commande suivante, en fonction de votre système d'exploitation Linux :

```
# Debian / Ubuntu
apt remove --purge tis-waptagent

# CentOS / Redhat
yum remove tis-waptagent
```

- une étape supplémentaire peut être effectuée à l'aide de ces commandes (WIP) :

```
rm -f /opt/wapt/

# Debian / Ubuntu
rm /etc/apt/sources.list.d/wapt.list

# CentOS / Redhat
rm /etc/yum/yum.repos.d/wapt.list
```

Désinstaller l'agent WAPT sur MacOS

- la désinstallation par défaut de l'agent WAPT peut être réalisée avec la commande suivante :

```
# List all files to delete
pkgutil --only-files --files com.tranquilit.tis-waptagent-enterprise > file_list

# Remove packages
sudo pkgutil --forget com.tranquilit.tis-waptagent-enterprise
```

Réactiver Windows Update avant de désinstaller l'agent WAPT

Dans le cas où vous avez utilisé WAPT pour gérer les mises à jour de Windows, vous voudrez peut-être réactiver le comportement par défaut de Windows Update avant de désinstaller l'agent WAPT.

Pour ce faire, voici un exemple de paquet à pousser avant de désinstaller l'agent WAPT :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('Disable WAPT WUA')
```

(suite sur la page suivante)

(suite de la page précédente)

```
    inifile_writestring(WAPT.config_filename,'waptwua','enabled','false')

    print('DisableWindowsUpdateAccess registry to 0')
    registry_set(HKEY_LOCAL_MACHINE,r'Software\Policies\Microsoft\Windows\WindowsUpdate',
↪'DisableWindowsUpdateAccess',0,REG_DWORD)

    print('AUOptions registry to 0')
    registry_set(HKEY_LOCAL_MACHINE,r'SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\
↪Auto Update','AUOptions',0,REG_DWORD)

    print('Enable wuauserv')
    run_notfatal('sc config wuauserv start= auto')
    run_notfatal('net start wuauserv')
```

6.7 Utiliser WAPT

Cette section aborde l'utilisation globale de WAPT.

Toutes les fonctionnalités de WAPT pour les *Administrateurs*, les *Utilisateurs* et les *Déployeurs de Paquets* sont expliquées en détail.

6.7.1 Utiliser la console WAPT

Installer l'agent WAPT sur les postes utilisateur de votre Organisation

Si vous ne l'avez pas déjà fait, installer l'agent WAPT sur un poste utilisateur.

L'installation de l'agent WAPT sur un client va enclencher l'enregistrement du client dans l'inventaire du serveur WAPT.

Le client apparaîtra alors dans la console WAPT.

Pour installer l'agent WAPT manuellement sur un poste client, télécharger l'agent WAPT : https://srvwapt.mydomain.lan/wapt/waptagent.exe puis lancer son installation.

Note: Si vous avez sauté l'étape de création de l'agent WAPT, visitez la documentation pour *créer et déployer l'agent WAPT*.

Sur votre **poste de management**, dans la console **`waptconsole.exe`** doit apparaître le client dans la console.

Note: Si le poste ne remonte pas directement dans la console à l'issue de l'installation de l'agent, taper manuellement la commande `wapt-get register -S` sur le client.

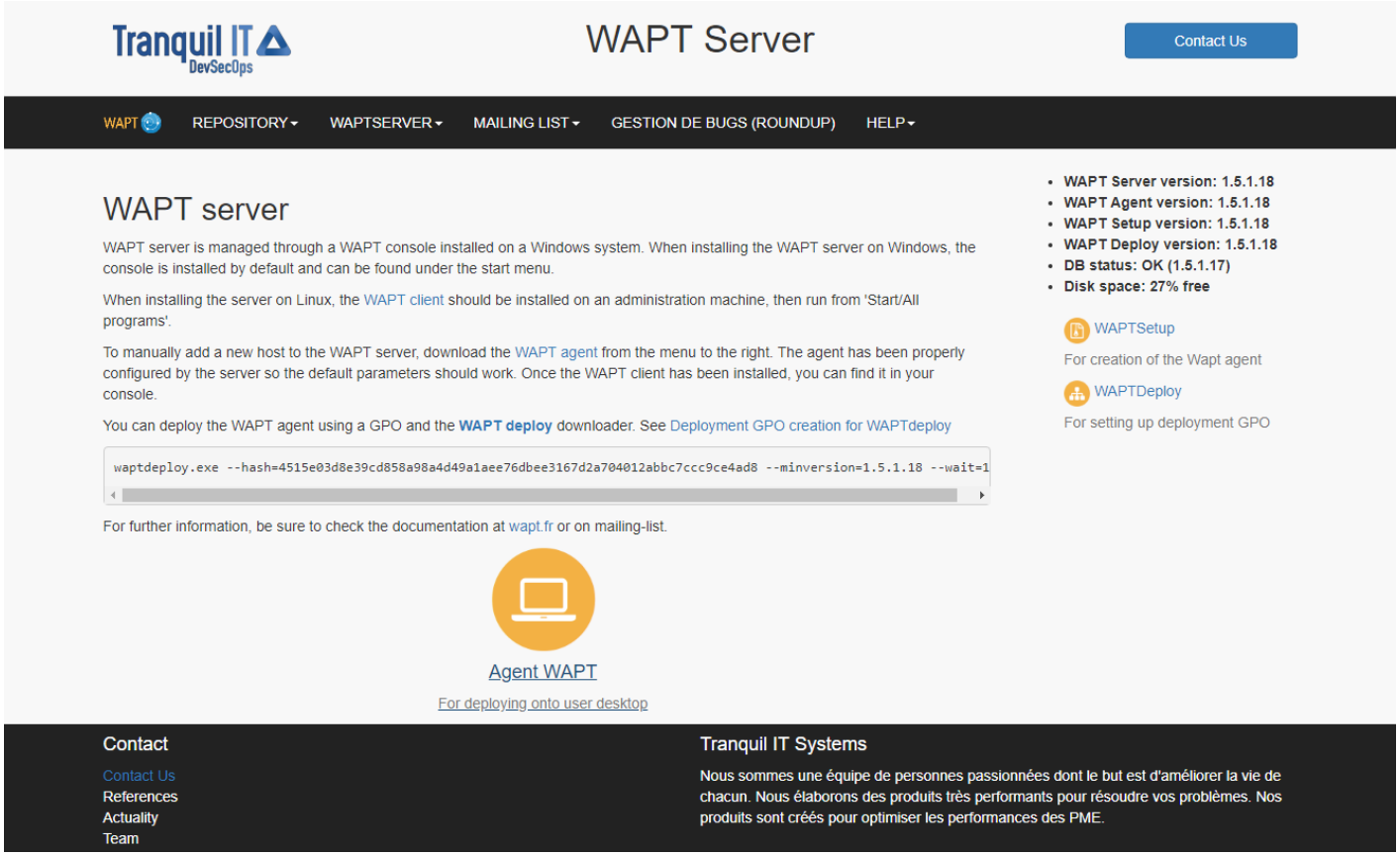


Figure71: Télécharger l'agent WAPT à déployer sur les clients

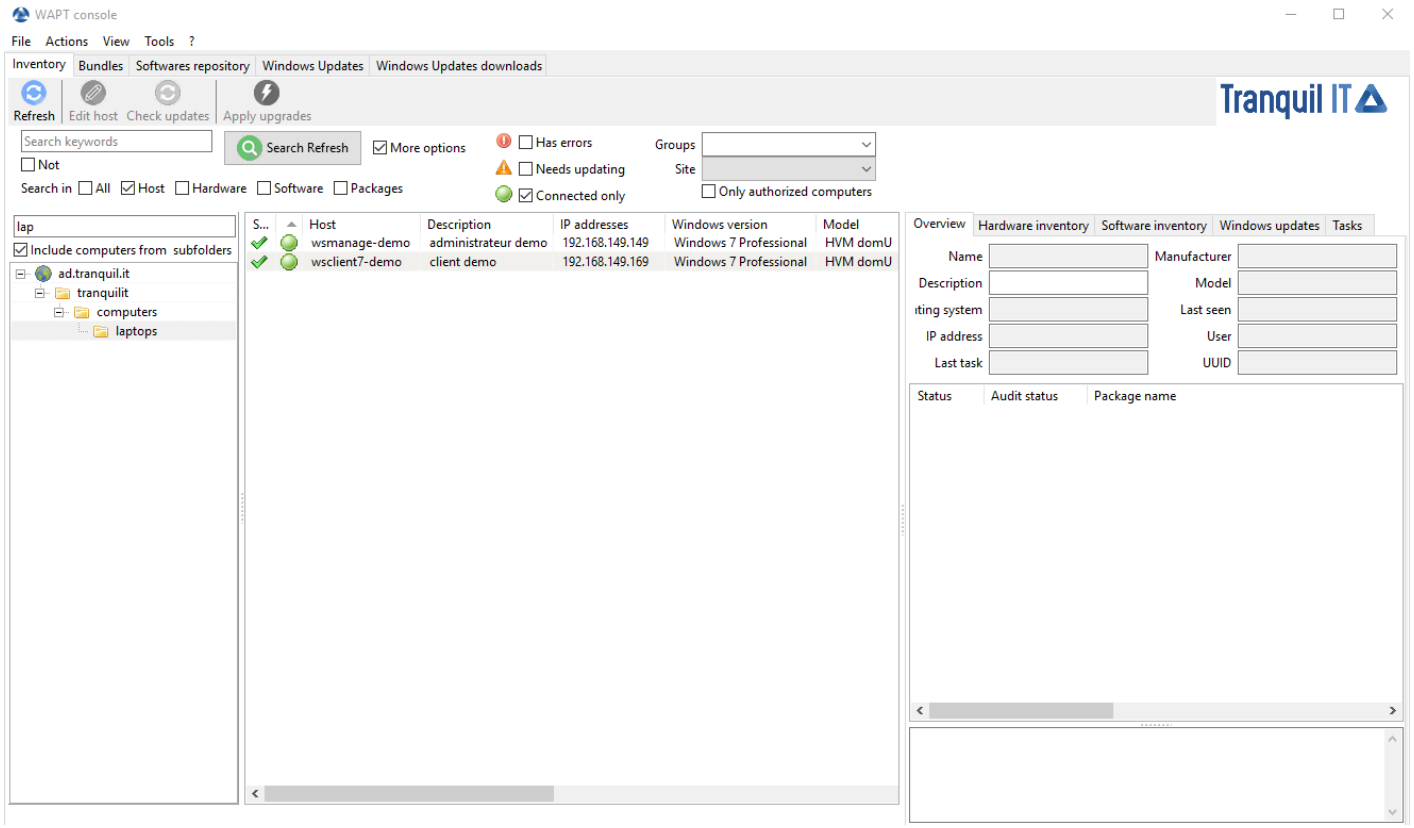


Figure72: Inventaire des clients enregistrés dans WAPT

Dupliquer des paquets

Principe de duplication de paquets

La duplication consiste à :

- importer un paquet WAPT existant depuis un dépôt externe ;
- changer le préfixe (par exemple de *tis* à *test*) ;
- re-signer le paquet avec votre clé privée d'*Administrateur* pour permettre le déploiement du paquet dupliqué sur vos postes clients ;
- et enfin, uploader celui-ci sur votre dépôt WAPT ;

Attention: En important un paquet dans votre dépôt, vous devenez responsable de ce paquet. **Il est signé avec votre signature**.

Tranquil IT décline toute responsabilité si vous utilisez des paquets WAPT récupérés depuis ses dépôts. Sans contrat de support, elle n'offre aucune garantie sur l'adéquation du paquet à votre usage ou la capacité du paquet à être conforme à aux politiques internes de sécurité de votre *Organisation*.

- se rendre dans l'onglet *Dépôt privé* ;

On obtient tous les paquets logiciels présents sur le dépôt WAPT.

Si aucun paquet n'a été importé, cette liste sera vide. Seul le paquet *test-waptupgrade* sera présent si l'agent WAPT a été généré précédemment. Consultez la documentation pour *créer un agent WAPT*.

Deux possibilités existent pour importer des paquets :

Importer depuis internet

Cette première méthode permet de télécharger des paquets directement depuis un dépôt WAPT extérieur à votre *Organisation*.

Pour importer depuis un dépôt différent de celui de Tranquil IT, indiquer dans les préférences l'adresse d'un autre dépôt WAPT. Par exemple : https://wapt.autredomaine.local/wapt/.

Note:

- Si aucun dépôt n'est renseigné, la valeur https://wapt.tranquil.it/wapt sera la valeur implicite.
- A partir de la version 1.3.12.13, la vérification de la validité du certificat SSL / TLS du dépôt extérieur est **activée par défaut**.

- cliquer sur *Importer* depuis internet ;

La liste affiche les paquets disponibles sur le serveur distant.

- pour importer un paquet, sélectionner un paquet puis *clic-droit* → *importer* ;
- valider la duplication dans le dépôt local ;
- cliquer sur *Oui* pour confirmer ;
- le téléchargement démarre . . .
- entrer le mot de passe de votre clé privée. . .

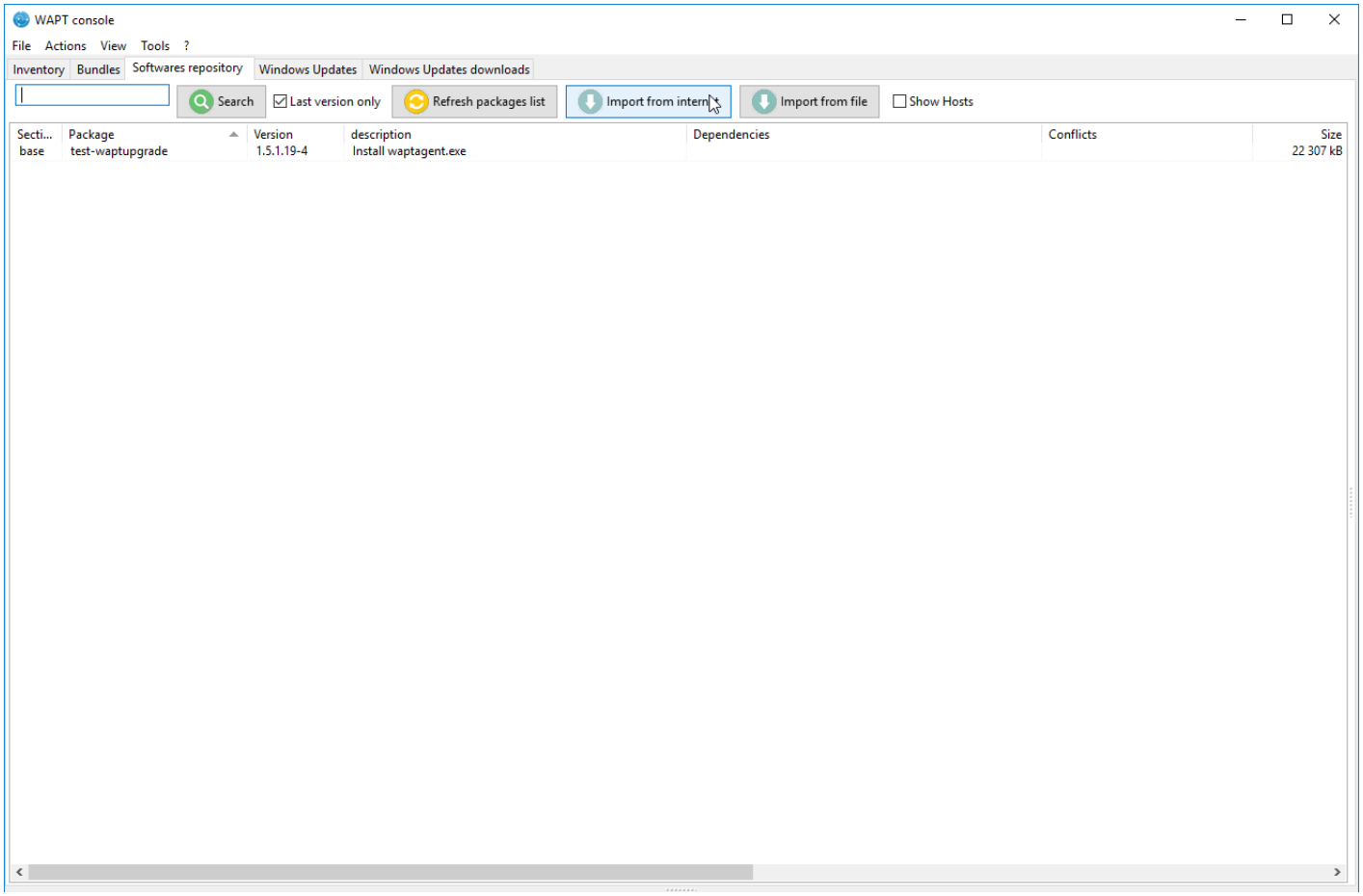


Figure73: Logiciels disponibles dans les dépôts dans la console WAPT

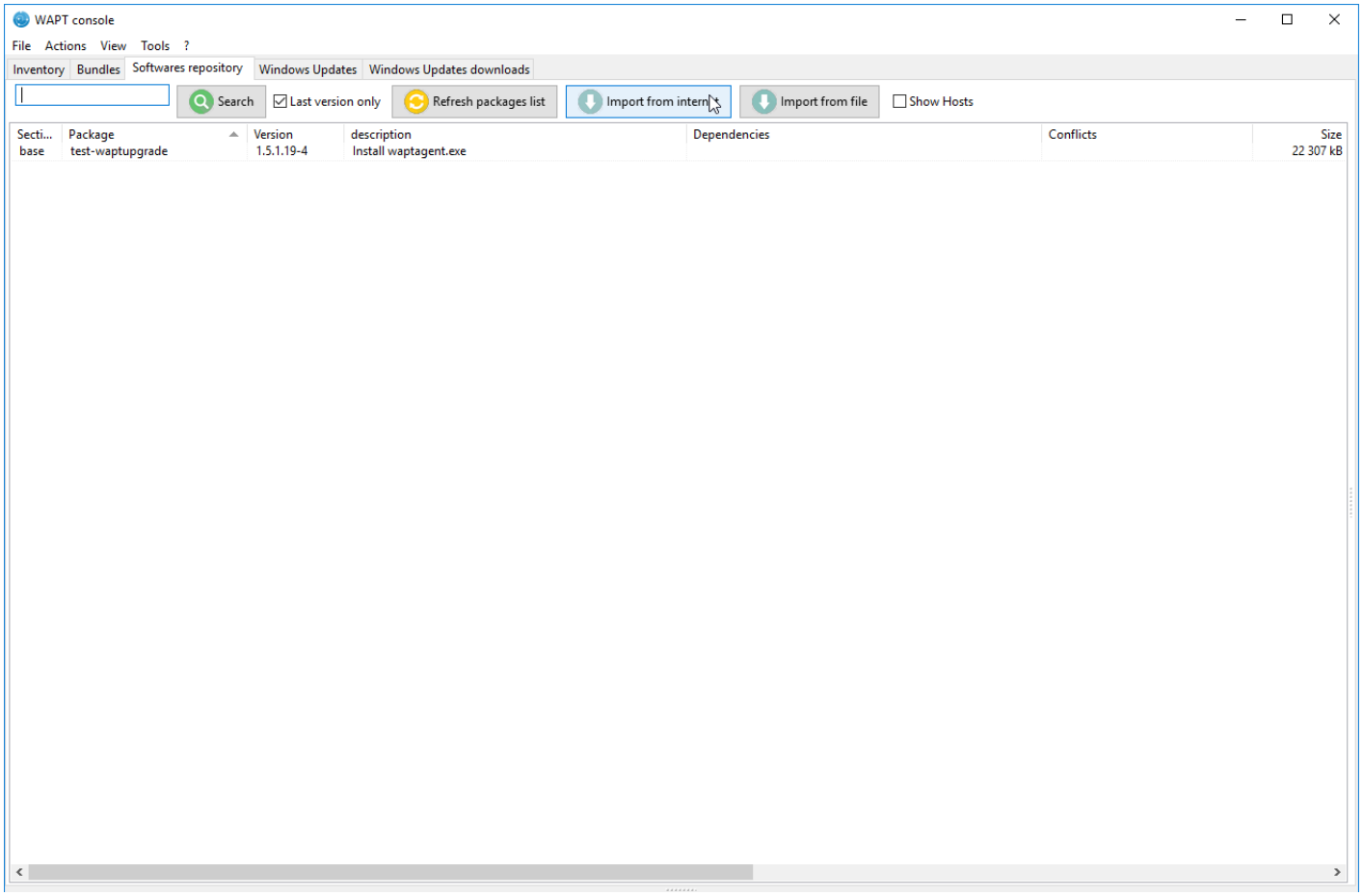


Figure74: Importer un paquet depuis internet

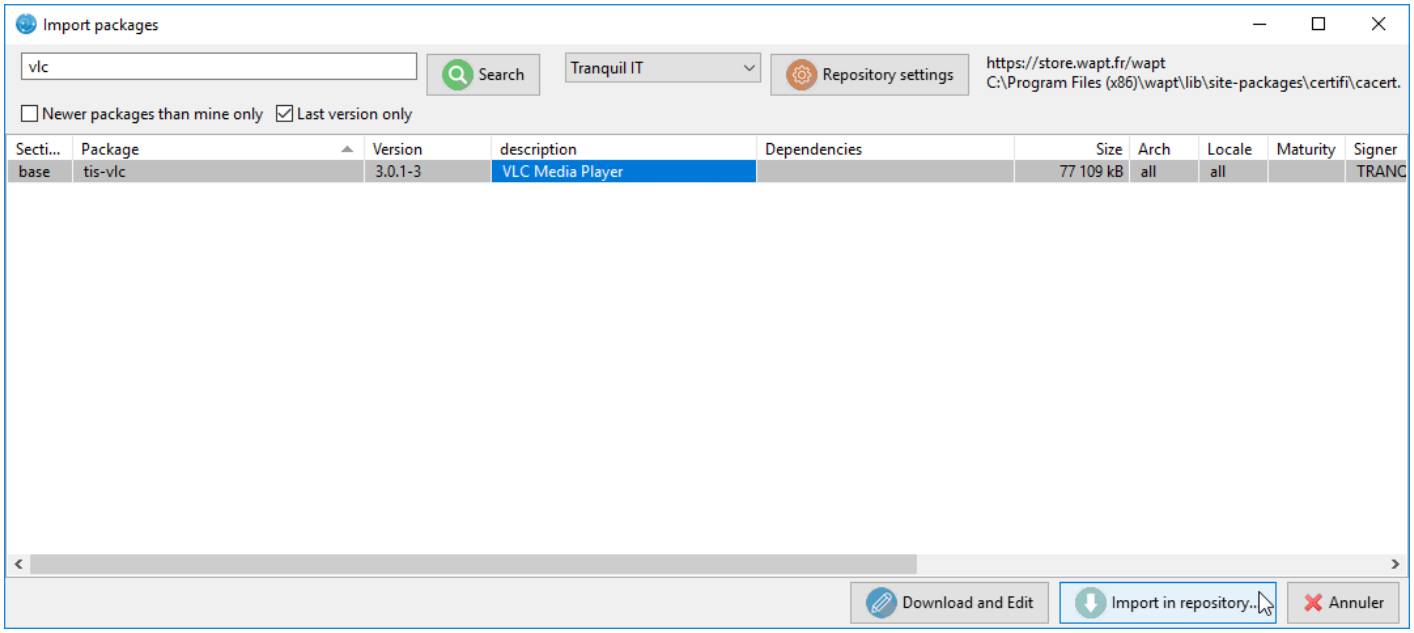


Figure75: Importer un paquet dans le dépôt

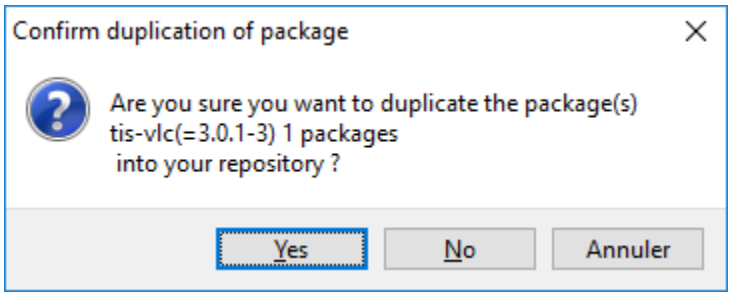


Figure76: Confirmer la duplication du paquet

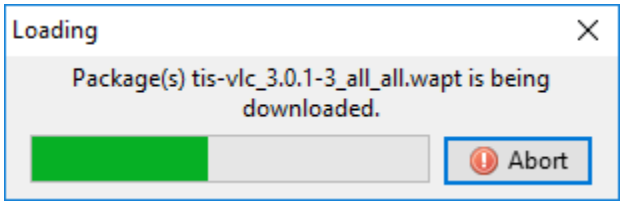


Figure77: Progression de la duplication du paquet

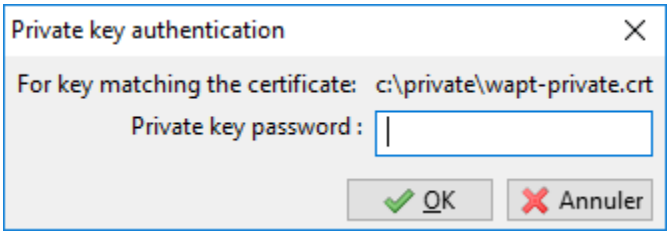


Figure78: Entrer le mot de passe de la clé privée

La console confirme la duplication du paquet dans le dépôt WAPT local.

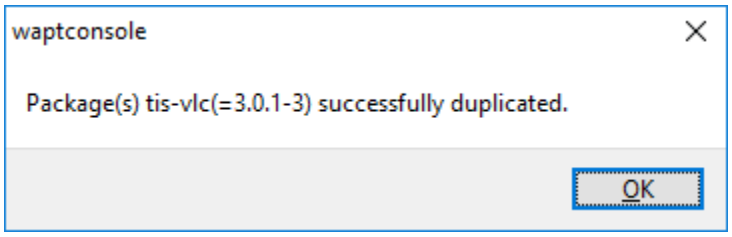


Figure79: Confirmation de la réussite de la duplication

Le paquet apparaît bien dans le dépôt WAPT local avec le préfixe de votre Organisation.

Attention: Si la vérification de la signature du paquet lors de l'import depuis Internet est activée, le certificat public doit être présent dans un des dossiers suivants :

- `C:\Program Files (x86)\wapt\ssl`;
- `%appdata%\waptconsole\ssl` ;

Si ce n'est pas le cas, l'erreur suivante se produira et empêchera l'import du paquet depuis ce dépôt.

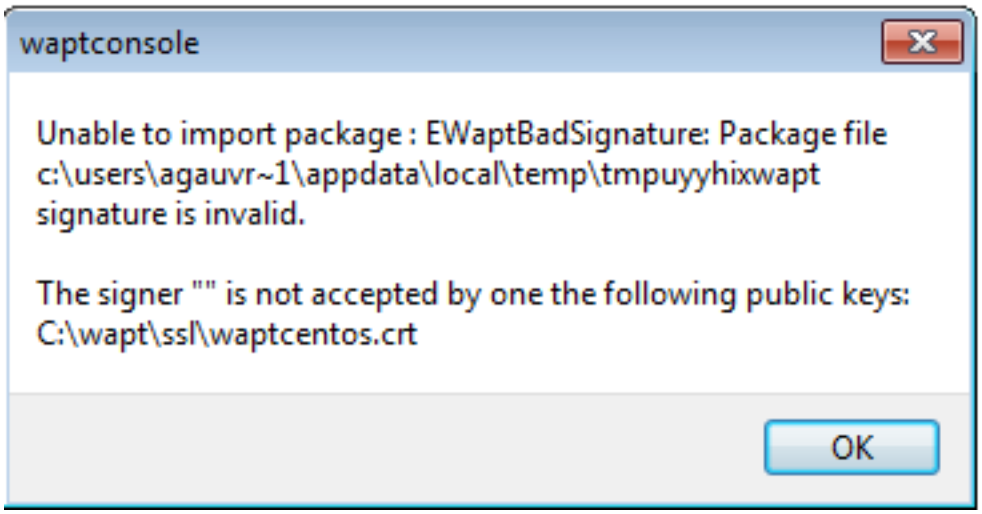


Figure81: Erreur lors de la validation de la signature du dépôt extérieur

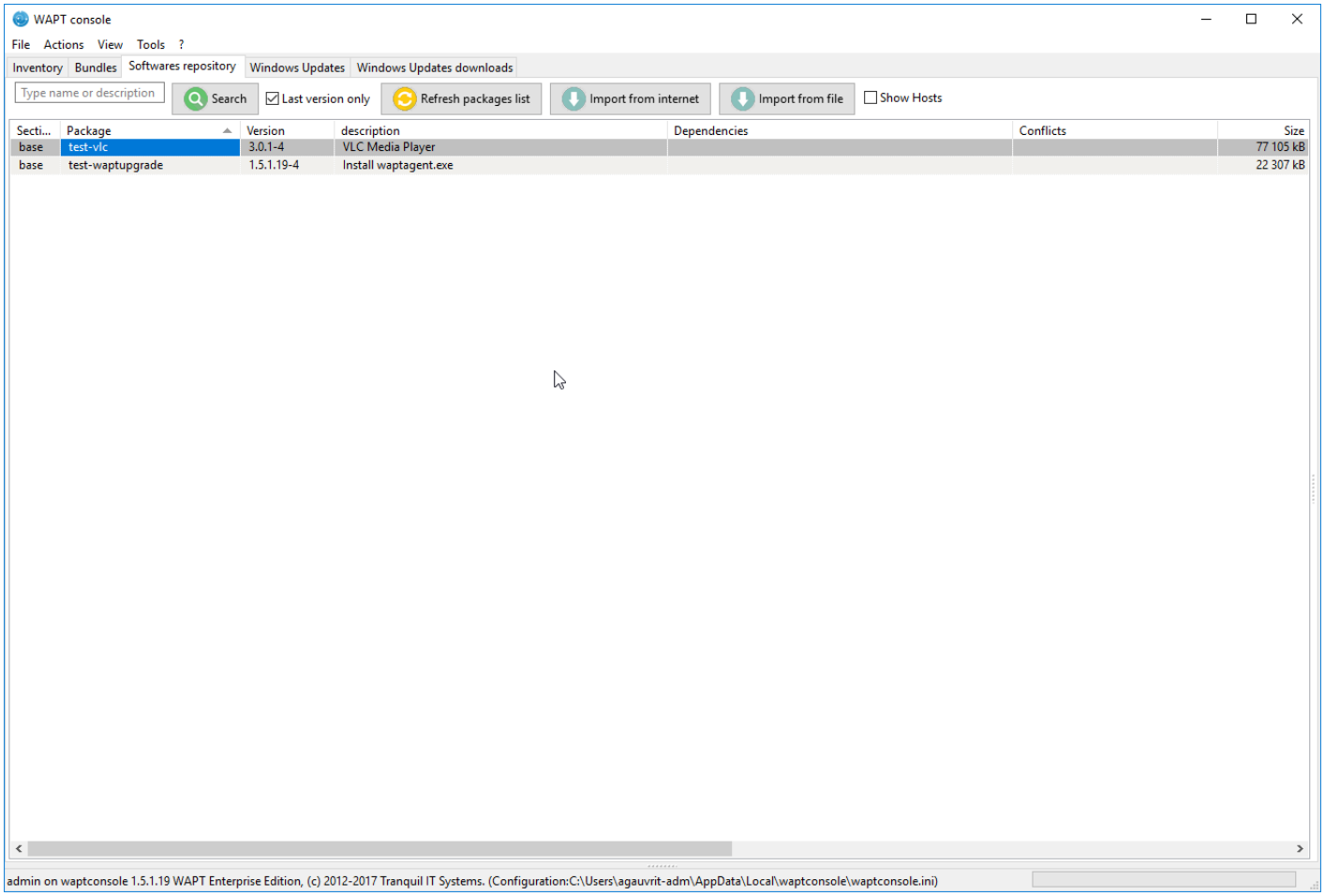


Figure80: Console WAPT affichant le paquet dupliqué

Editer un paquet avant de l'importer

Avec la sortie de WAPT 1.3.12.13, il est désormais possible d'éditer directement un paquet depuis le dépôt externe avant de l'importer dans le dépôt local.

Pour cela, choisir la deuxième option *Télécharger et éditer* lors de l'import du paquet depuis Internet.

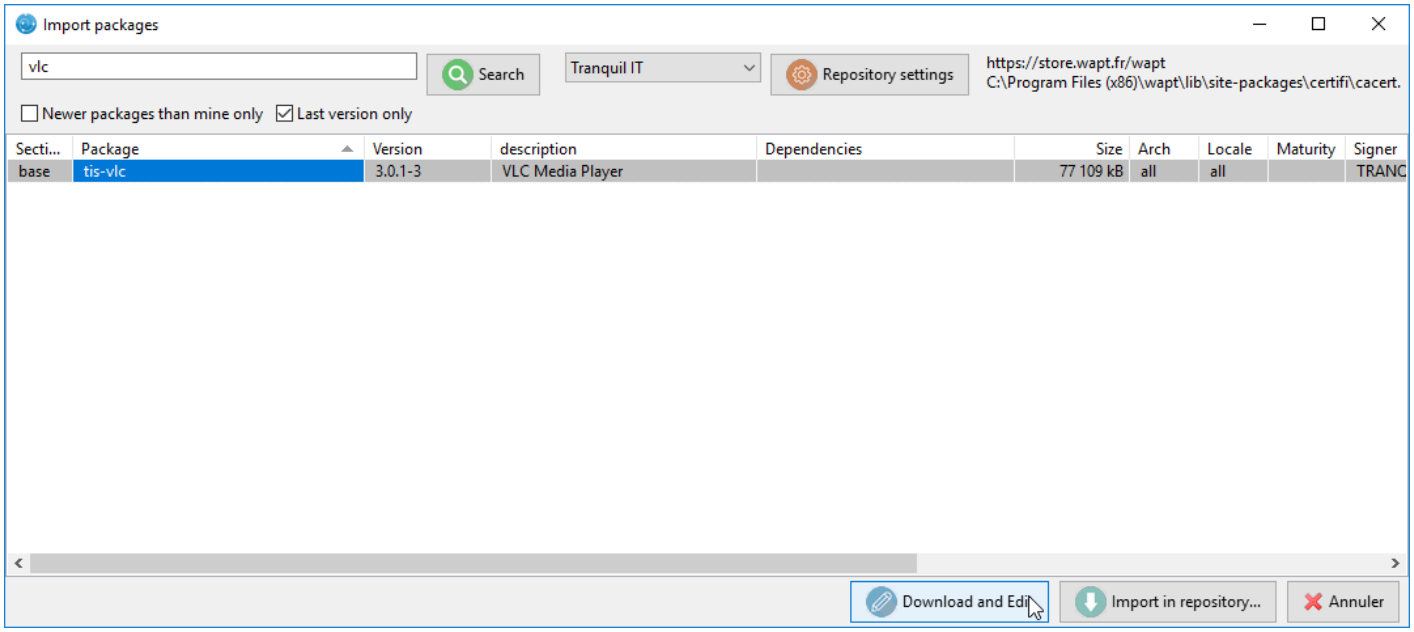


Figure82: Procédure pour télécharger et éditer un paquet

L'éditeur **`PyScripter`**, si installé au préalable, s'ouvre alors avec le projet WAPT du paquet.

Voir la documentation pour savoir comment *créer des paquets WAPT*.

Importer depuis un fichier

Cette deuxième méthode permet d'indiquer un fichier `.wapt` stocké sur n'importe quel support.

- cliquer sur *Importer* depuis un fichier ;
- sélectionner le fichier pour l'importer ;
- cliquer sur *Ouvrir* pour importer le fichier ;

La console confirme la duplication du paquet dans le dépôt WAPT local.

Le paquet apparaît bien dans le dépôt WAPT local avec le préfixe de votre Organisation.

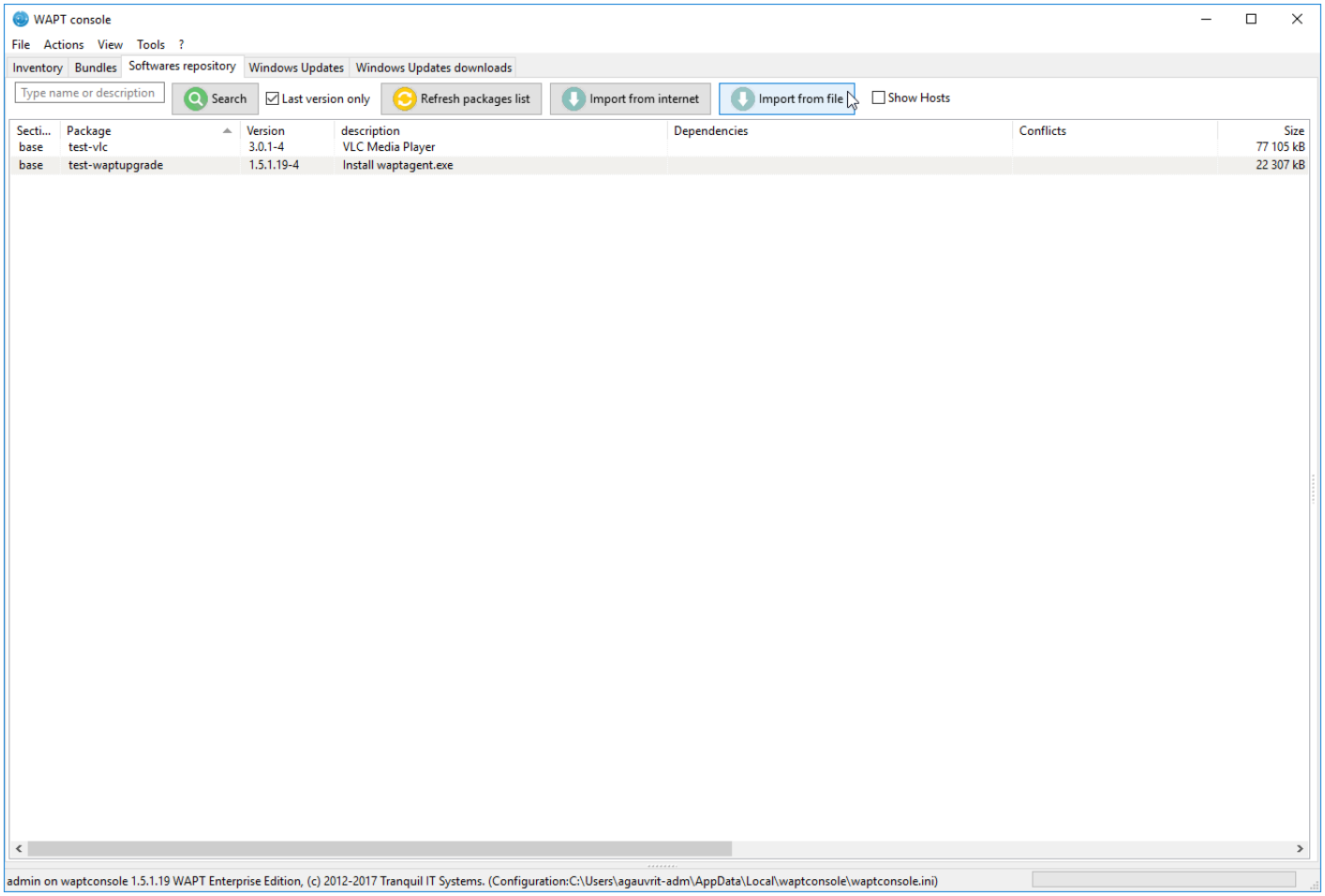


Figure83: Importer à partir d'un fichier

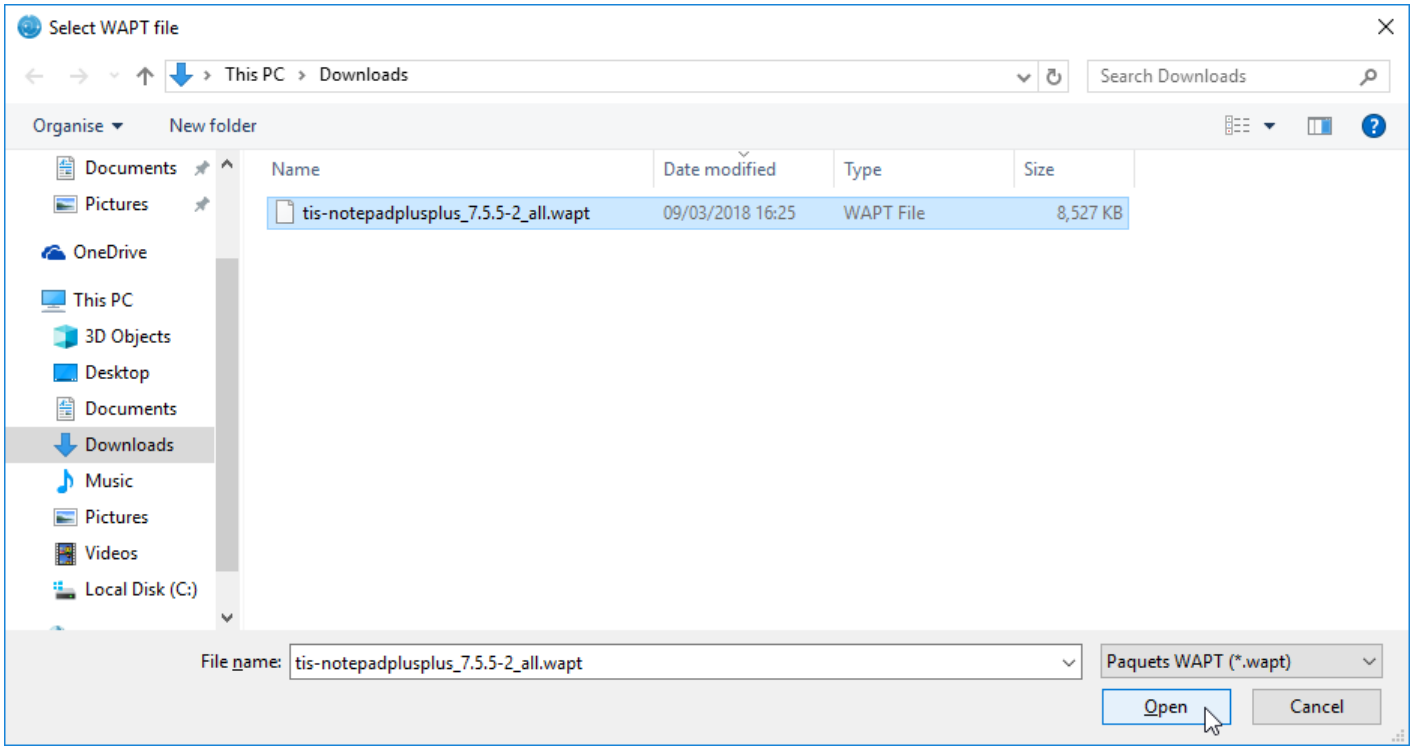


Figure84: Sélectionner le fichier à importer

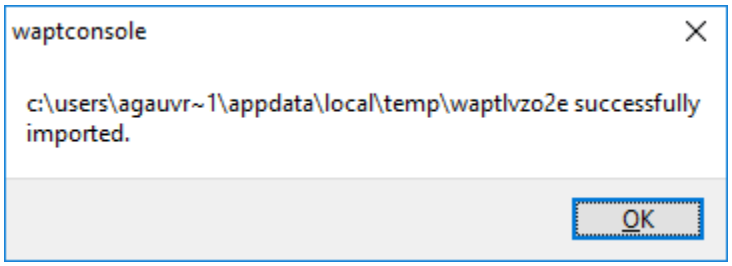


Figure85: Importation du fichier réussie

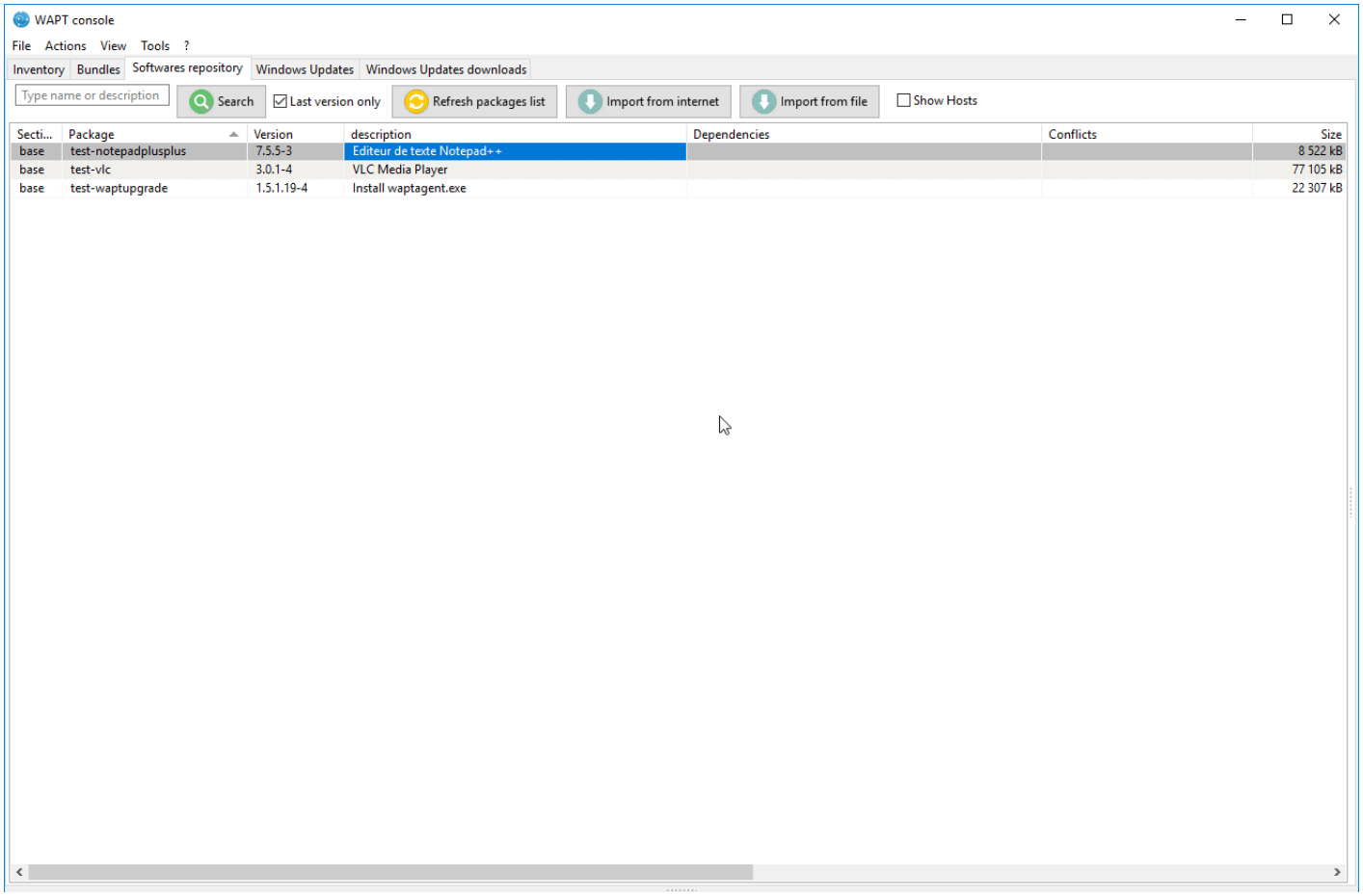


Figure86: Importer un paquet dans le dépôt

Changer le préfixe et re-signer le paquet

Lors d'un import, le changement de préfixe et la re-signature sont transparents et automatiques.

Une fois que le paquet est prêt, il est transféré vers le dépôt WAPT local.

Déployer des paquets depuis la console

- éditer le poste sur lequel le logiciel sera déployé ;

Note: La sélection multiple en utilisant les touches habituelles `Control-A` `Shift-Flèche` est possible.

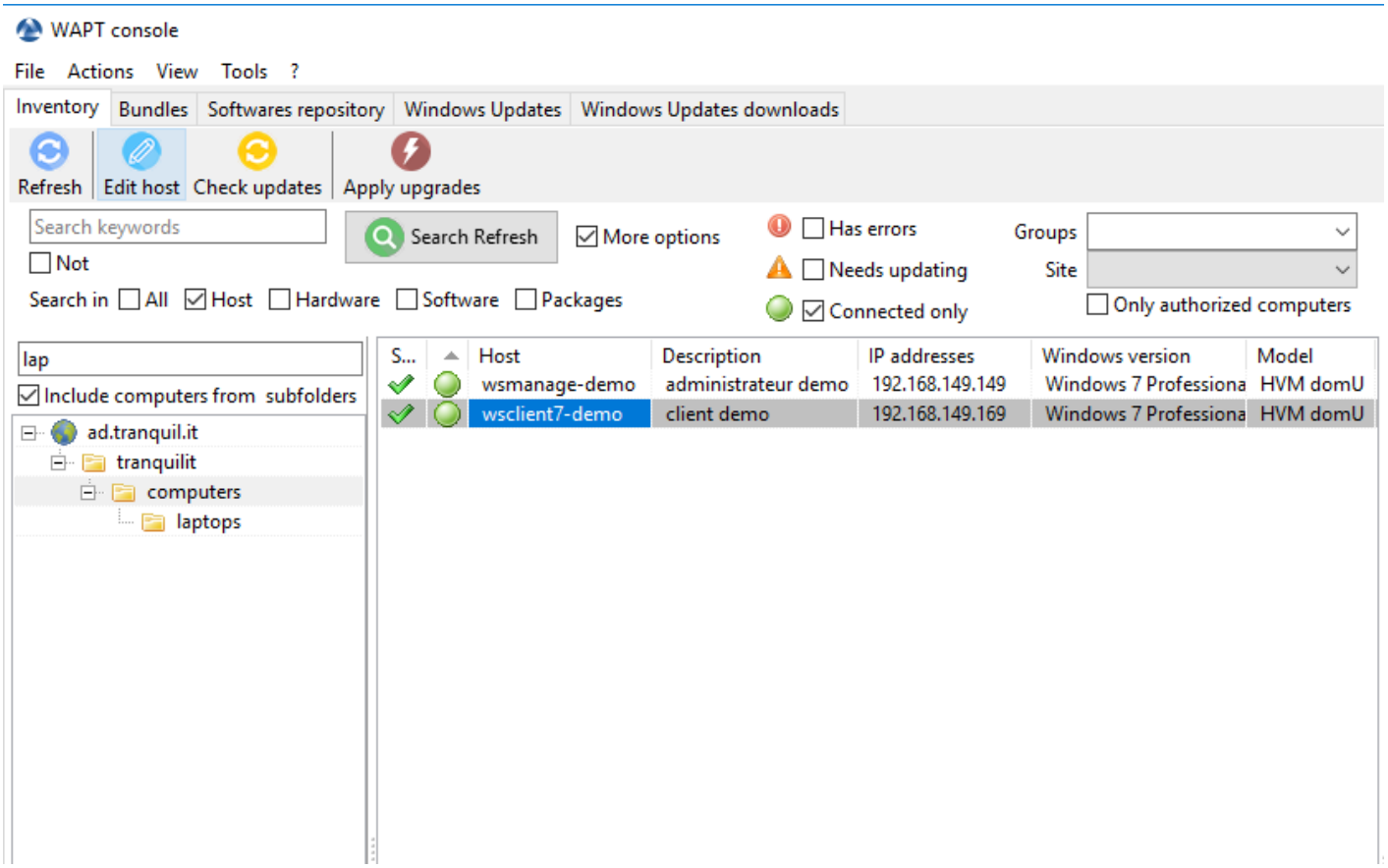


Figure87: Sélectionner le poste à configurer

- Une fenêtre s'ouvre, à droite apparaît la liste des paquets disponibles sur le dépôt WAPT local et à gauche la liste des paquets actuellement attribués au poste.
- glisser-déposer le paquet souhaité depuis l'encart de droite vers l'encart de gauche ;
- cliquer sur *Enregistrer et Appliquer à la machine* lance immédiatement l'installation sur le poste concerné ;
- cliquer sur *Enregistrer* pour enregistre la nouvelle configuration. La mise à jour sera appliquée au prochain cycle de mises à jour de l'agent WAPT ;

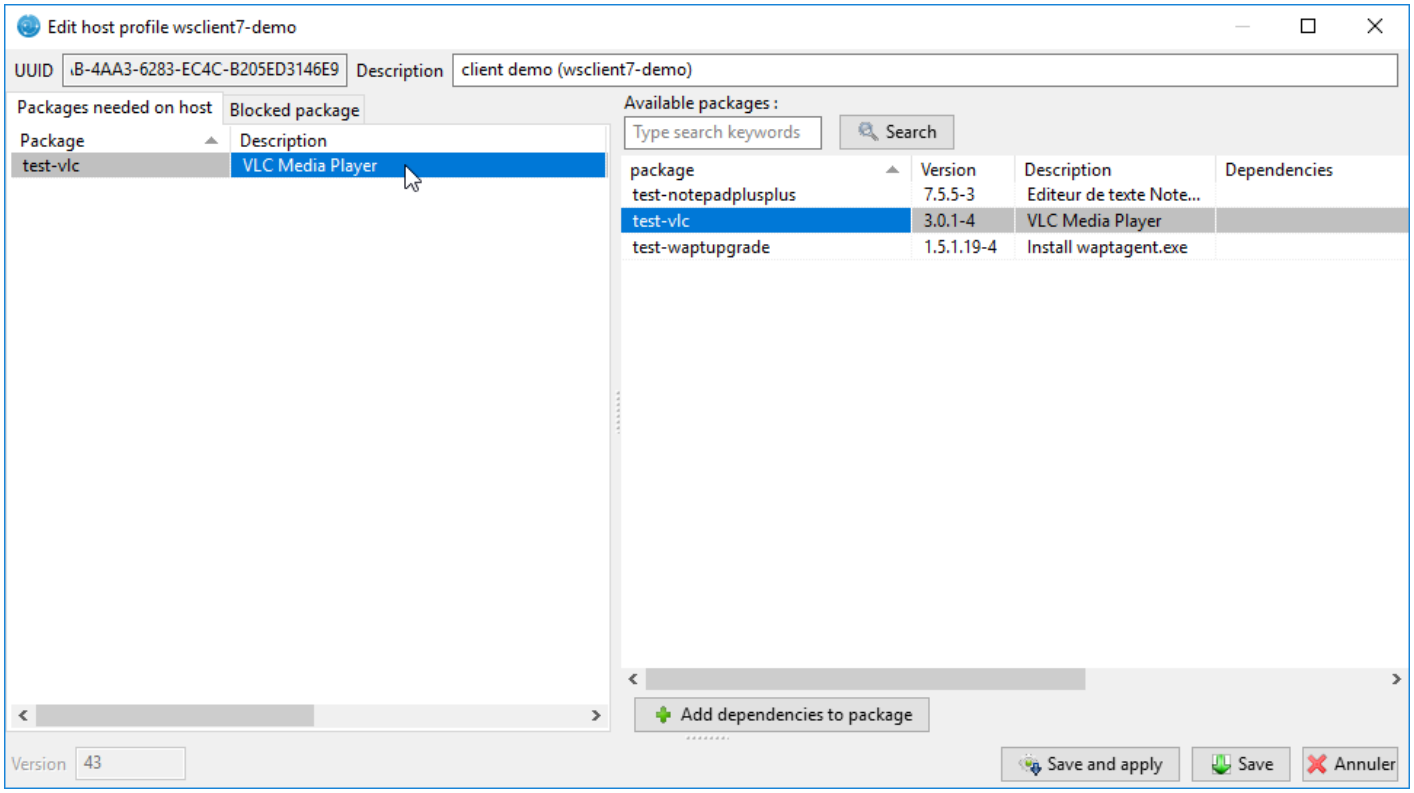


Figure88: Glisser-déposer le paquet sur le poste ou la sélection de postes

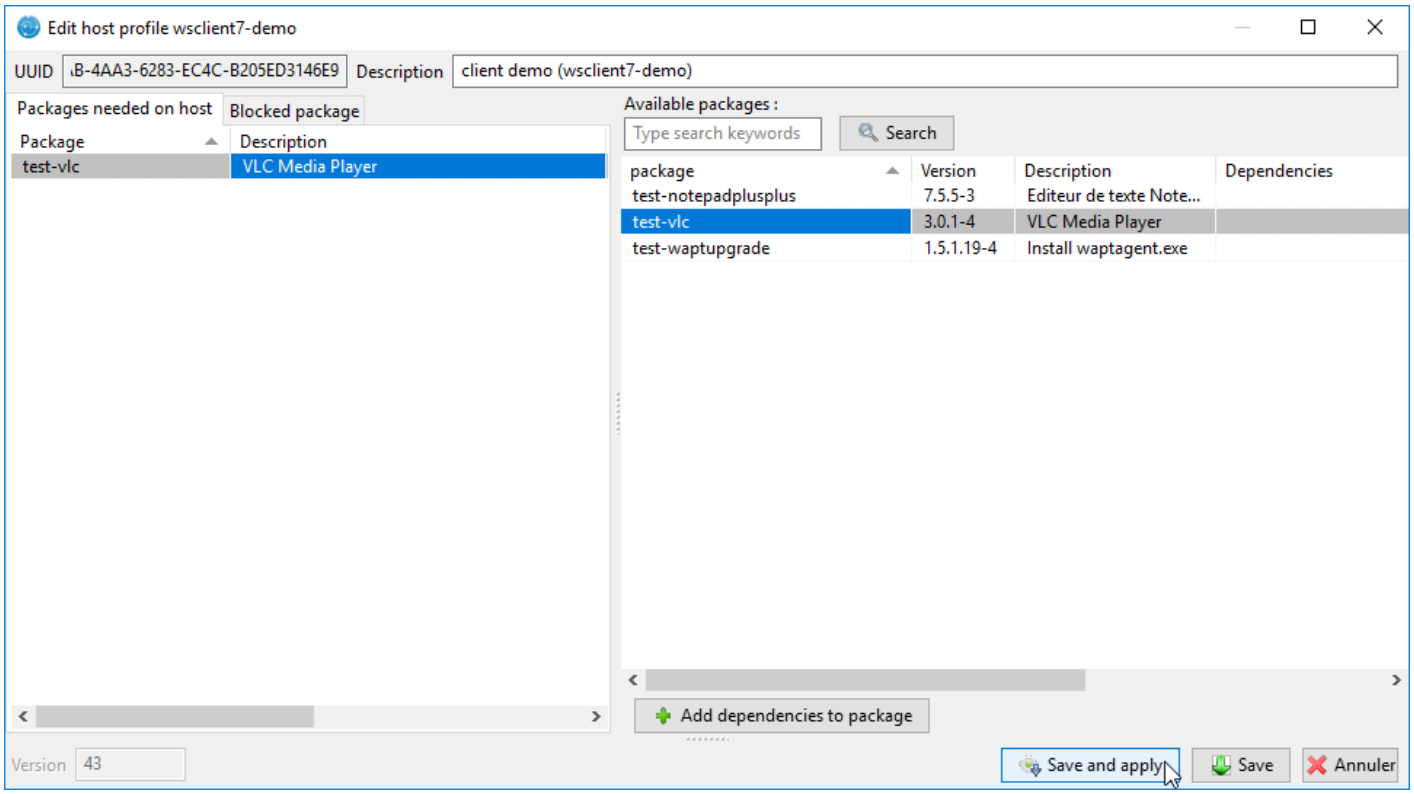


Figure89: Enregistrer et appliquer la configuration à la machine

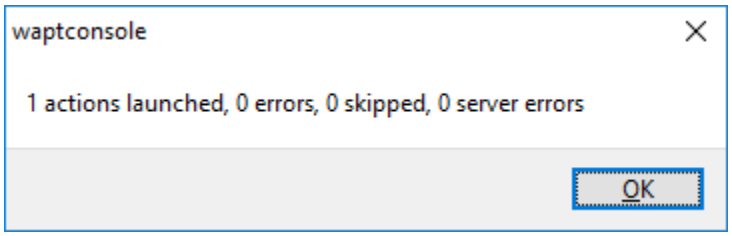


Figure90: Mise à jour lancée

Pour lancer l'installation des paquets, cliquer successivement sur *Lancer la mise à jour des paquets disponibles* puis *Lancer l'installation des paquets*.

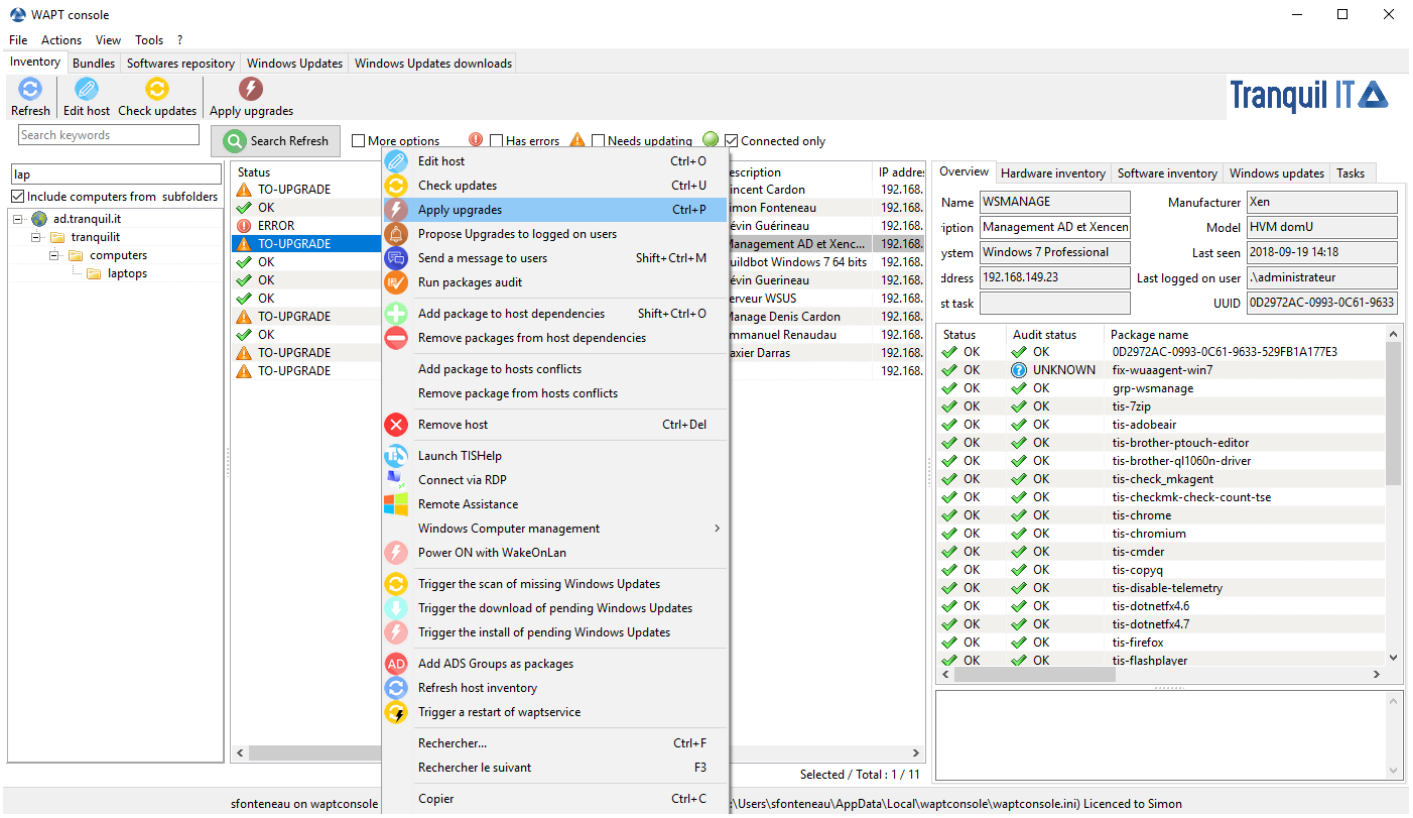


Figure91: Lancer la mise à jour des paquets disponibles

L'installation du paquet WAPT se lance sur le poste ou les postes sélectionnés.

6.7.2 Utiliser la console WAPT (version détaillée)

Note: Certaines fonctionnalités détaillés ici sont disponible uniquement en version **Entreprise**.

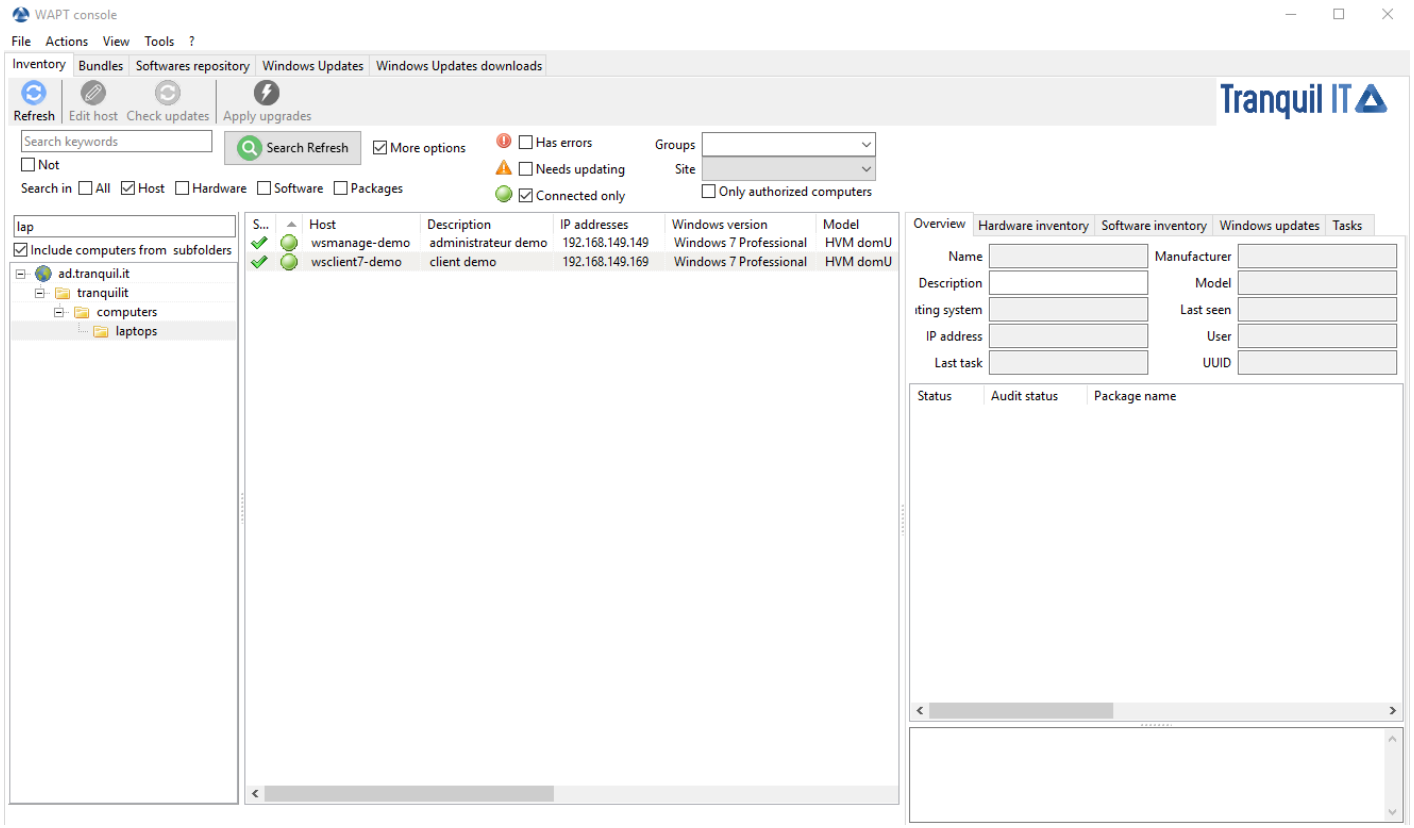


Figure92: Inventaire des clients enregistrés dans WAPT

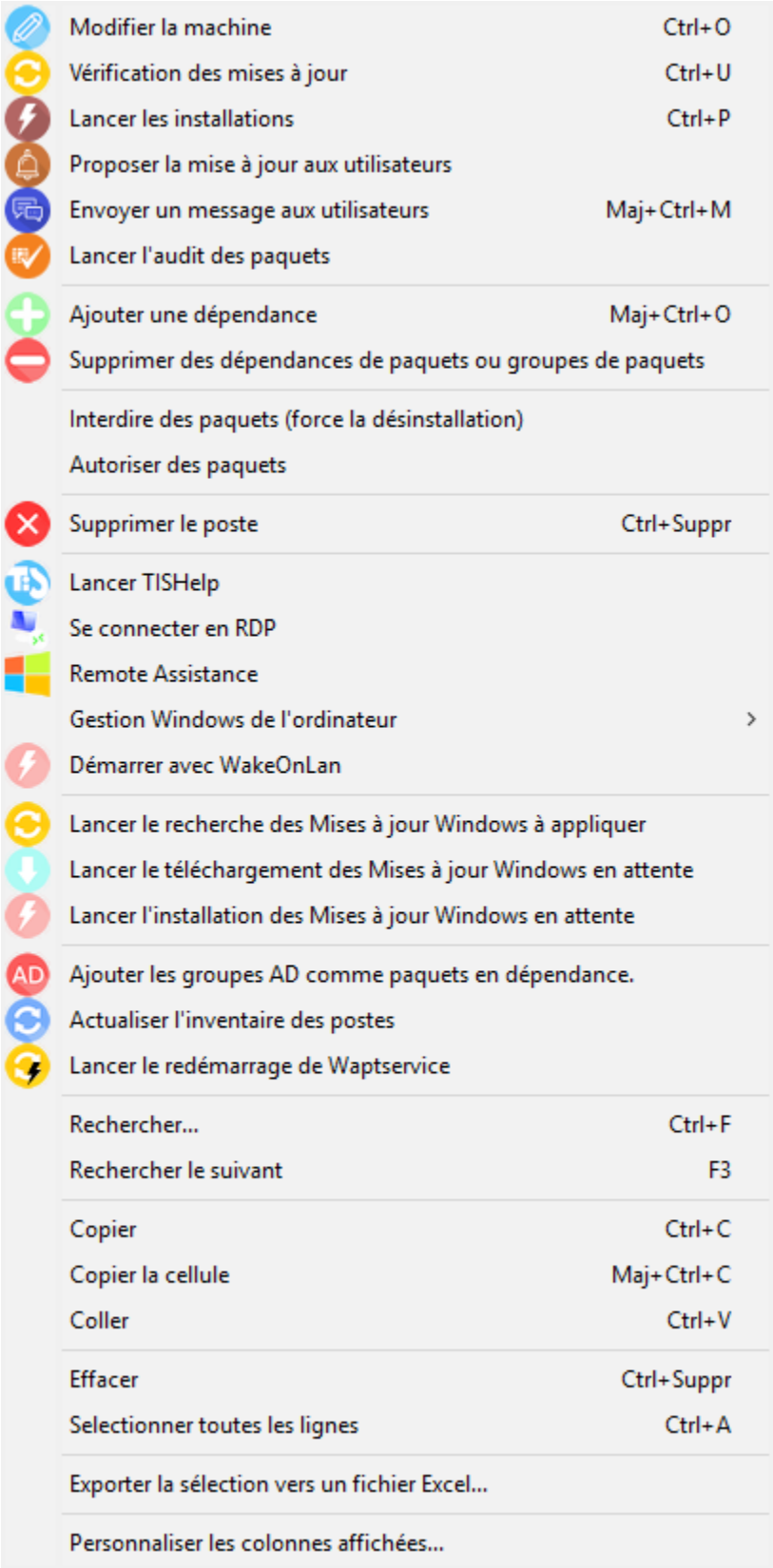


Figure93: Menu de configuration des postes

Agir sur les machines

Table16: Liste des fonctions de configuration des Postes client dans la console WAPT

Description	Multi-sélection
Modifier la liste des paquets de la machine	*yes*
Actualiser la liste des paquets disponibles	*yes*
Proposer aux utilisateurs de lancer un upgrade	*yes*
Envoyer un message aux machines sélectionnées	*yes*
Auditer les paquets installés sur les postes sélectionnés	*yes*
Ajouter un ou plusieurs paquets aux machines sélectionnées	*yes*
Retirer un ou plusieurs paquets aux machines sélectionnées	*yes*
Interdire l'installation des paquets sur les machines	*yes*
Ajouter un ou plusieurs paquets aux machines sélectionnées	*yes*
Supprimer la machine et/ou le paquet machine	*yes*
Lancer TISHelp sur la machine distante	*no*
Lancer une connexion VNC avec la machine distante	*no*
Se connecter avec Veyon	*no*
Lancer une connexion RDP avec le poste sélectionné	*no*
Lancer une assistance à distance (msra.exe)	*no*
Gérer les machines avec **`compmgmt.msc`**	*no*
Gérer les Utilisateurs et les Groupes avec **`lusrmgr.msc`**	*no*
Gérer les services avec **`services.msc`**	*no*
Envoyer une requête WakeOnLAN aux machines sélectionnées	*yes*
Rechercher des mises à jour Windows à appliquer	*yes*
Télécharger les mise à jour Windows en attente	*yes*
Installer les mises à jour Windows en attente	*yes*
Ajouter les groupes AD des machines sélectionnées en dépendance	*yes*
Lancer une remontée d'inventaire des machines sélectionnées	*yes*
Redémarrer le service WAPT	*yes*

Rechercher une machine

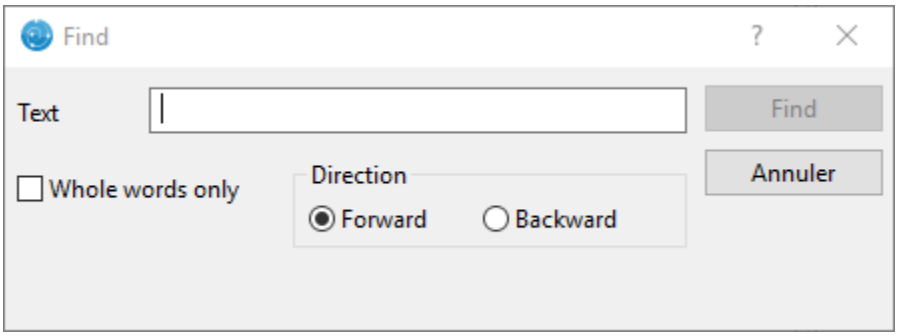


Figure94: Fonction *Rechercher* de WAPT

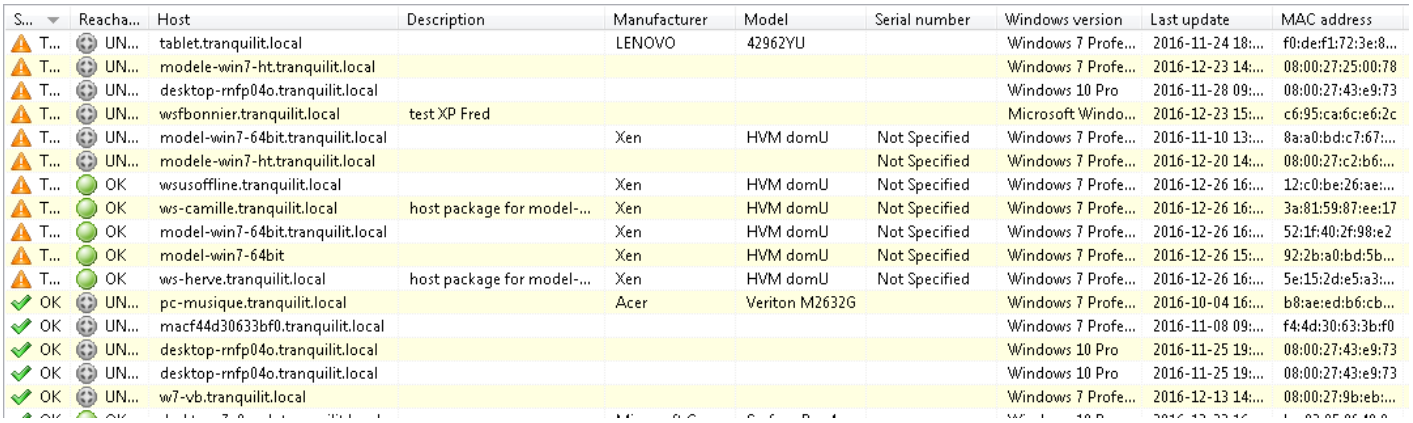
Permet de rechercher une valeur dans la colonne sélectionnée.

Afficher l'inventaire

Quand les clients lancent leur **`register`**, ils remontent certaines informations au serveur WAPT.

Les informations affichées dans la console ne sont donc pas actualisées en temps réel, il faudra régulièrement actualiser l'affichage dans la console pour faire apparaître les nouvelles informations.

Cliquer sur le bouton *Actualiser* ou appuyer sur la touche `F5` du clavier.



S...	Reacha...	Host	Description	Manufacturer	Model	Serial number	Windows version	Last update	MAC address
T...	UN...	tablet.tranquilit.local		LENOVO	42962YU		Windows 7 Profe...	2016-11-24 18:...	f0:de:f1:72:3e:8...
T...	UN...	modele-win7-ht.tranquilit.local					Windows 7 Profe...	2016-12-23 14:...	08:00:27:25:00:78
T...	UN...	desktop-rnfp04o.tranquilit.local					Windows 10 Pro	2016-11-28 09:...	08:00:27:43:e9:73
T...	UN...	wsfbonnier.tranquilit.local	test XP Fred				Microsoft Windo...	2016-12-23 15:...	c6:95:ca:6c:e6:2c
T...	UN...	model-win7-64bit.tranquilit.local		Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-11-10 13:...	8a:a0:bd:c7:67:...
T...	UN...	modele-win7-ht.tranquilit.local				Not Specified	Windows 7 Profe...	2016-12-20 14:...	08:00:27:c2:b6:...
T...	OK	wsusoffline.tranquilit.local		Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-12-26 16:...	12:c0:be:26:ae:...
T...	OK	ws-camille.tranquilit.local	host package for model-...	Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-12-26 16:...	3a:81:59:87:ee:17
T...	OK	model-win7-64bit.tranquilit.local		Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-12-26 16:...	52:1f:40:2f:98:e2
T...	OK	model-win7-64bit		Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-12-26 15:...	92:2b:a0:bd:5b...
T...	OK	ws-herve.tranquilit.local	host package for model-...	Xen	HVM domU	Not Specified	Windows 7 Profe...	2016-12-26 16:...	5e:15:2d:e5:a3:...
OK	UN...	pc-musique.tranquilit.local		Acer	Veriton M2632G		Windows 7 Profe...	2016-10-04 16:...	b8:ae:ed:b6:cb...
OK	UN...	macf44d30633bf0.tranquilit.local					Windows 7 Profe...	2016-11-08 09:...	f4:4d:30:63:3b:f0
OK	UN...	desktop-rnfp04o.tranquilit.local					Windows 10 Pro	2016-11-25 19:...	08:00:27:43:e9:73
OK	UN...	desktop-rnfp04o.tranquilit.local					Windows 10 Pro	2016-11-25 19:...	08:00:27:43:e9:73
OK	UN...	w7-vb.tranquilit.local					Windows 7 Profe...	2016-12-13 14:...	08:00:27:9b:eb:...

Figure95: Console WAPT affichant l'inventaire

La console liste les machines inscrites sur le serveur WAPT et certaines informations utiles à leur gestion.

Sélectionner une machine dans la liste affiche ses détails d'inventaire dans l'onglet *Inventaire matériel* et *Inventaire logiciel* dans la partie droite de la console.

Consulter l'inventaire matériel et inventaire logiciel des machines

Les informations affichées par défaut dans l'onglet *Inventaire matériel* sont :

- le nom de la machine ;
- la description de la machine ;
- le système d'exploitation ;
- l'adresse IP de la machine ;
- la dernière tâche WAPT exécutée ;
- le constructeur de la machine ;
- le modèle de machine ;
- la date de la dernière mise à jour de la machine ;
- l'utilisateur connecté actuellement ou dernièrement sur le poste ;

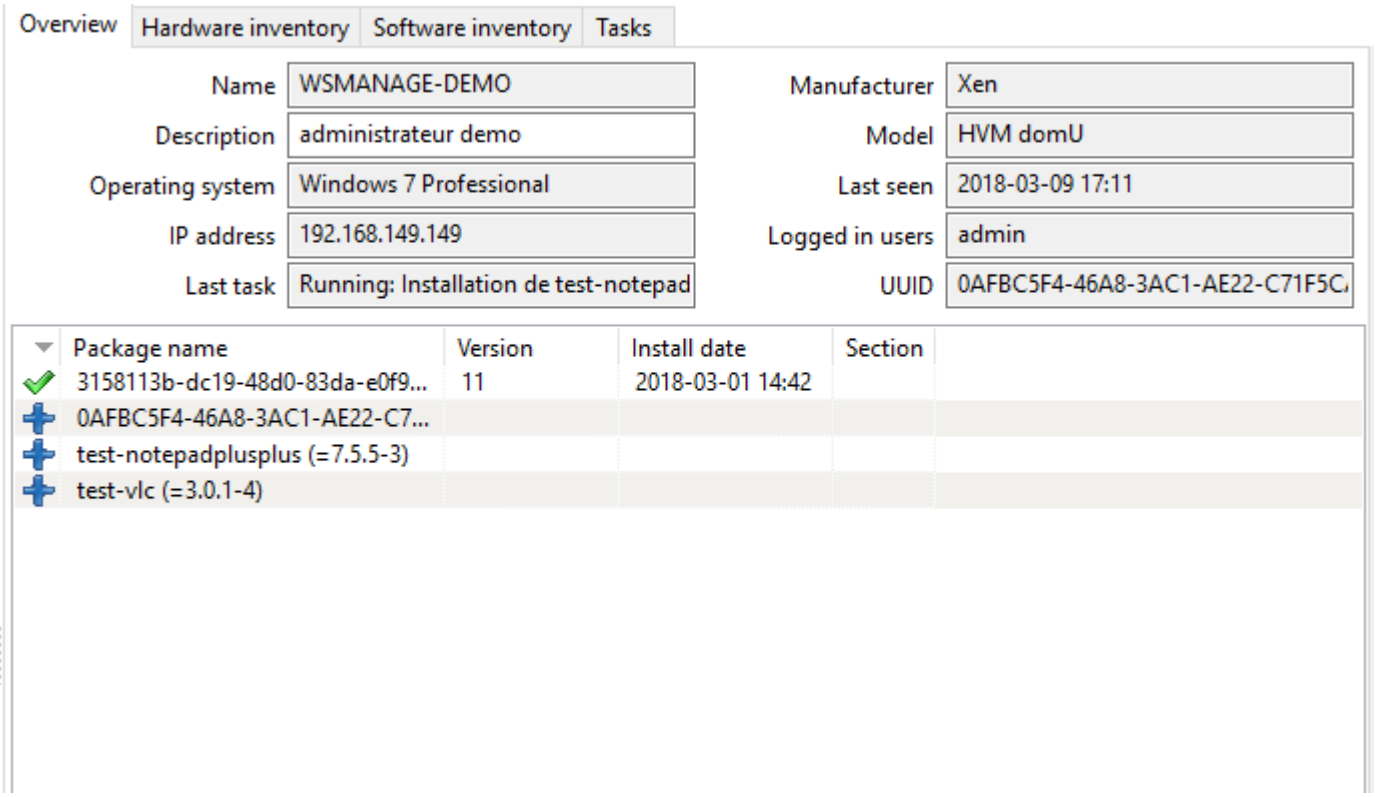


Figure96: Résumé de l'état de la machine

Table17: Statut des paquets dans la console WAPT

Description	Statut
La liste des paquets WAPT installés.	Statut : **OK**
Liste des paquets en attente d'installation	Statut : **MISSING**
Liste des paquets en attente d'une mise à jour	Statut : **NEED-UPGRADE**
Liste des paquets dont l'installation a échoué	Statut : **ERROR**

Lorsqu'un paquet est en statut `ERROR`, cliquer sur le paquet pour afficher l'erreur remontée.

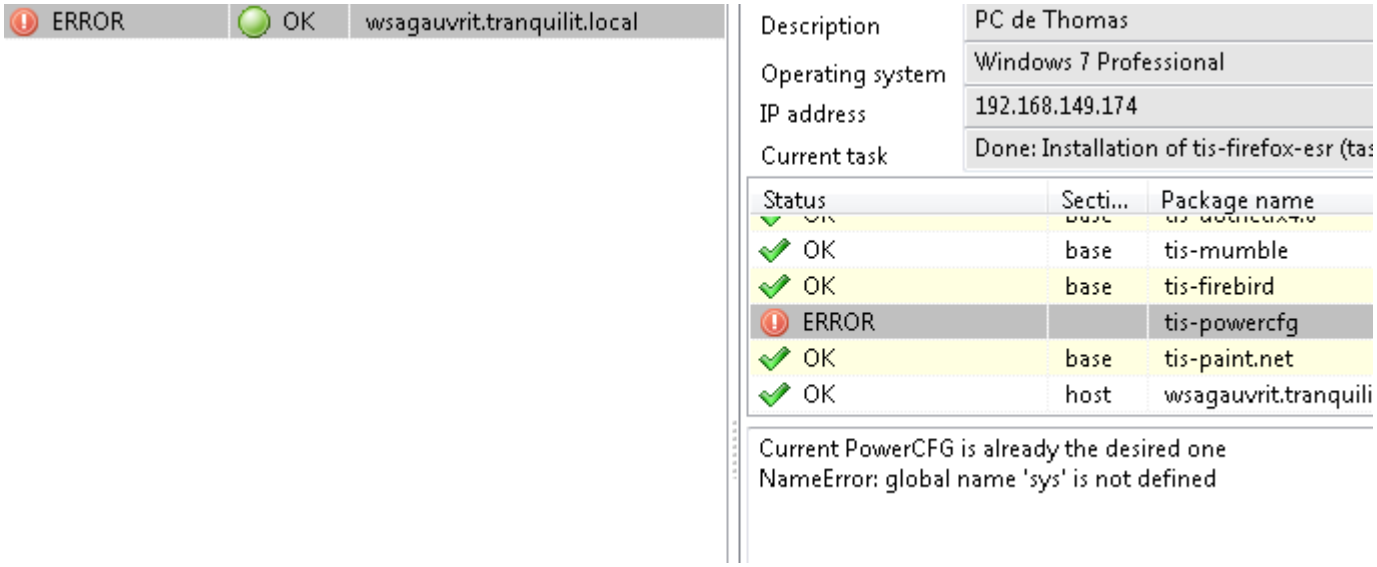


Figure97: Détail de l'erreur

Agir sur les paquets installés sur la machine

Indication:

- la sélection multiple est possible ;
- la machine doit être joignable au moment de l'action ;
- si plusieurs machines sont sélectionnées, l'action sera lancée sur toutes les machines sélectionnées ;

Table18: Agir sur les paquets installés sur la machine

Action	Description
Installer un paquet	installe le paquet sélectionné sur les machines sélectionnées
Forcer un paquet	force la réinstallation d'un paquet sélectionné sur les machines sélectionnées
Supprimer un paquet	désinstalle le paquet sélectionné sur les machines sélectionnées
Oublier un paquet	demande aux machines sélectionnées de ne plus utiliser WAPT pour gérer le paquet sélectionné
Auditer le paquet	lance un audit forcé du paquet sélectionné

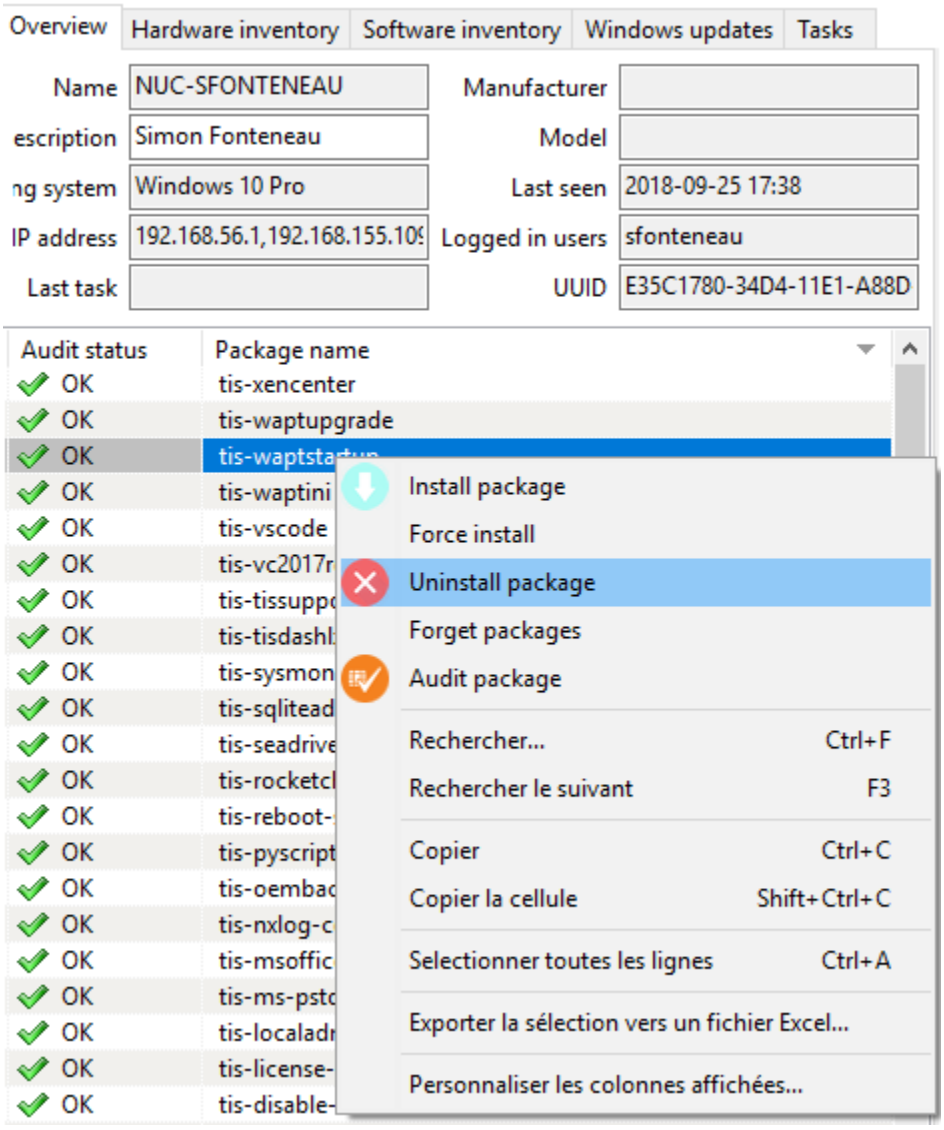


Figure98: Action possibles pour les paquets WAPT

Onglet Matériel

Les informations affichées par défaut dans l'onglet *Inventaire matériel* sont :

- les informations sur les composants matériels de la machine ;
- diverses informations sur la machine ;
- diverses informations sur l'état de WAPT ;

Un *Filtre* permet de rechercher des machines.

Indication: Les filtres fonctionnent en expression régulière.

Pour ajouter une colonne dans la vue grille, glisser-déposer une propriété d'inventaire matériel depuis la grille *Inventaire Matériel* vers la grille principale.

Exemple : glisser la propriété `physical_memory` sur la partie gauche, et la colonne *physical_memory* apparaît dans la grille principale.

Onglet Logiciel

Les informations affichées par défaut dans l'onglet *Inventaire logiciel* sont :

- *éditeur* ;
- *nom du logiciel* ;
- *version du logiciel* ;
- *date d'installation* ;
- *clé de désinstallation* ;
- *chaîne de désinstallation* ;

Onglet Windows Updates

Les informations affichées par défaut dans l'onglet *Windows Updates* sont :

- la version de l'agent Windows Updates ;
- la date du dernier scan Windows Updates ;
- la durée du dernier scan ;
- le status de WAPTWUA ;
- la date de modification du dernier `wsusscn2.cab` traité par WAPT ;
- status de WAPTWUA Enabled (True / False) ;

  La grille affiche ensuite la liste des cab Windows installés ou en attente d'installation.

Les informations affichées par défaut la grille de l'onglet *Windows Updates* sont :

- *Status* ;
- *Product* ;

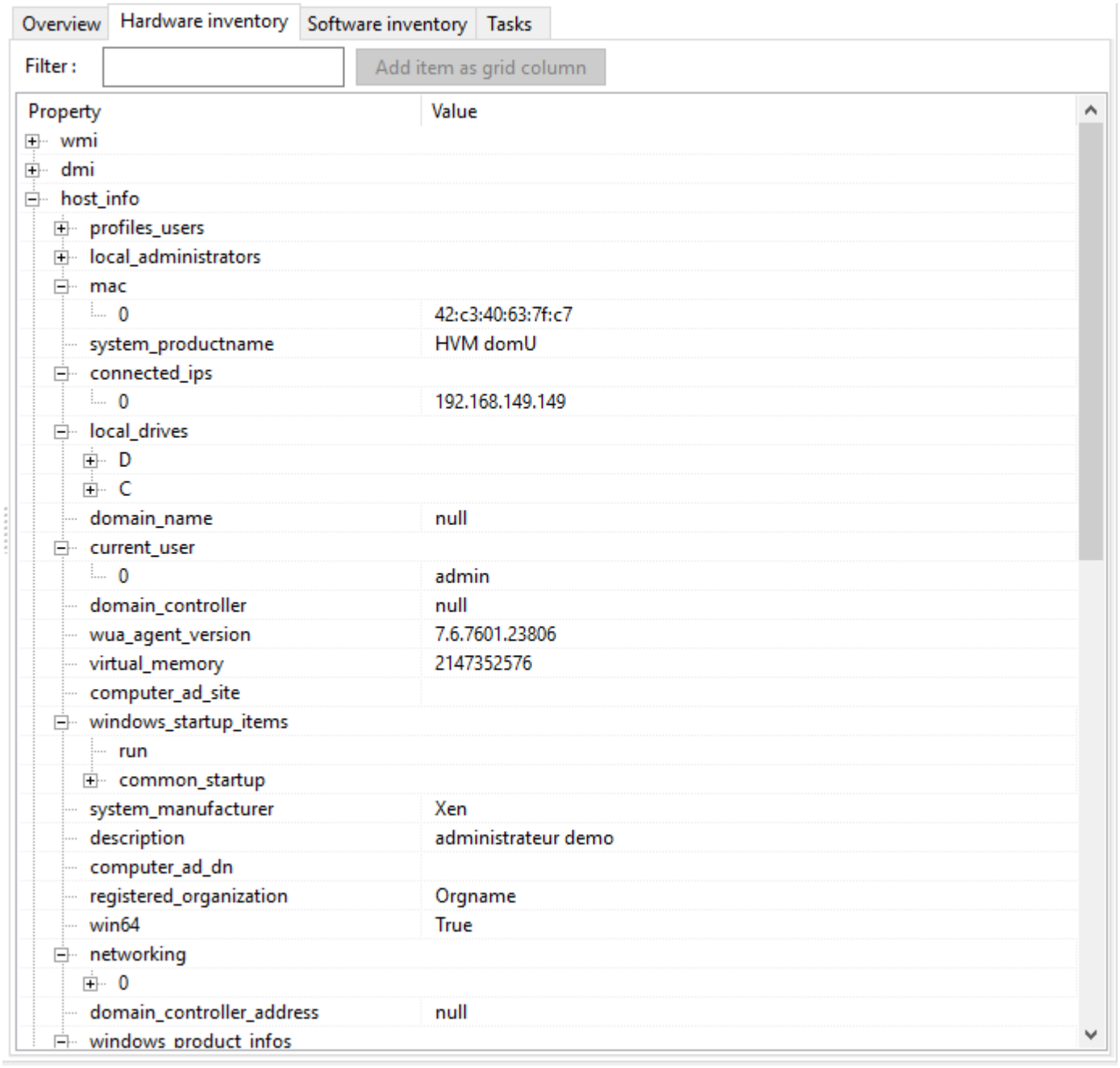


Figure99: Inventaire matériel d'une machine

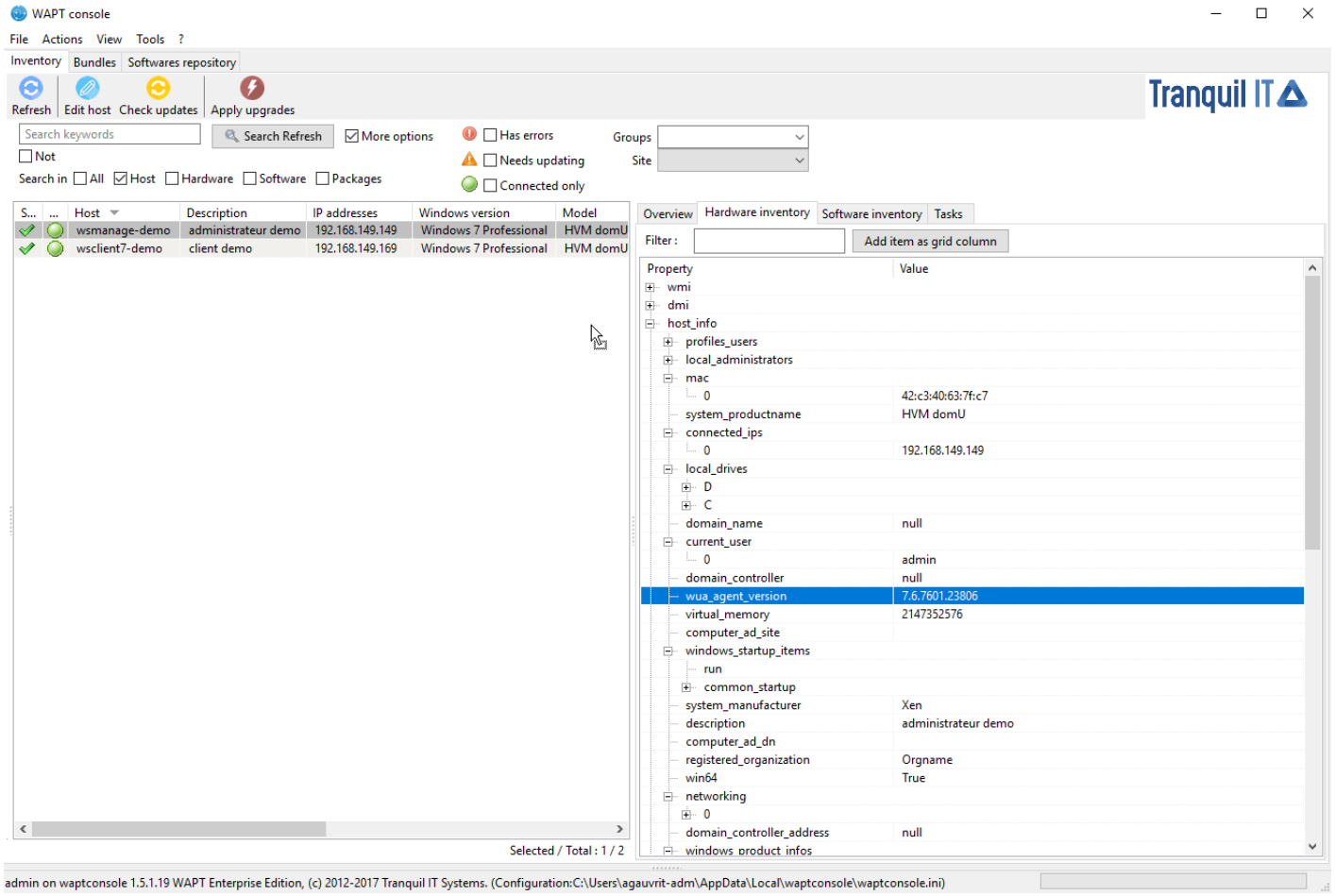
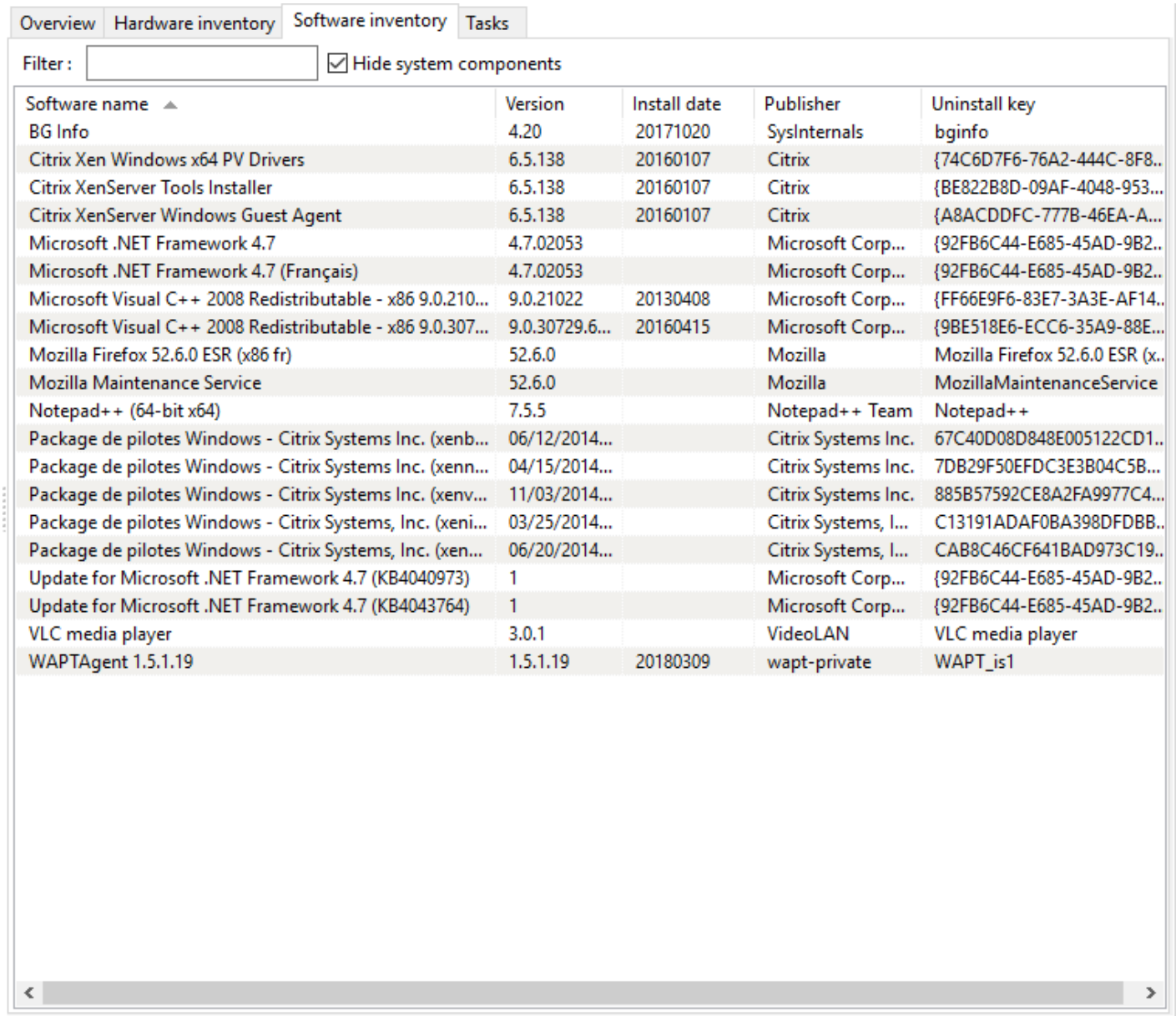


Figure100: Ajouter un critère à la grille principale de la console WAPT



Software name	Version	Install date	Publisher	Uninstall key
BG Info	4.20	20171020	SysInternals	bginfo
Citrix Xen Windows x64 PV Drivers	6.5.138	20160107	Citrix	{74C6D7F6-76A2-444C-8F8..
Citrix XenServer Tools Installer	6.5.138	20160107	Citrix	{BE822B8D-09AF-4048-953...
Citrix XenServer Windows Guest Agent	6.5.138	20160107	Citrix	{A8ACDDFC-777B-46EA-A...
Microsoft .NET Framework 4.7	4.7.02053		Microsoft Corp...	{92FB6C44-E685-45AD-9B2..
Microsoft .NET Framework 4.7 (Français)	4.7.02053		Microsoft Corp...	{92FB6C44-E685-45AD-9B2..
Microsoft Visual C++ 2008 Redistributable - x86 9.0.210...	9.0.21022	20130408	Microsoft Corp...	{FF66E9F6-83E7-3A3E-AF14..
Microsoft Visual C++ 2008 Redistributable - x86 9.0.307...	9.0.30729.6...	20160415	Microsoft Corp...	{9BE518E6-ECC6-35A9-88E...
Mozilla Firefox 52.6.0 ESR (x86 fr)	52.6.0		Mozilla	Mozilla Firefox 52.6.0 ESR (x..
Mozilla Maintenance Service	52.6.0		Mozilla	MozillaMaintenanceService
Notepad++ (64-bit x64)	7.5.5		Notepad++ Team	Notepad++
Package de pilotes Windows - Citrix Systems Inc. (xenb...	06/12/2014...		Citrix Systems Inc.	67C40D08D848E005122CD1..
Package de pilotes Windows - Citrix Systems Inc. (xenn...	04/15/2014...		Citrix Systems Inc.	7DB29F50EFDC3E3B04C5B...
Package de pilotes Windows - Citrix Systems Inc. (xenv...	11/03/2014...		Citrix Systems Inc.	885B57592CE8A2FA9977C4...
Package de pilotes Windows - Citrix Systems, Inc. (xeni...	03/25/2014...		Citrix Systems, I...	C13191ADAF0BA398DFDBB..
Package de pilotes Windows - Citrix Systems, Inc. (xen...	06/20/2014...		Citrix Systems, I...	CAB8C46CF641BAD973C19..
Update for Microsoft .NET Framework 4.7 (KB4040973)	1		Microsoft Corp...	{92FB6C44-E685-45AD-9B2..
Update for Microsoft .NET Framework 4.7 (KB4043764)	1		Microsoft Corp...	{92FB6C44-E685-45AD-9B2..
VLC media player	3.0.1		VideoLAN	VLC media player
WAPTAgent 1.5.1.19	1.5.1.19	20180309	wapt-private	WAPT_is1

Figure101: Inventaire des clients enregistrés dans WAPT

- *Update ID* ;
- *Kbids* ;
- *Published on* ;
- *Installed on* ;
- *Severity on* ;
- *Classification* ;
- *Title* ;
- *Download size* ;

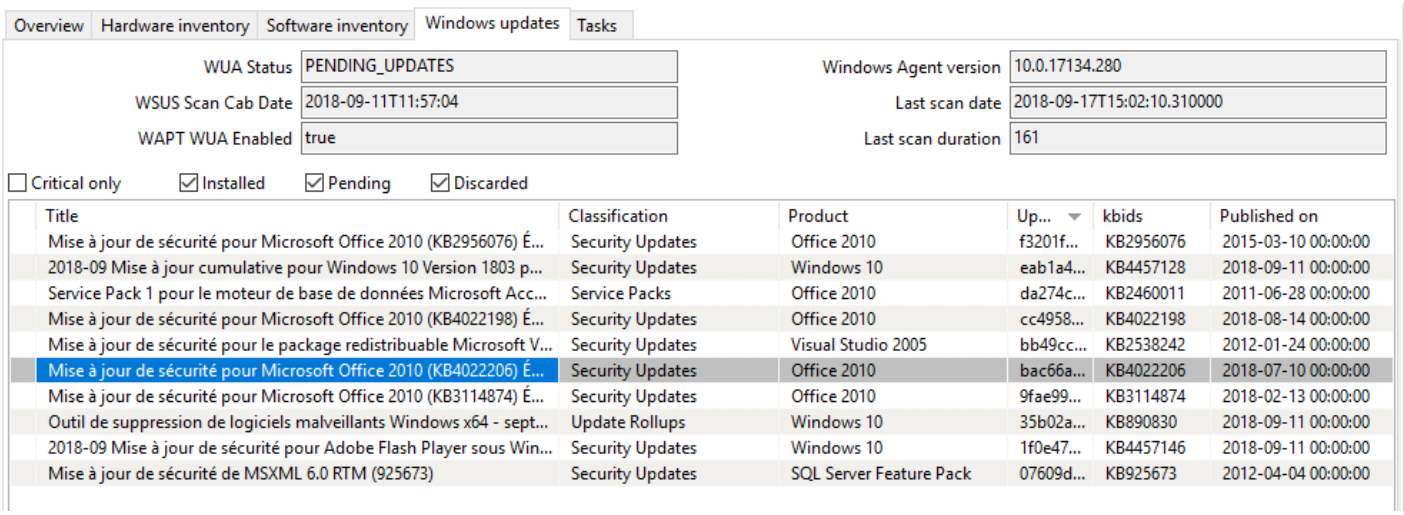


Figure102: Inventaire des mises à jour Windows

Onglet Tâches

Les informations affichées par défaut dans l'onglet *Tâches* sont :

- les tâches en attente ;
- les tâches terminées ;
- les tâches en erreur ;

Rechercher sur l'ensemble des postes

Il est possible de faire une recherche globale sur la totalité des informations décrites précédemment.

Cocher ou décocher les filtres de votre choix.

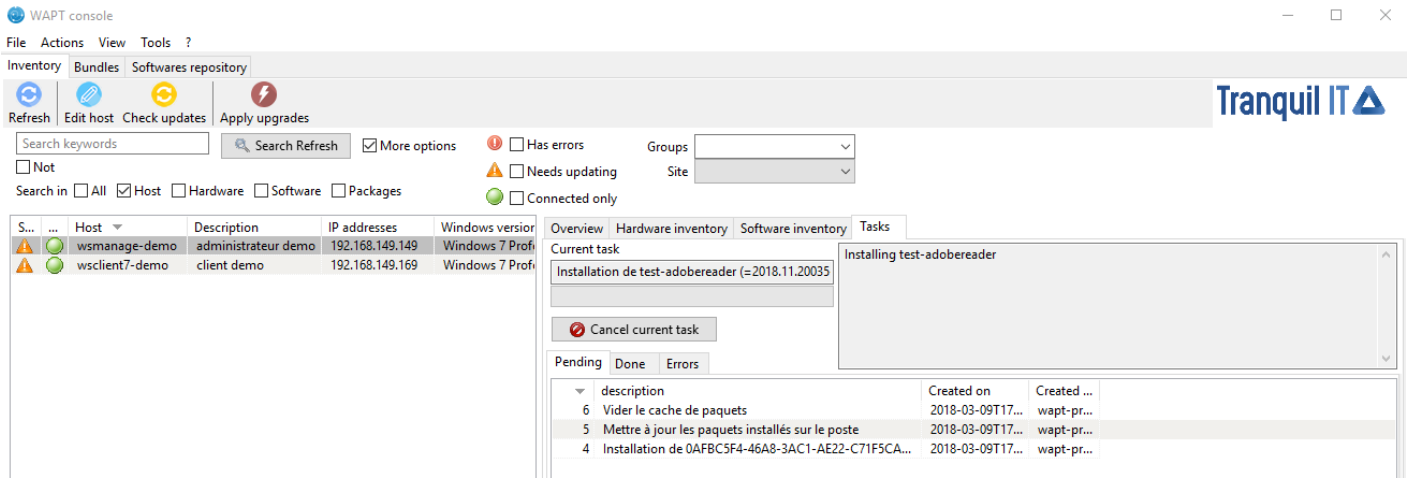


Figure103: Détail des tâches en attente sur la machine

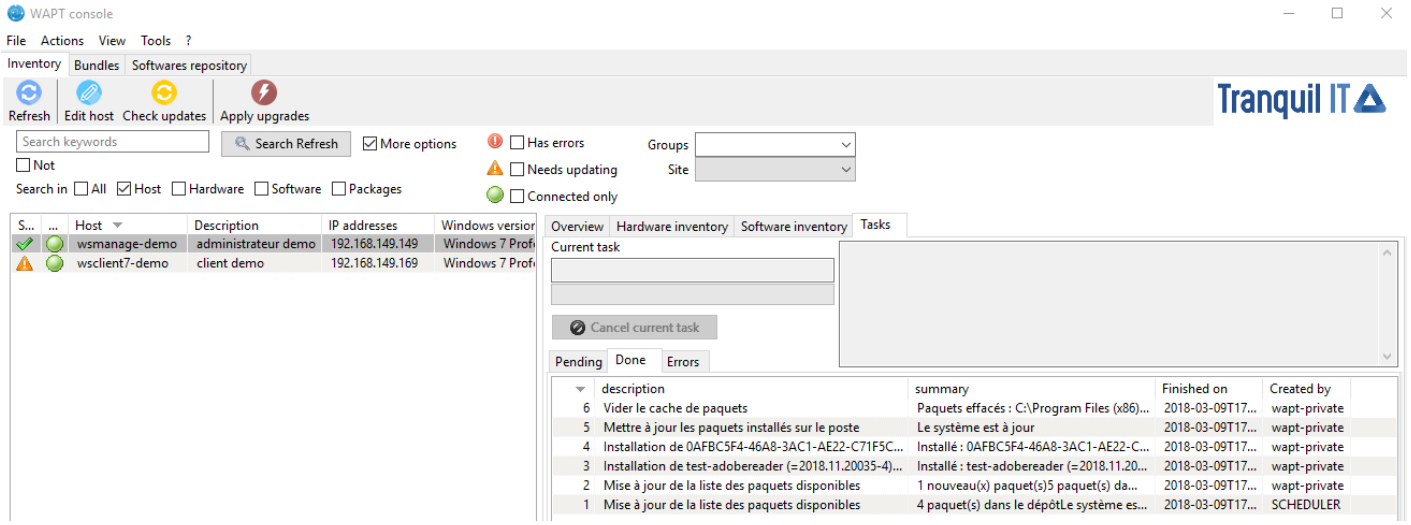


Figure104: Détail des tâches terminées sur la machine

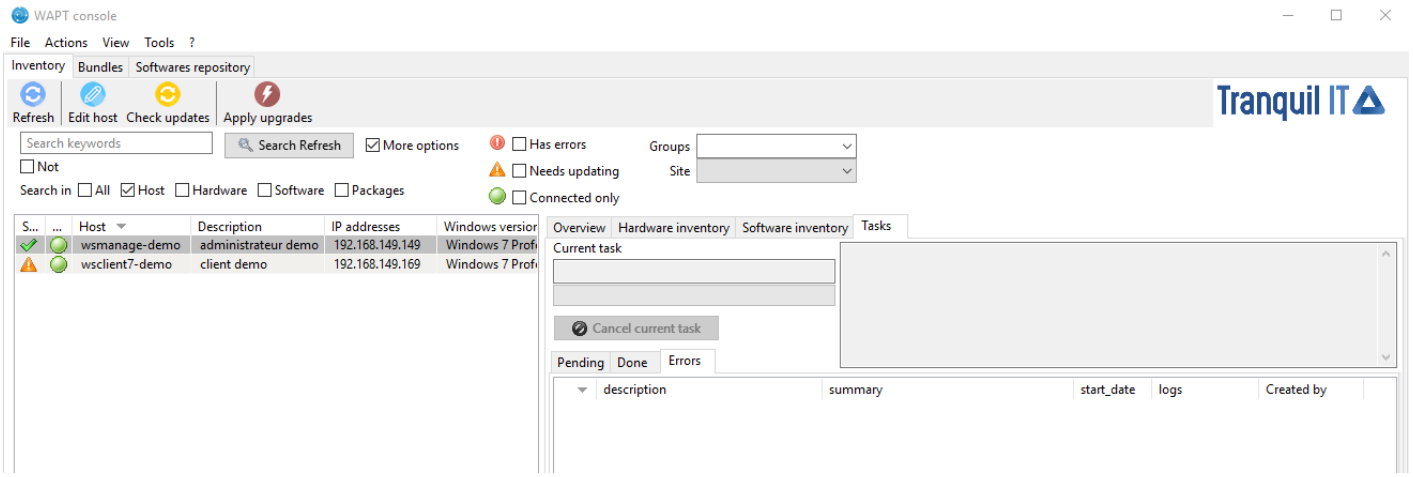


Figure105: Détail des tâches en erreur sur la machine

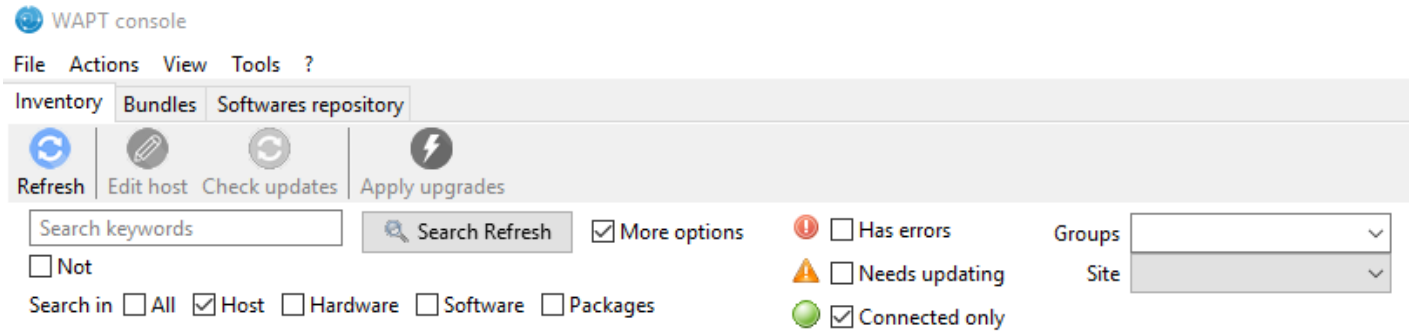


Figure106: Fonctions de recherches avancées de la console WAPT

Table19: Choix des filtres

Choix possibles	Description
Machine	Partie host dans l'onglet *Inventaire matériel* lorsque l'on sélectionne une machine
Matériel	*DMI* dans l'onglet *Inventaire matériel* lorsque l'on sélectionne une machine
Logiciel	Partie *Inventaire logiciel* lorsque l'on clique sur une machine
Paquet	Liste des paquets installés sur les postes sélectionnés
Ont des erreurs	Rechercher uniquement les machine pour lesquelles une tâche ne s'est pas terminée correctement
Nécessite des mise à jour	Rechercher uniquement des machines qui nécessitent une mise à jour
Sélection du groupe	Filtre de recherche sur les membres appartenant à ce paquet *group*

Indication: Les filtres fonctionnent en expression régulière.

Effectuer une recherche à partir d'un paquet

Dans *Dépôt privé* sélectionné le paquet qui vous intéresse puis cliquez sur *Show Host*.

La grille affichera les postes où le paquet est installé. (Le filtrage est uniquement actif sur l'attribut *Package* du paquet sélectionné).

Les différentes colonnes affichent les informations concernant le paquet installé sur la machine (ex : *version du paquet*, *status du paquet*, *status de l'audit*, *date d'installation*, *architecture*).

Vous pouvez également par exemple ajouter les colonnes *Log install* et *Last Audit Output* pour afficher d'un seul coup d'oeil les log d'installation et d'audit.

Créer un groupe de paquets

Les groupes de paquets permettent de créer des listes de paquets à affecter en dépendance à une machine.

Pour créer un groupe de paquets, se rendre dans l'onglet *Groupes de paquets* :

- cliquer sur *Nouveau groupe de paquets* ;
- donner un nom au groupe ;

Indication: Si vous nommez un groupe avec le même nom qu'un groupe de machines dans votre annuaire Active Directory (Microsoft ou Samba-AD), alors les machines du groupe AD seront affectées au groupe WAPT.

- renseigner la description, ajouter des paquets au groupe avec glisser-déposer ou avec un clique-droit sur le nom du paquet, ajouter au groupe ;
- cliquer sur *Enregistrer* pour enregistrer le groupe ;

Indication: Pour désinstaller un paquet, il est possible d'ajouter des paquets interdits au groupe.

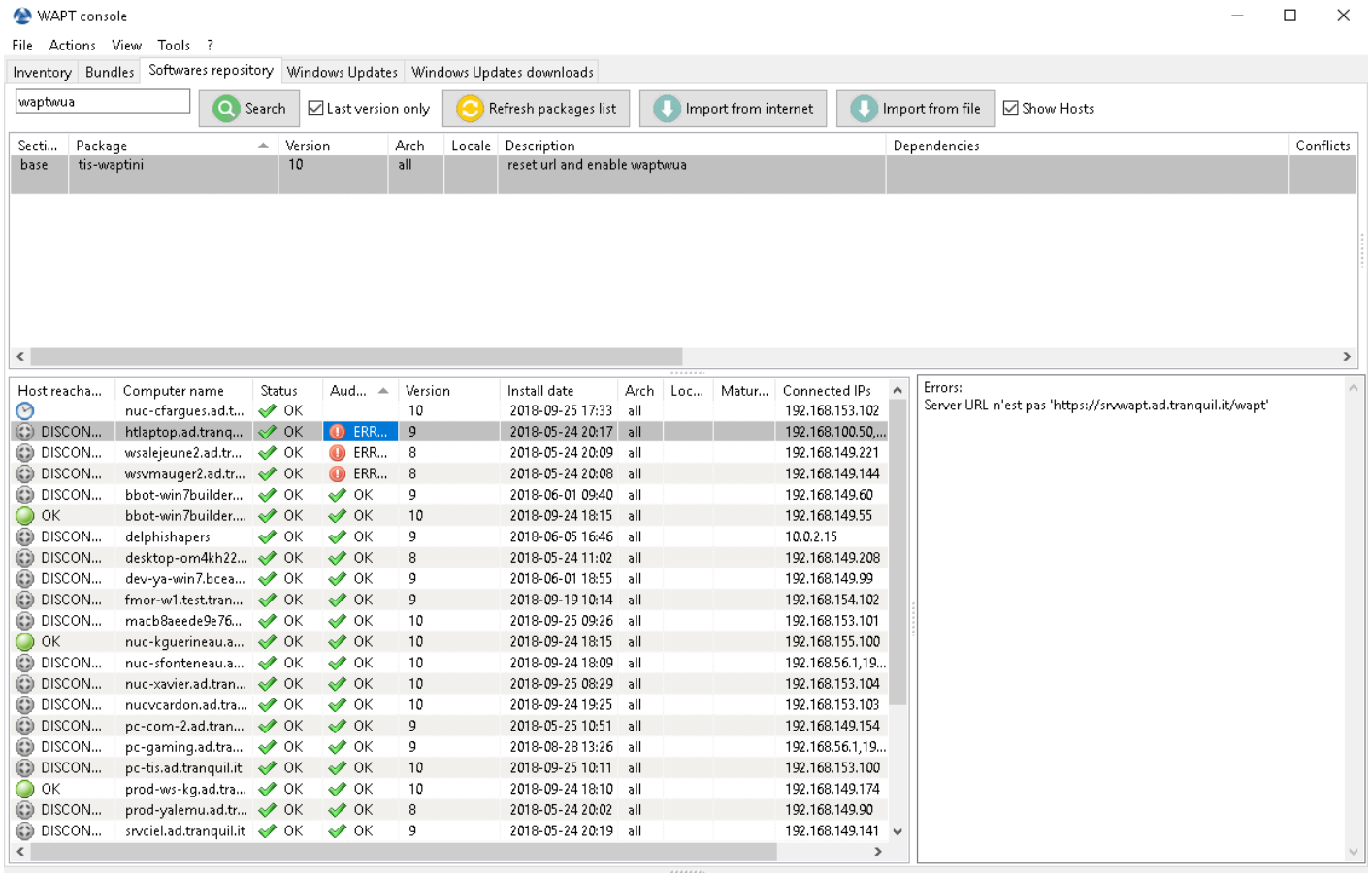


Figure107: Filtrage par paquet

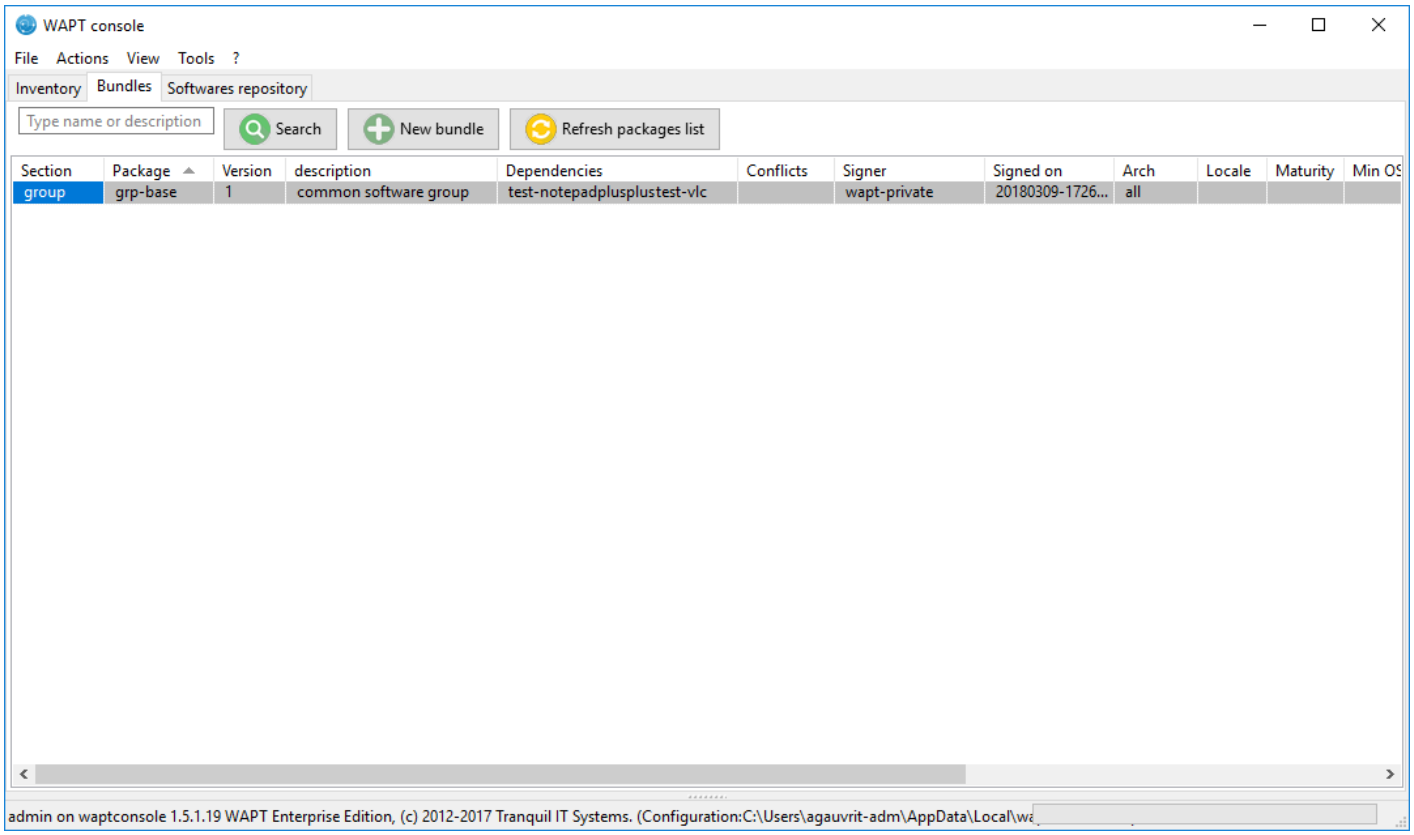


Figure108: Grille des groupes de paquets

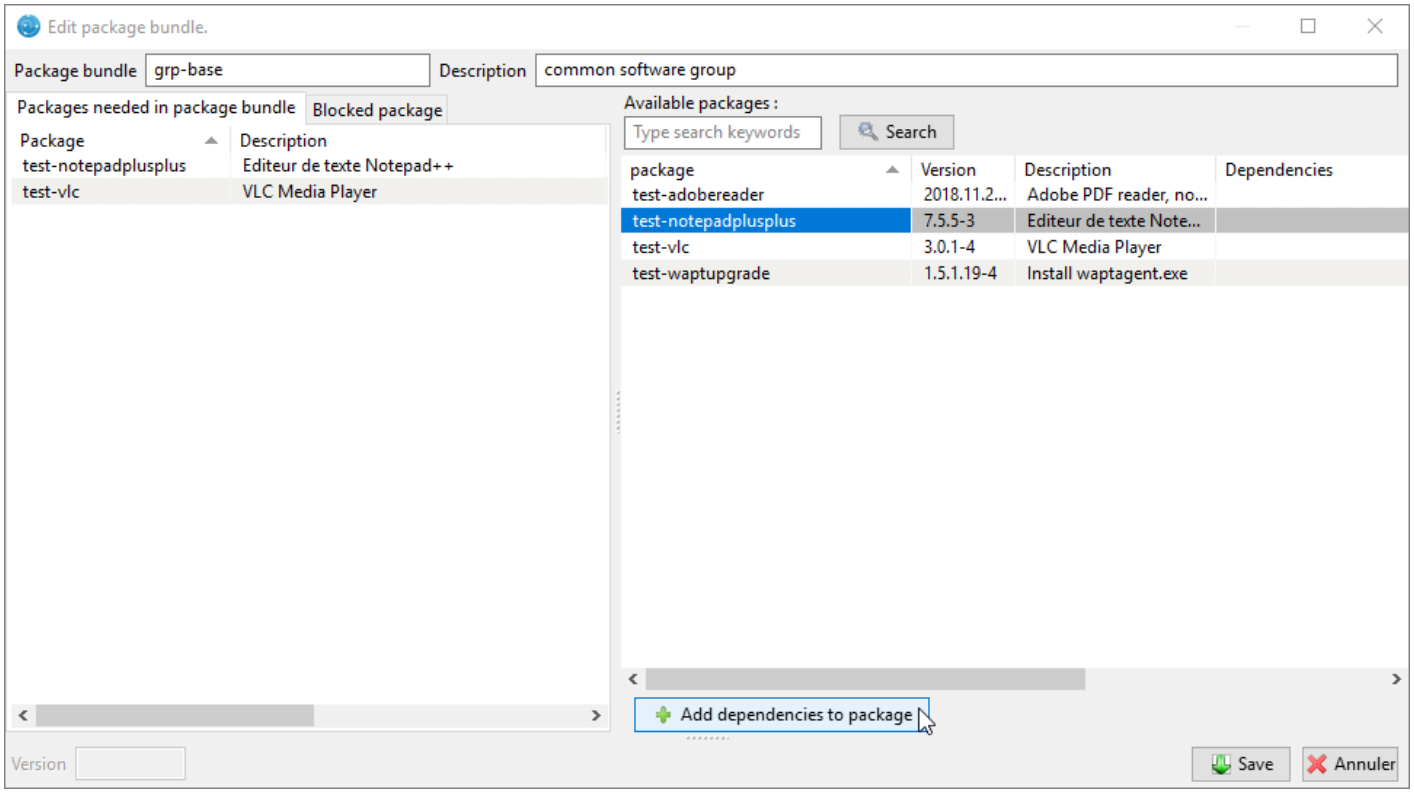


Figure109: Créer un groupe de paquets

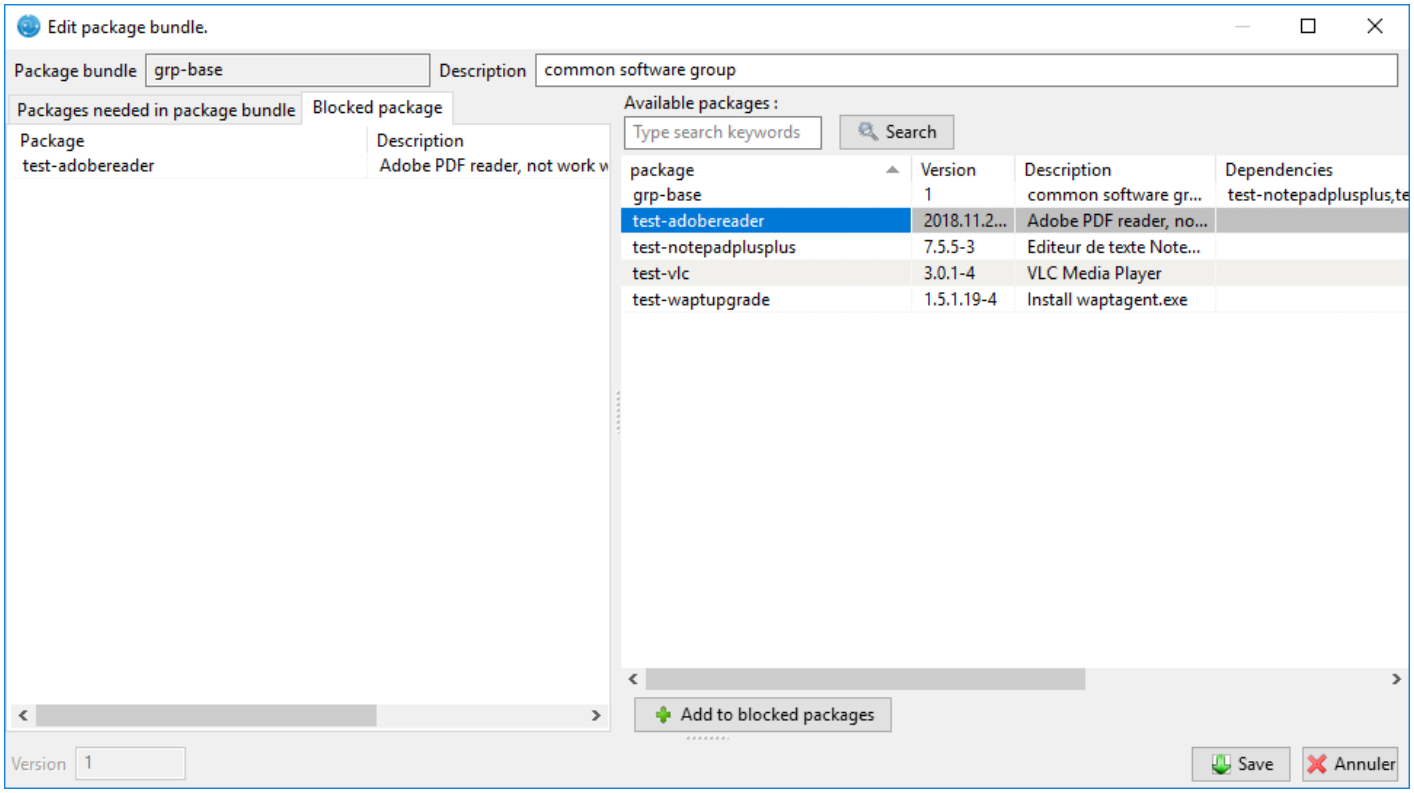


Figure110: Interdire un paquet

Dans *Dépôt privé* apparaît la liste des paquets présents dans le dépôt WAPT local. Par défaut, la console n'affichera que les dernières versions pour chaque paquet.

Pour afficher toutes les versions des paquets, décocher *Dernière version seulement*. Pour supprimer un paquet du dépôt, *Clic-droit → Supprimer*.

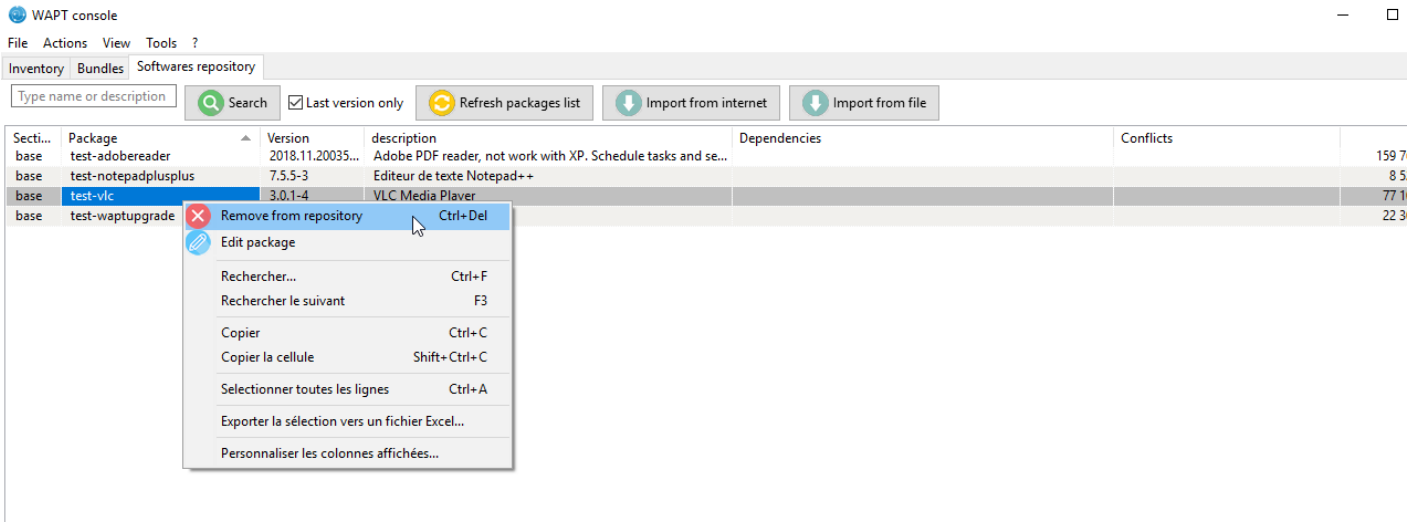


Figure111: Supprimer un paquet

Pour éditer un paquet, *Clic-droit → Editer le paquet*, le paquet sera téléchargé dans le dossier défini avec le paramètre `default_sources_root` de la console.

Modifier le paquet comme désiré, reconstruire le paquet et le renvoyer sur le dépôt. Une fois le paquet uploadé, actualiser la liste des paquets à l'aide du bouton *Actualiser les paquets disponibles* ou de la touche `F5` du clavier.

Une barre de recherche est également disponible pour filtrer les paquets lorsqu'ils sont nombreux.

Indication: With the *section* drop-down menu, you can choose to create a profile package rather than a *group* package.

Nettoyer le cache de la console

Lors de l'import depuis Internet, la console télécharge le paquet dans `%appdata%localwaptconsolecache`.

Pour vider ce cache, depuis la console, cliquer sur *Outils → Nettoyer le cache local*.

Changer le mot de passe du serveur WAPT

Pour changer le mot de passe du serveur WAPT, aller sur *Outils → Changer le mot de passe*, renseigner l'ancien mot de passe et saisir le nouveau.

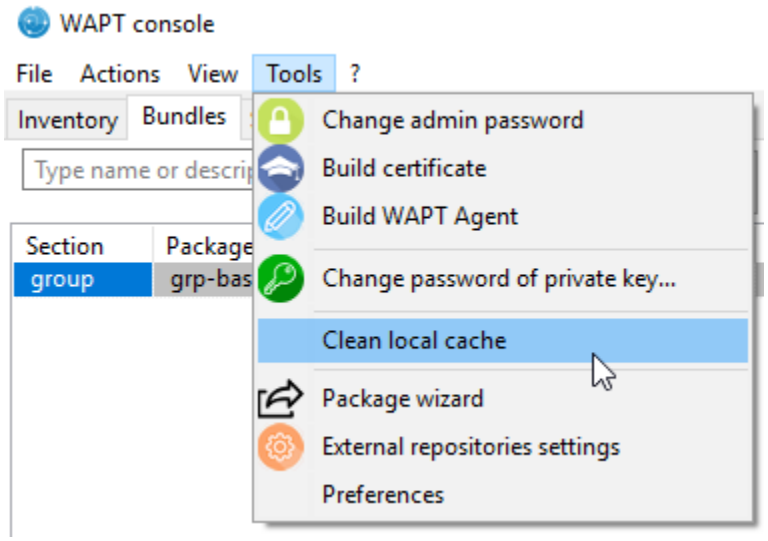


Figure112: Nettoyer le cache local

Modifier la configuration WAPT locale de la console

Pour modifier les paramètres d'affichage de la console : *Affichage* → *Préférences d'affichage*.

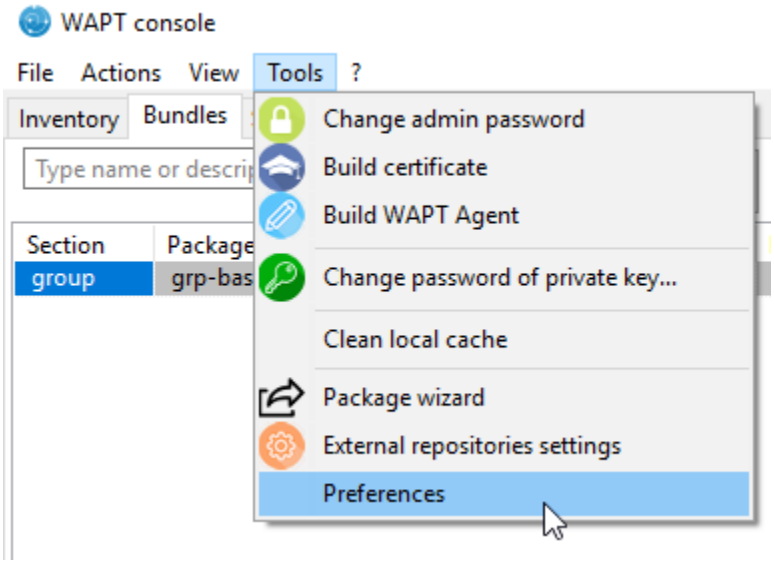


Figure113: Options de configuration de la console WAPT

- onglet *Basique* des options de base ;

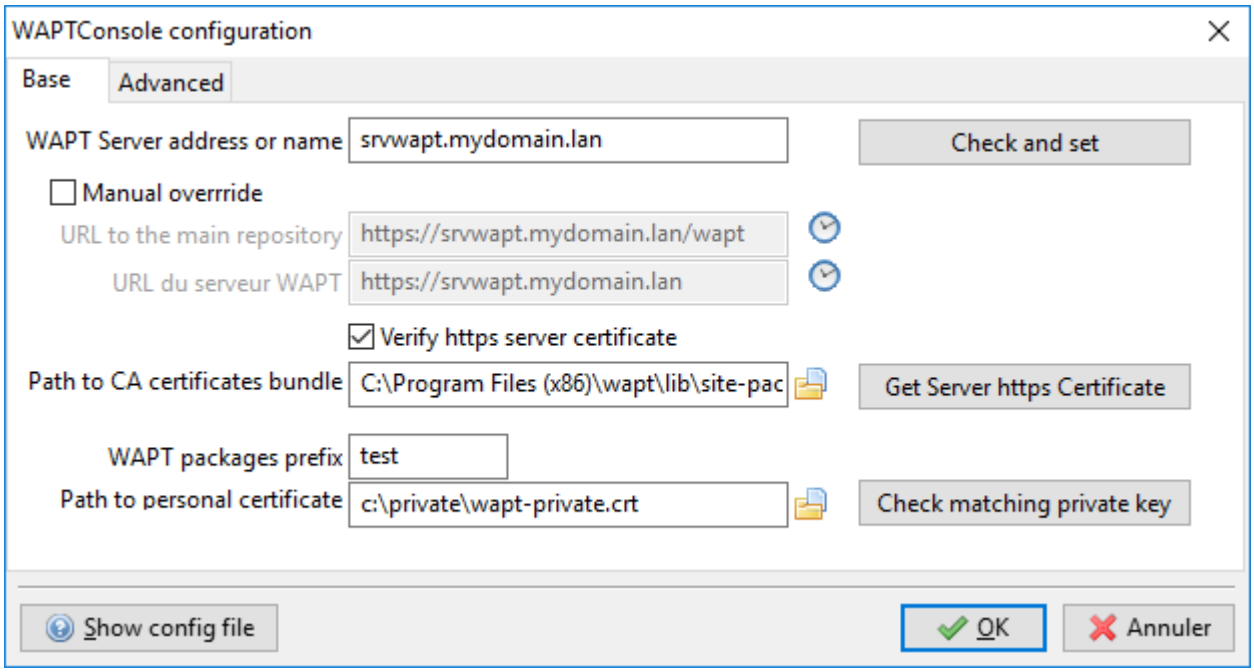


Figure114: Options de configuration de la console WAPT

Paramètres	Description	Exemple
Adresse du serveur WAPT	URL de votre serveur WAPT	srvwapt.mydomain.lan
URL du dépôt principal	URL du dépôt principal (uniquement si spécifier manuellement est coché)	https://srvwapt.mydomain.lan/wapt/
URL de votre serveur WAPT	URL du serveur WAPT (uniquement si spécifier manuellement est coché)	https://srvwapt.mydomain.lan/
Vérification du certificat https	Indique si le certificat https doit être verifié	yes
Chemin vers le bundle de certificats	Chemin vers le bundle de certificats qui permettera la vérification https	Voir *la documentation pour activer la vérification du certificat HTTPS*.
Préfixe pour la création de paquets WAPT	Préfixe donné aux paquets lors de la duplication.	prefix
Chemin du certificat personnel	Chemin vers le certificat associée a la clé privé pour signer les paquets	`C:\private\mykey.crt`

Indication: Le bouton *Récupération du certificat serveur* permet de télécharger le certificat HTTPS du serveur dans `WAPT\ssl\server` et d'indiquer à la console de vérifier la connexion HTTPS avec ce bundle. Vous faites ainsi du *Certificate pinning*. Vous devez en revanche être certain de communiquer avec le bon serveur quand vous cliquez sur le bouton.

- onglet *Avancé* des options avancées ;

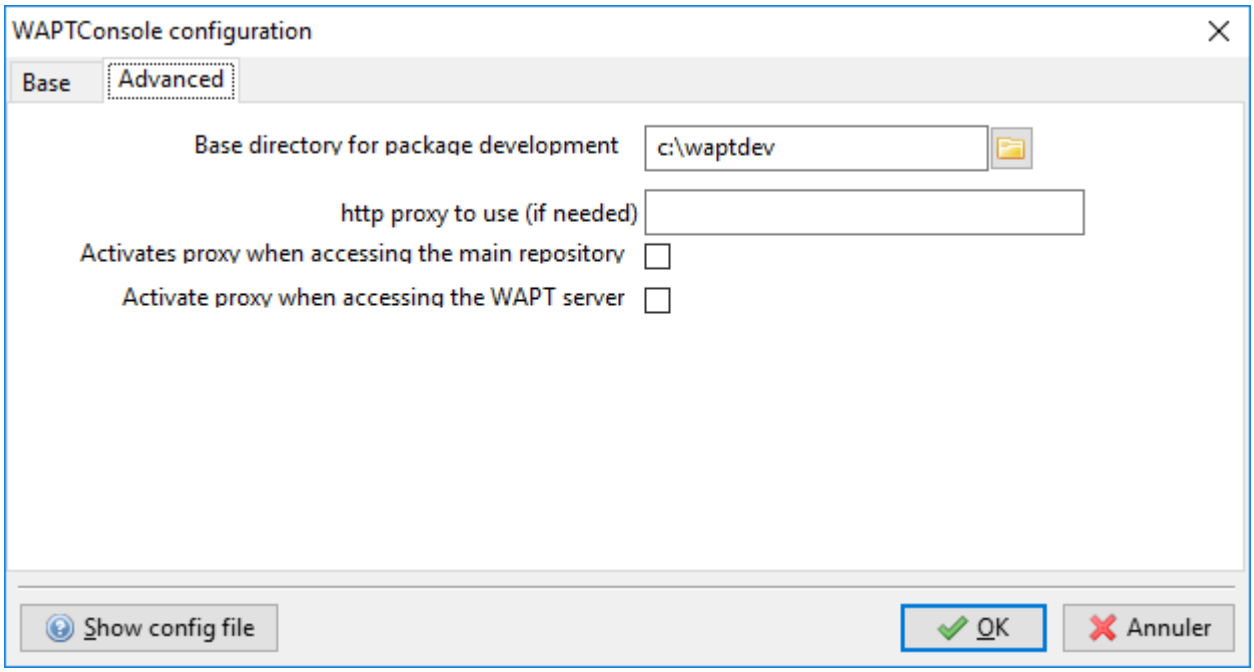


Figure115: Options de configuration de la console WAPT

Table20: :header-rows: 1

Paramètres	Description	Exemple
Répertoire de base pour waptdev	Indique le répertoire pour le stockage des paquets en cours de développement	`C:\waptdev`
Proxy http à utiliser	Indique un proxy a utiliser par l'agent wapt pour contacter le serveur	*http://srvproxy.mydomain.lan:8080*
Activation du proxy	Active le proxy pour le dépôt et / ou le serveur	False

Pour modifier les paramètres d'affichage de la console : *Affichage* → *Préférences d'affichage*.

Table21: :header-rows: 1

Paramètres	Description	Exemple
Nombre maximum de poste affiché	Indique le nombre de poste maximum à afficher. (ce paramétrage permet d'optimiser la console)	2000
Langue	Défnit la langue pour la console WAPT	English
Voir les informations de débug dans la console WAPT	Permet d'afficher les informations de débug dans la console	True
Autoriser les outils externe dans les menus contextuels des hôtes	TODO	True
Activer les fonctionalités d'administration	TODO	True
Cacher les actions non disponibles	TODO	True

Générer un nouveau certificat public

Nouveau dans la version 1.3.12.13.

La génération d'un nouveau certificat public permet d'actualiser le certificat public sans recréer de couple clé privée / certificat public.

Pour cela cliquer sur *Outils* → *Création de certificat*

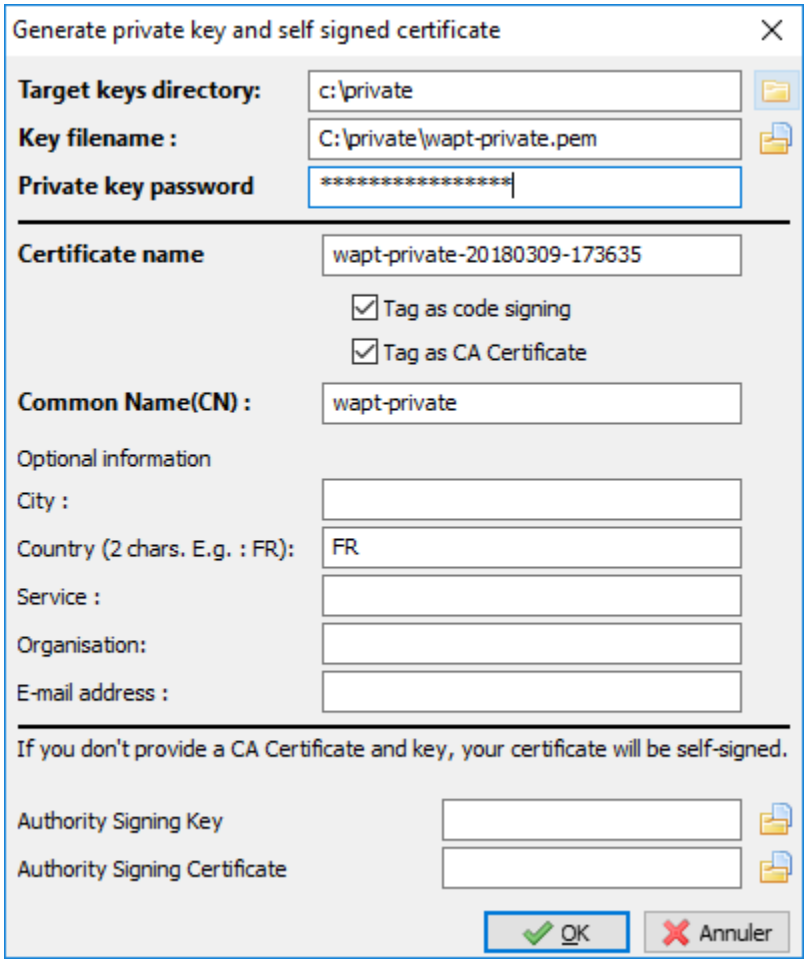


Figure116: Générer un certificat auto-signé

La clé privée est récupérée depuis la configuration existante, changer le `Common Name` puis générer le nouveau certificat.

L'ancien certificat sera écrasé.

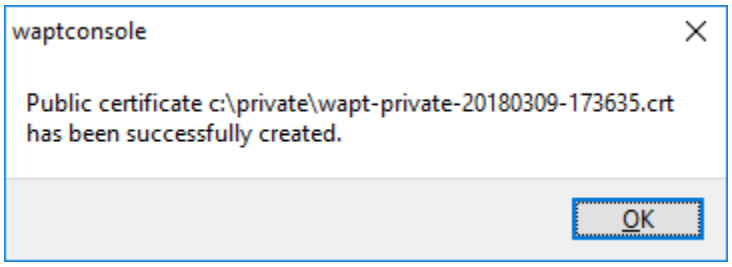
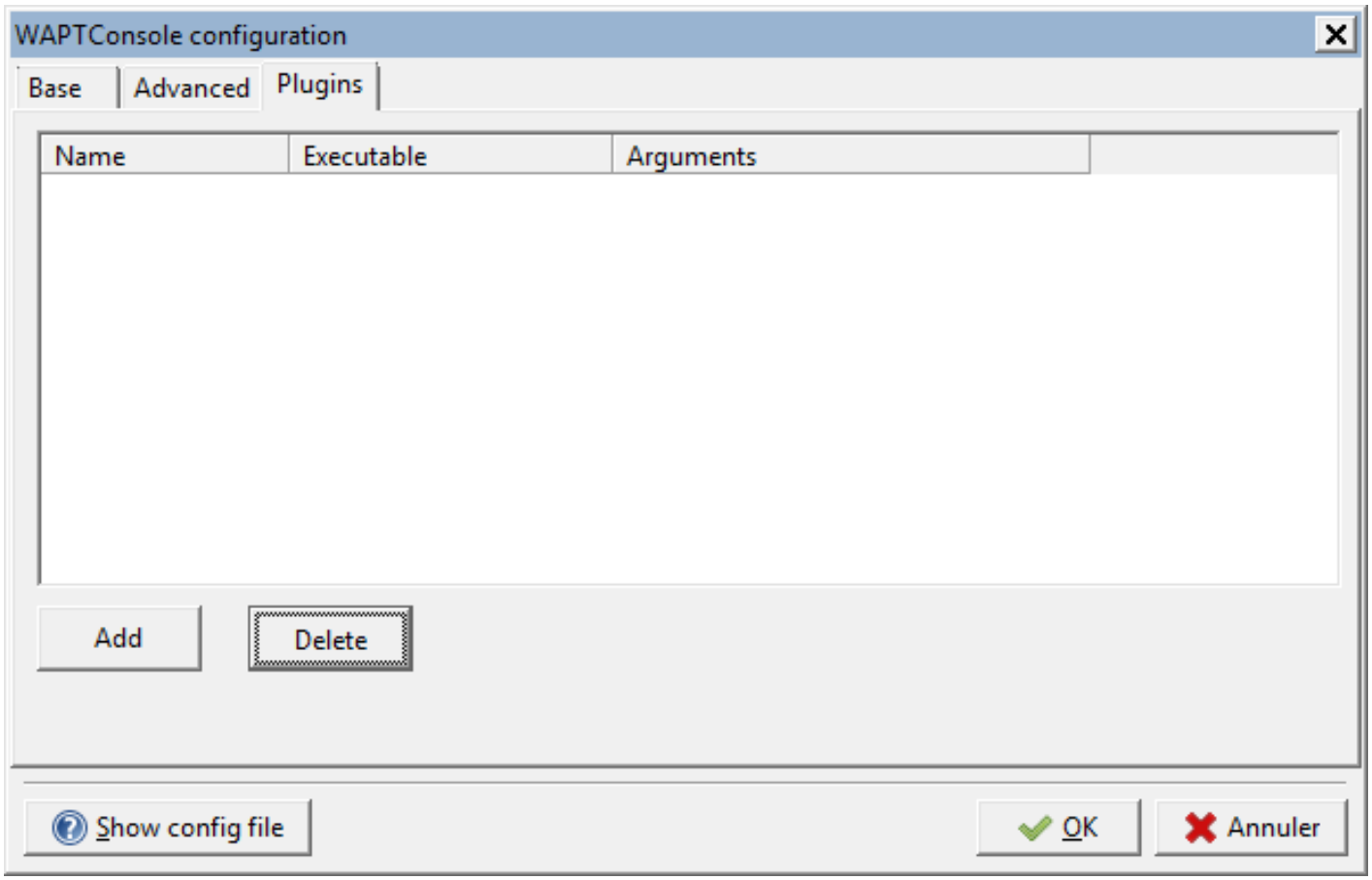


Figure117: Génération du certificat public confirmé

Ajouter des plugins dans la console

Nouveau dans la version 1.7.

Pour cela cliquer sur *Outils* → *Préférences* → *Plugins*.

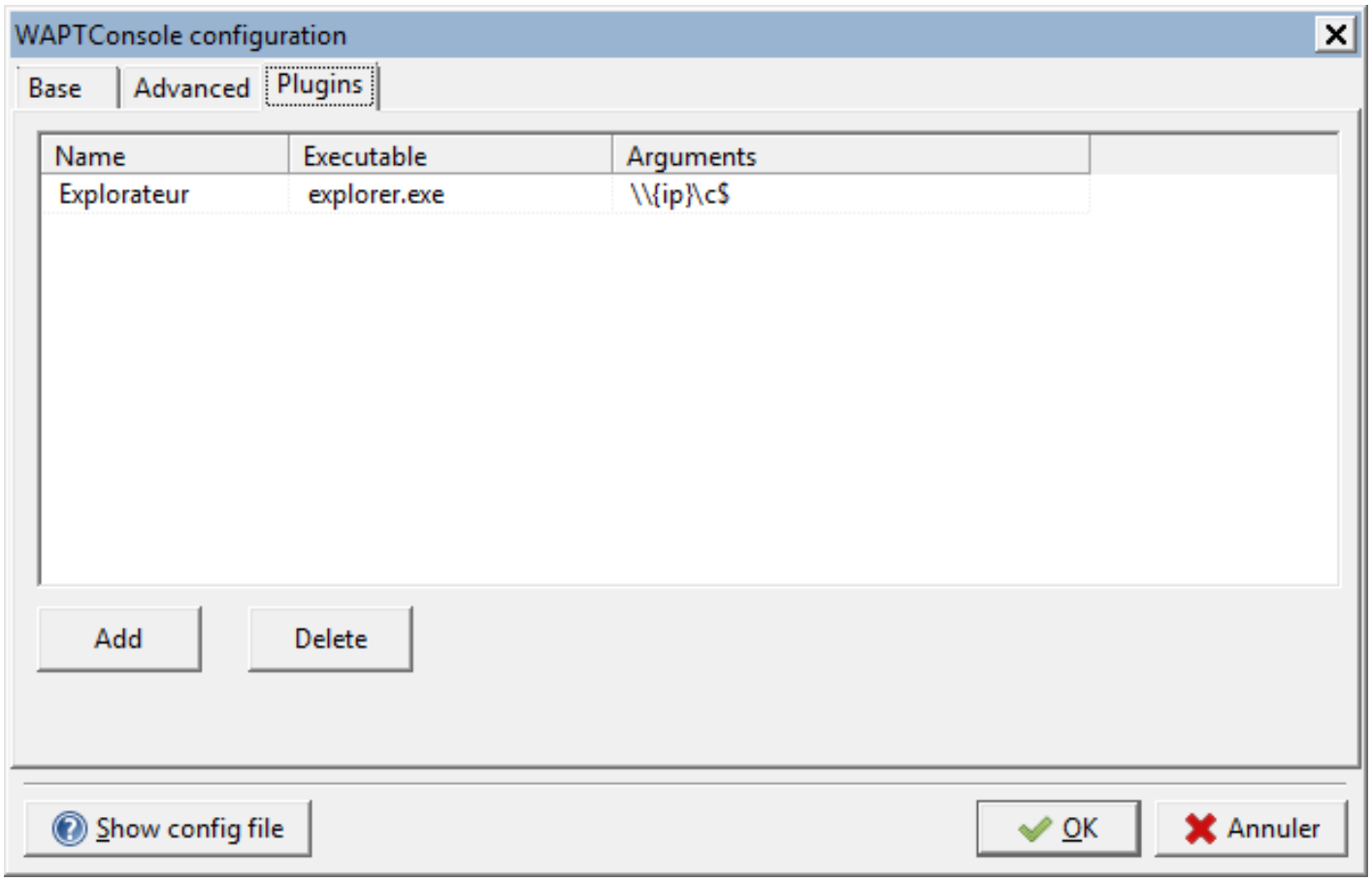


Cliquer sur *Ajouter* pour ajouter des plugins, puis éditer les colonnes correspondantes

Table22: :header-rows: 1

Colonne	Description
Nom	Nom qui apparaîtra dans le menu
Exécutable	Chemin vers l'exécutable qui sera exécuté après le clic
Paramètres	Arguments passés à l'exécutable. Des variables peuvent être employées telles {ip}, {uuid} or {computer_fqdn}

Les plugins apparaîtront alors dans le menu :



6.7.3 Utilisation du WAPTtray.

Le WAPTtray est utilisable en compte utilisateur, il est situé dans le dossier `C:\Program Files (x86)\wapt\wapttray.exe`.

Le WAPTtray s'ouvre au démarrage de la session si l'option a été cochée lors de l'installation. Il se place dans la barre des tâches lors de l'ouverture de session.

On peut aussi lancer le WAPTtray avec une GPO de démarrage exécutant `C:\Program Files (x86)\wapt\wapttray.exe`.

L'icône est pratique pour les utilisateurs qui souhaitent choisir le bon moment pour la mise à jour de leur poste.

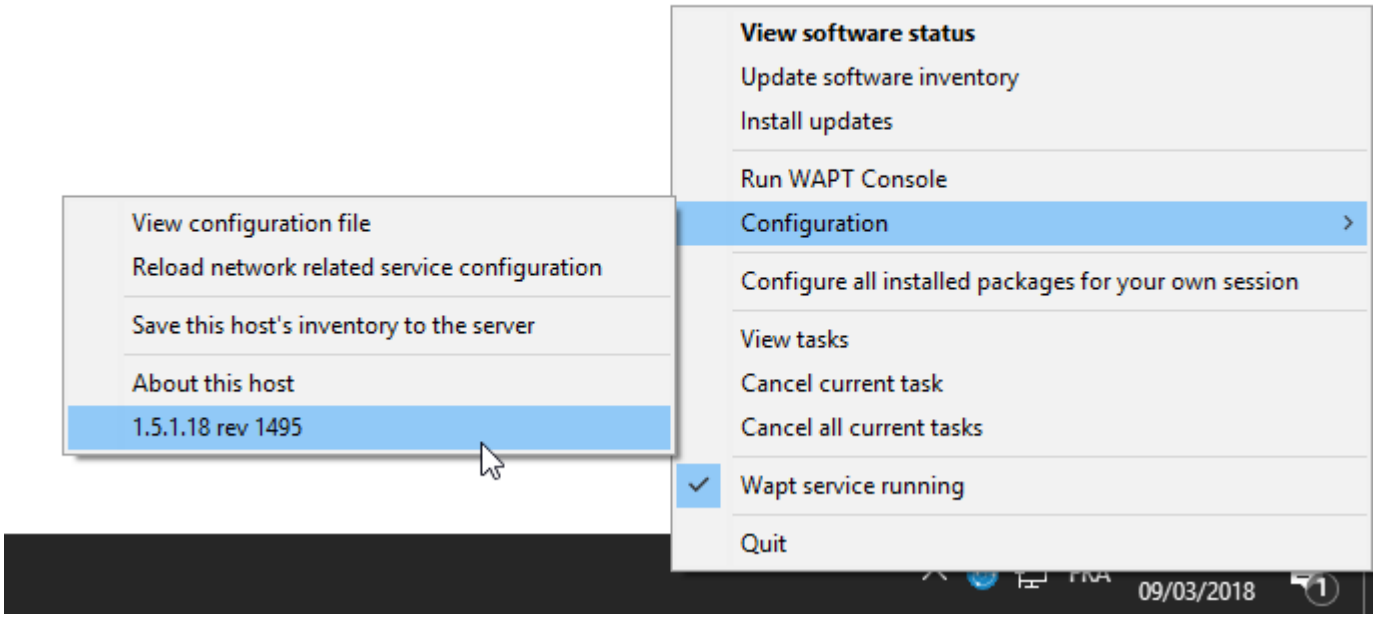


Figure118: WAPTtray dans la barre des tâches

Fonctionnalités de WAPTtray

Table23: Liste des fonctionnalités de WAPTtray

Action	Description
Afficher le statut des logiciels	lance l'interface web locale dans un navigateur
Lancer l'installation d'une mise à jour	lance l'installation des paquets lorsque des mises à jours sont disponibles.
Actualiser la liste des paquets disponibles	rafraîchit la liste des paquets disponibles. Un double-clic sur l'icône produit le même effet.
Lancer la console	lance la console d'administration WAPT.
Afficher le fichier de configuration	lance l'édition du fichier `C:\Program Files (x86)\wapt\wapt-get.ini` en mode *Administrateur*.
Recharger la configuration réseau du service	recharge la connexion au serveur WAPT en cas de reconfiguration du réseau
Forcer la remontée d'inventaire vers le serveur	force la remontée d'information d'inventaire au serveur
Configurer tous les paquets installés pour l'Utilisateur	lance un **`session-setup`** pour configurer l'environnement Utilisateur pour tous les paquets installés sur la machine.
Annuler les tâches WAPT en cours d'exécution sur la machine	affiche les tâches en cours d'exécution, permet d'annuler une tâche en cours d'exécution, permet d'annuler toutes les tâches en cours d'exécution
Arrêter et relancer le service service WAPT	arrête et lance le service `WAPTService`
Quitter WAPTtray	ferme l'*icône WAPTtray* sans couper le service WAPT local

6.7.4 Utiliser WAPT en ligne de commande

L'agent WAPT installé permet l'utilisation de WAPT en ligne de commande à l'aide du programme **wapt-get**.

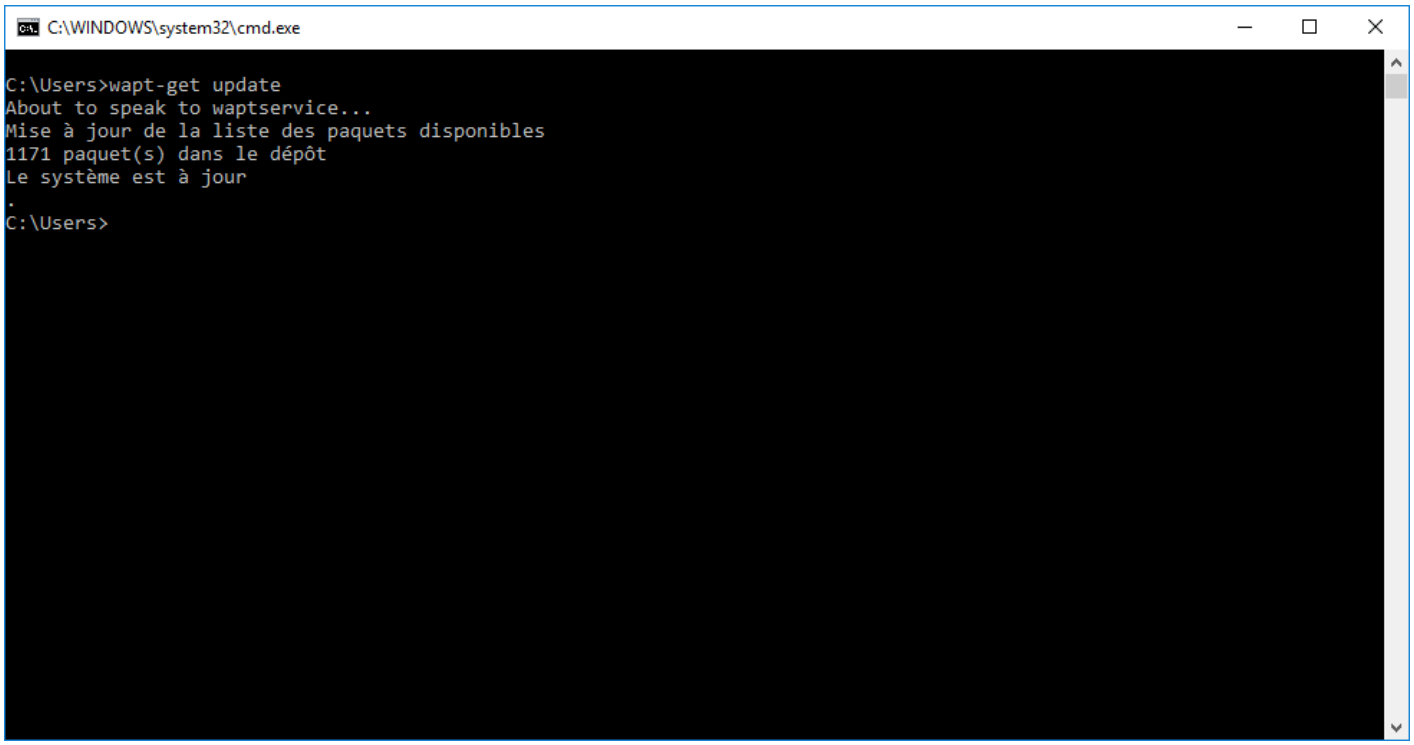


Figure119: Invite de commande Windows

Note:

- par défaut la ligne de commande va essayer d'effectuer les actions demandées avec le compte qui a lancé la console **cmd.exe** ;
- si l'*Utilisateur* n'est pas un *Administrateur Local* ou si le **cmd.exe** n'a pas été lancé avec les droits *Administrateur Local*, la commande sera passée au **waptservice** ;
- par sécurité, certaines actions demandent un identifiant et un mot de passe ;
- seuls les membres du groupe *Administrateur Local* de la machine et les membres du groupe de sécurité *waptselfservice* de l'Active Directory sont autorisés ;
- pour forcer le passage par le service WAPT en tant qu'*Administrateur Local*, ajouter simplement *-S* après **wapt-get** ;

Utiliser les fonctions les plus courantes de la ligne de commande

wapt-get update

La commande **update** permet de mettre à jour la liste des paquets disponibles.

L'agent WAPT local télécharge le fichier `Packages` depuis le dépôt et le compare avec sa base de données.

- si des mises à jour sont disponibles, l'agent bascule alors les paquets concernés dans l'état **TO-UPDATE** ;
- si de nouveaux logiciels ont été ajoutés sur le dépôt, ils deviennent téléchargeables sur le poste client ;

Note: L'action **update** ne télécharge pas les paquets, elle met uniquement à jour la base locale des paquets.

La commande `wapt-get update` renvoie :

```
Update package list
Total packages: 751
Added packages:

Removed packages:

Upgradable packages:
upgrade
additional
install
remove
Repositories URL:
https://srvwapt.mydomain.lan/wapt
https://srvwapt.mydomain.lan/wapt-host
```

wapt-get upgrade

La commande **upgrade** lance l'installation des paquets en attente de mise à jour ou d'installation.

L'agent WAPT local télécharge si nécessaire les paquets WAPT dans son cache local puis installe les paquets.

Indication: Il est recommandé de lancer un **update** avant de lancer un **upgrade** ;

Sans un **update** préalable, l'agent WAPT n'installera rien ;

La commande `wapt-get upgrade` renvoie :

```
Installing tis-mumble
Shutting down Mumble
installing Mumble 1.2.8
Installing w7demo.domain.lan

=== install packages ===
w7demo.domain.lan (=3) | w7demo.domain.lan (3)
```

(suite sur la page suivante)

(suite de la page précédente)

```
=== additional packages ===
tis-mumble                    | tis-mumble (1.2.8-1)
```

wapt-get search

La commande **`search`** permet de rechercher un ou des paquet(s) sur les dépôt enregistrés.

La recherche prend un argument en paramètre. Cet argument servira à chercher dans le nom du paquet et dans sa description.

La commande `wapt-get install tis-firefox` renvoie :

Package name	Version	Plateform	Description
tis-firefox	50.0.2-73	all	Mozilla Firefox Web Browser in French
tis-firefox-en	50.0.1-58	all	Mozilla Firefox Web Browser in English
tis-firefox-esr	45.6.0-4	all	Mozilla Firefox Web Browser ESR
tis-flashplayer	24.0.0.186-1	all	Adobe Flashplayer for Firefox

wapt-get install

La commande **`install`** lance l'installation d'un paquet.

La commande prend un argument en paramètre. Cet argument est le nom du paquet contenant le préfixe.

Pour l'installation de Mozilla Firefox il faudra taper la commande `wapt-get install <prefix>-firefox`.

Note: Si le paquet n'est pas téléchargé au préalable, la commande **`install`** téléchargera préalablement le paquet en cache.

Attention: L'installation d'un paquet WAPT avec **`install`** n'ajoute pas le paquet en dépendance au poste.

Il est installé sur la machine, mais en cas de réinstallation du poste il ne sera pas réinstallé automatiquement.

La commande `wapt-get install tis-firefox` renvoie :

```
installing WAPT packages tis-firefox
Installing tis-firefox.local/wapt/tis-firefox_50.0.2-73_all.wapt: 44796043 / 44796043 (100%)␣
↪(33651 KB/s)
Firefox Setup 50.0.2.exe successfully installed.
Disabling auto update
Disabling profile migration from ie
Override User UI

=== install packages ===
tis-firefox                   | tis-firefox (50.0.2-73)
```

wapt-get remove

La commande **remove** lance la désinstallation d'un paquet.

La commande prend un argument en paramètre. Cet argument est le nom du paquet contenant le préfixe.

Pour désinstaller Mozilla Firefox il faudra taper la commande : **wapt-get remove <prefix>-firefox**.

Attention: La désinstallation d'un paquet WAPT avec **remove** ne supprime pas la dépendance entre le paquet machine et le paquet.

Le paquet sera effectivement désinstallé de la machine, mais il sera réinstallé automatiquement lors du prochain **upgrade**.

Afin de supprimer totalement un paquet d'un poste il faut faire un **remove** puis éditer la configuration de la machine via la console et supprimer la dépendance au paquet.

La commande `wapt-get remove tis-firefox` renvoie :

```
Removing tis-firefox ...

=== Removed packages ===
  tis-firefox
```

wapt-get clean

C:\Program Files (x86)\wapt\

La commande **clean** est lancée à chaque fin de mise à jour pour économiser le stockage sur le disque dur de la machine.

La commande `wapt-get clean` renvoie :

```
Removed files:
C:\Program Files (x86)\wapt\cache\tis-mumble_1.2.8-1_all.wapt
C:\Program Files (x86)\\wapt\cache\tis-vlc_2.2.4-2_all.wapt
```

Utiliser les lignes de commande spéciales

wapt-get register

La commande **wapt-get register <description>** permet de remonter l'inventaire matériel et logiciel de la machine sur le serveur d'inventaire WAPT.

Indication: Vous pouvez préciser une description supplémentaire en paramètre de **register**, cette description sera affichée dans la console WAPT comme description du poste.

Vous pouvez donc profiter de WAPT pour améliorer la gestion administrative de votre parc en affectant à la machine le nom d'un service ou le nom du principal utilisateur de la machine.

La commande **wapt-get register "PC de Thomas** ne renvoie rien ;

wapt-get download

La commande **wapt-get download <nom du paquet>** télécharge le paquet WAPT dans le cache local situé à `C:\Program Files\wapt\cache`.

La commande **wapt-get download tis-7zip** renvoie :

```
Downloading packages tis-7zip (=16.4-8)

Downloaded packages:
  C:\Program Files (x86)\wapt\cache\tis-7zip_16.4-8_all_all.wapt
```

wapt-get download-upgrade

La commande **wapt-get download-upgrade** télécharge les paquets à mettre à jour dans le cache WAPT local `C:\Program Files (x86)\wapt\cache`.

La commande **wapt-get download-upgrade** renvoie :

```
=== downloaded packages ===
C:\Program Files (x86)\wapt\cache\tis-firebird_2.5.5.26952-1_all.wapt
```

wapt-get show

La commande **wapt-get show <nom du paquet>** affiche les informations contenues dans l'index `Packages`.

Si plusieurs versions du paquet sont disponibles sur le dépôt, toutes les versions sont affichées.

La commande **wapt-get show tis-firebird** renvoie :

```
Display package control data for tis-firebird

package           : tis-firebird
version           : 2.5.5.26952-1
architecture      : all
section           : base
priority          : optional
maintainer        : Hubert TOUVET
description       : Firebird database SQL superserver with admin tools (Firebird Project)
filename          : tis-firebird_2.5.5.26952-1_all.wapt
size              : 7012970
repo_url          : https://srvwapt.mydomain.lan/wapt
md5sum            : 6f6d70630674f5d58a5259b1e6752221
repo              : global
```

wapt-get list

La commande **wapt-get list** affiche la liste des paquets WAPT actuellement installés sur la machine.

La commande **wapt-get list** renvoie :

package	version	install status	install_date	description
tis-7zip	16.4-8	OK	2016-12-01T17:43	7-zip compression and archiving software for x86 and x64
tis-brackets	1.8-1	OK	2016-12-01T17:44	Brackets is a lightweight
tis-ccleaner	5.23.5808-0	OK	2016-12-01T18:55	the right choice utility to quickly clean up, repair and optimize Windows
tis-rsat-win7x64	2	OK	2016-12-02T10:46	package for MS RSAT Remote server admin windows6.1-kb958830-x64 pour Win7 SP1
tis-rsat-x64	1	OK	2016-12-02T10:51	package for MS RSAT Remote server admin windows6.1-kb958830-x64 pour Win7 SP1
tis-dotnetfx4.6	4.6.2-1	OK	2016-12-09T16:05	dot net FX 4.6.2 Framework CLient. Replaces 4/4.5/4.5.1/4.5.2/4.6/4.6.1

wapt-get upgradedb

La commande **wapt-get upgradedb** met à jour le schéma de la base de données locale WAPT si nécessaire.

La commande **wapt-get upgradedb** renvoie :

```
WARNING upgrade db aborted: current structure version 20161109 is newer or equal to requested␣
↪structure version 20161109
No database upgrade required, current 20161109, required 20161109
```

wapt-get setup-tasks - wapt-get enable-tasks - wapt-get disable-tasks

La commande **wapt-get setup-tasks** ajoute des tâches planifiées **update** et **upgrade**

Indication: Cette fonction est utile lorsque que l'on ne souhaite pas utiliser le service **waptservice**, sinon le service **waptservice** s'en chargera.

Pour que cela fonctionne, il faut au préalable avoir configuré les paramètres dans le fichier `wapt-get.ini` :

- *waptupdate_task_maxruntime*;
- *waptupgrade_task_maxruntime*;
- *waptupdate_task_period*;
- *waptupgrade_task_period*;

Ensuite :

- la commande **wapt-get enable-tasks** activera les tâches planifiées ;
- la commande **wapt-get disable-tasks** désactivera les tâches planifiées ;

wapt-get add-upgrade-shutdown - wapt-get remove-upgrade-shutdown

- la commande **wapt-get add-upgrade-shutdown** ajoute la stratégie locale **waptexit** des scripts d'arrêt du système pour **waptexit** ;
- la commande **wapt-get remove-upgrade-shutdown** supprime la stratégie locale des scripts d'arrêt du système pour **waptexit** ;

wapt-get inventory

La commande **wapt-get inventory** affiche le contenu de l'inventaire local de la machine au format JSON.

La commande **wapt-get inventory** renvoie :

```
{
  "wapt": {
    "setuphelpers-version": "1.3.8",
    "waptserver": {
      "dnsdomain": "mydomain.lan",
      "proxies": {
        "http": null,
        "https": null
      },
      "server_url": "https://srvwapt.mydomain.lan"
},
...
```

wapt-get update-status

La commande **wapt-get update-status** renvoie l'inventaire local vers le serveur d'inventaire WAPT.

Note: Si un composant matériel a changé sur la machine, cette commande ne remontera pas la nouvelle information au serveur WAPT.

Dans ce cas, il faudra lancer la commande **inventory**.

La commande **wapt-get update-status** renvoie :

```
Inventory correctly sent to server https://srvwapt.mydomain.lan.
```

wapt-get setlocalpassword

La commande **wapt-get setlocalpassword** permet de définir un mot de passe local pour installer des paquets WAPT.

La commande **wapt-get setlocalpassword** renvoie :

```
Local password:
Confirm password:
Local auth password set successfully
```

wapt-get reset-uuid

La commande **wapt-get reset-uuid** permet de récupérer le *UUID* de la machine à partir du BIOS et le remonter au serveur WAPT.

La commande **wapt-get reset-uuid** renvoie :

```
New UUID: B0F23D44-86CB-CEFE-A8D6-FB8E3343FE7F
```

wapt-get generate-uuid

La commande **wapt-get generate-uuid** permet de générer un *UUID* aléatoire pour la machine et de le remonter au serveur WAPT.

Indication: Certaines machines ont un BIOS avec des *UUID* identiques. Cette anomalie est un défaut de paramétrage du BIOS par le constructeur de la machine car les *UUID* devraient être uniques.

La commande command:*generate-uuid* existe pour résoudre ce problème.

La commande **wapt-get generate-uuid** renvoie :

```
New UUID: 6640f174-de90-4b00-86f7-d7834ceb45bc
```

wapt-get get-server-certificate

La commande **wapt-get get-server-certificate** permet de télécharger le certificat SSL du serveur WAPT, quand on utilise HTTPS pour les échanges avec le serveur.

C:\Program Files (x86)\wapt\

La commande **wapt-get get-server-certificate** renvoie :

```
Server certificate written to C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt
```

wapt-get enable-check-certificate

La commande **wapt-get enable-check-certificate** permet de télécharger le certificat SSL du serveur WAPT et active le contrôle des échanges avec le serveur.

La commande **wapt-get enable-check-certificate** renvoie :

```
Server certificate written to C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt
wapt config file updated
```

wapt-get session-setup

La commande **wapt-get session-setup** lance l'exécution des personnalisations logicielles en contexte utilisateur.

Indication: Les instructions en **session-setup** sont définies dans le fichier `setup.py` des paquets applicatifs.

L'ensemble des instructions est sauvegardé en base locale.

Le script **session-setup** est lancé à chaque démarrage mais l'exécution d'un **session-setup** ne se fait qu'une fois par utilisateur et par version de paquet.

Note: Le paramètre *ALL* lancera le **session-setup** de tous les paquets WAPT installés sur la machine.

La commande **wapt-get session-setup ALL** renvoie :

```
Configuring tis-7zip ... No session-setup. Done
Configuring tis-ccleaner ... Already installed. Done
Configuring tis-vlc ... No session-setup. Done
Configuring mdl-tightvnc ... No session-setup. Done
Configuring tis-brackets ... No session-setup. Done
Configuring mdl-firefox-esr ... No session-setup. Done
Configuring tis-rsat-x64 ... No session-setup. Done
Configuring tis-dotnetfx4.6 ... No session-setup. Done
Configuring tis-rsat-win7x64 ... No session-setup. Done
Configuring tis-mumble ... No session-setup. Done
Configuring tis-paint.net ... No session-setup. Done
Configuring wsagauvrit.domain.lan ... No session-setup. Done
```

Utiliser la ligne de commande pour créer des paquets WAPT

wapt-get make-template

La commande `wapt-get make-template <msi or exe file> <package name>` permet de générer un modèle de paquet logiciel à partir d'un installeur exécutable au format MSI ou EXE.

Vous trouverez ici la procédure complète pour *créer des paquets WAPT*.

Indication:

- Si vous avez au préalable installé le paquet `tis-waptdev` sur votre machine de développement, l'éditeur **`PyScripter`** se lancera automatiquement en ouvrant le projet de création de paquet.

La commande `wapt-get make-template C:\Users\User\Downloads\tightvnc-2.8.5-gpl-setup-64bit.msi tis-tightvnc` renvoie :

```
Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\tis-tightvnc-wapt
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\tis-tightvnc-wapt
```

wapt-get make-host-template

La commande `wapt-get make-host-template <host FQDN>` permet de générer un modèle de paquet machine vide.

La commande `wapt-get make-host-template host01.mydomain.lan` renvoie :

```
Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\host01.mydomain.lan-wapt
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\host01.mydomain.lan-wapt
```

wapt-get make-group-template

La commande `wapt-get make-group-template <name of group>` permet de générer un modèle de paquet groupe vide.

La commande `wapt-get make-group-template accounting` renvoie :

```
Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\accounting-wapt
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\accounting-wapt
```

wapt-get list-registry

La commande `wapt-get list-registry <keyword>` permet de rechercher un mot clé parmi les logiciels installés sur la machine.

Cette commande affiche un tableau contenant la clé de désinstallation de chaque logiciel correspondant au critère de recherche.

La commande `wapt-get list-registry firefox` renvoie :

```
UninstallKey                          Software                                Version
↪ Uninstallstring
-------------------------------------------------------------------------------------------
↪-----------------------------------------------------
Mozilla Firefox 45.5.0 ESR (x64 fr)   Mozilla Firefox 45.5.0 ESR (x64 fr)   45.5.0
↪ "C:\Program Files\Mozilla Firefox\uninstall\helper.exe"
```

wapt-get sources

La commande `wapt-get sources <package name>` permet de télécharger les sources depuis un dépôt versionné type Git / SVN.

La commande `wapt-get sources tis-firefox` ne renvoie rien ;

wapt-get build-package

La commande `wapt-get build-package <path to the package>` permet de construire le paquet WAPT et le signer avec la clé privée de l'*Administrateur*.

Note: Il convient de s'assurer que le chemin de la clé privée, le préfixe et le chemin de développement par défaut sont renseignés.

La commande `wapt-get build-package C:\waptdev\tis-tightvnc-wapt` renvoie :

```
Building  C:\waptdev\tis-tightvnc-wapt

Package tis-tightvnc (=2.8.5.0-0) content:
 setup.py
 tightvnc-2.8.5-gpl-setup-64bit.msi
 WAPT\control
 WAPT\wapt.psproj
...done. Package filename C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt
Signing C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt

7-Zip [64] 16.04: Copyright (c) 1999-2016 Igor Pavlov: 2016-10-04

Open archive: C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt
--
Path = C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt
Type = zip
Physical Size = 1756459

Updating archive: C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt
```

(suite sur la page suivante)

(suite de la page précédente)

```
Items to compress: 0

Files read from disk: 0
Archive size: 1755509 bytes (1715 KiB)
Everything is Ok
Package C:\waptdev\tis-tightvnc_2.8.5.0-0_all.wapt signed: signature:
mOQINvKGfmcW4nu05aVc8MJqMtXdPv5I0qo5zCfMkIWvEeYYDDfnZLakPkXiqptiqcNbCdY8vOPs
qFMqwSMYUyKJ8d3DHEk8kdlIldkLsiAejkdsoiZDKlEFVCJgdKI13x4FcPfoZNw5DFPzmCZKbgkU
pWvGbGFwUx/3d9zcliciN82F0FveC6C0mqoh5A==

You can upload to repository with
  C:\Program Files (x86)\wapt\wapt-get.exe upload-package "C:\waptdev\tis-tightvnc_2.8.5.0-0_all.
↪wapt"
```

wapt-get sign-package

La commande `wapt-get sign-package <path to the package>` permet de signer un paquet téléchargé manuellement avec la clé privée de l'*Administrateur* en ligne de commande.

Attention: La commande **`sign-package`** ne renomme pas le paquet avec le préfixe de l'*Organisation*.

La commande `wapt-get sign-package C:\\waptdev\\smp-7zip_16.4.0.0-1_all.wapt` renvoie :

```
Signing C:\waptdev\smp-7zip_16.4.0.0-1_all.wapt

7-Zip [64] 16.04: Copyright (c) 1999-2016 Igor Pavlov: 2016-10-04

Open archive: C:\waptdev\smp-7zip_16.4.0.0-1_all.wapt
--
Path = C:\waptdev\smp-7zip_16.4.0.0-1_all.wapt
Type = zip
Physical Size = 2857855

Updating archive: C:\waptdev\smp-7zip_16.4.0.0-1_all.wapt

Items to compress: 0

Files read from disk: 0
Archive size: 2856021 bytes (2790 KiB)
Everything is Ok
Package C:\waptdev\smp-7zip_16.4.0.0-1_all.wapt signed: signature:
lAxMJBKlnZLFQG81Rwb80+cB6XHcNjazmVJI7+PLLcPfFkFVC5wojyMPVMKhUrjrSlWomj85L8CY
gZv/FsVspUij45TcikukbF8Rr+jy6saHskg42XINqZWCnP28k4bkIREdzYIkuKDABfr15gt3ecuN
E21ZU/SI8BtXOX/80w9hpbP6ivCzTaYZZk18dhLDzV04xM9QwPSZ2mjQspbVklpm2NL4F6gb5b9D
EwMjus74/MNc6BZeKtMcFcE3Ft18ROAJeF5hLws24jjCv6Gjjus+zlGlepWK0M2p7rIdvmC1BWB/
Y6e1mQpSoisAvhOpATFPqNJca/QTMANKiTD3OA==
```

wapt-get build-upload

La commande `wapt-get build-upload <chemin du paquet>` permet de construire et d'uploader le paquet résultant sur le dépôt WAPT local.

Indication: Avec le paramètres *-i* on incrémente directement la version du paquet WAPT sans avoir à modifier le fichier `control`.

La commande `wapt-get -i build-upload C:\waptdev\tis-tightvnc-wapt` renvoie :

```
Building  C:\waptdev\tis-tightvnc-wapt
Package tis-tightvnc (=2.8.5.0-1) content:
 setup.py
 tightvnc-2.8.5-gpl-setup-64bit.msi
 WAPT\control
 WAPT\wapt.psproj
...done. Package filename C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
Signing C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt

7-Zip [64] 16.04: Copyright (c) 1999-2016 Igor Pavlov: 2016-10-04

Open archive: C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
--
Path = C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
Type = zip
Physical Size = 1756458

Updating archive: C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt

Items to compress: 0

Files read from disk: 0
Archive size: 1755509 bytes (1715 KiB)
Everything is Ok
Package C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt signed: signature:
FVn2yx77TwUHaDauSPHxJZiPAyMQe4PqLF5n6wY9YPAwY4ijHe6NgDFrexXf8ZYbHAiNa5b8V/Qj
wTVHiqpbXnZotiVIGrJDhgbaLwZ9CK6pfWiflC4126nx6PMF3T1i6w0R0NOE2wJpOSRYESk7lDUz
9CPfzJCLcOXwh0F5eZc96wbkDkSbpn1f+x5tOlvyy/FW2m8RbZQhJcO21j9gGX7It0QNecaOxXgz
qkZZKBDNASOBYAF22M1+zHb59DWQ63Q8yMj5t5szEUTkGtQNG6vZz3gb9Yraq361BIGaBDYUM31j
ZgpaHvP0vdK3c1x1mhyhC7q6eZ/UCW5tETTCiA==

Uploading files...
WAPT Server user :admin
WAPT Server password:
Status: OK, tis-tightvnc_2.8.5.0-1_all.wapt uploaded, 1 packages analysed
```

wapt-get duplicate

La commande `wapt-get duplicate <package source> <package new_duplicate>` permet de dupliquer localement un paquet WAPT du dépôt.

La commande `wapt-get duplicate tis-firefox tis-firefox-custom` renvoie :

```
Package duplicated. You can build the new WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\tis-firefox-custom-wapt
You can build and upload the new WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\tis-firefox-custom-wapt
```

wapt-get edit

La commande `wapt-get edit <package name>` permet d'éditer un paquet.

La commande `wapt-get edit tis-firefox` renvoie :

```
Package edited. You can build and upload the new WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe -i build-upload C:\waptdev\tis-firefox-wapt
```

wapt-get edit-host

La commande `wapt-get edit-host <host FQDN>` permet d'éditer un paquet *host* depuis le dépôt WAPT.

wapt-get upload-package

La commande `wapt-get upload-package <path to the package>` permet de charger un paquet sur le dépôt WAPT.

La commande `wapt-get upload-package C:\\waptdev\ is-tightvnc_2.8.5.0-1_all.wapt` renvoie :

```
WAPT Server user :admin
WAPT Server password:
tis-tightvnc_2.8.5.0-1_all.wapt uploaded, 1 packages analysed
result: OK
```

wapt-get update-packages

La commande `wapt-get update-packages <path to folder>` permet de scanner un dossier local et de créer un fichier d'index `Packages`.

La commande `wapt-get update-packages D:\\Data\\WAPT` renvoie :

```
Packages filename: D:\waptdev\Packages
Processed packages:
  D:\Data\WAPT\groupe_base.wapt
  D:\Data\WAPT\tis-firefox_50.1.5.0-0_all.wapt
  D:\Data\WAPT\tis-tightvnc_2.8.5.0-1_all.wapt
  D:\Data\WAPT\tis-7zip_16.4.0.0-1_all.wapt
```

(suite sur la page suivante)

(suite de la page précédente)

```
  D:\Data\WAPT\tis-mumble_3.14-3_all.wapt
  D:\Data\WAPT\tis-noforcereboot_1.0-1_all.wapt
Skipped packages:
```

6.7.5 Utiliser WAPTexit

Le composant **waptexit** permet l'installation des mises à jour et des paquets WAPT sur le système lors de l'extinction de l'ordinateur, à la demande de l'utilisateur ou à un moment programmé.

Son fonctionnement est simple. Si des paquets sont en attente d'installation, ils sont installés.

Indication: Dans quelles situations utiliser WAPTexit ?

La méthode WAPTexit est efficace dans la grande majorité des situations quand on veut laisser les choses se faire par elles-mêmes. Elle ne nécessite pas l'intervention de l'*Utilisateur* ou de l'*Administrateur*.

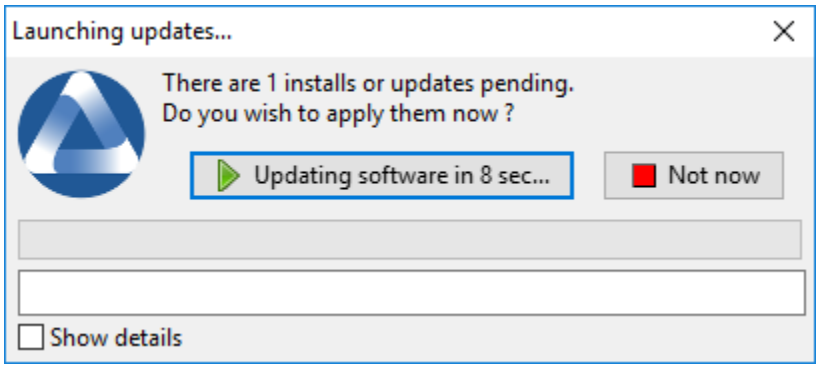


Figure120: Fenêtre WAPTexit
WAPTexit

waptexit s'exécute à l'extinction de la machine; il est installé par défaut avec l'agent WAPT.

Le comportement de **waptexit** est personnalisable dans `C:\Program Files (x86)\wapt\wapt-get.ini`.

Exécution manuelle

En créant un raccourci sur le bureau des utilisateurs, on permet aux utilisateurs de mettre à jour leurs ordinateurs facilement en cliquant sur l'icône *WAPTexit*.

Le comportement de **waptexit** est personnalisable dans `C:\Program Files (x86)\wapt\wapt-get.ini`.

Déclencher WAPTexit avec une tâche planifiée

On peut déployer un GPO ou un paquet WAPT qui déclenchera WAPTexit à un moment programmé à l'avance.

Le déclenchement de WAPTexit avec une tâche planifiée est mieux adapté aux serveurs qui ne sont pas arrêtés fréquemment.

Vous pouvez adapter la procédure décrivant comment déployer l'agent WAPT pour *déclencher le script WAPTexit.exe au moment de votre choix*.

Indication: Vous pouvez utiliser le script suivant pour votre tâche programmée, adapté à votre besoin (**Enterprise uniquement**) :

```
waptpython -c "from waptenterprise.waptservice.enterprise import start_waptexit;
start_waptexit('',{'only_priorities':False,'only_if_not_process_running':True,
'install_wua_updates':False,'countdown':300},'schtask')"
```

Avertissement: Tous les logiciels en cours d'exécution qui sont mis à jour peuvent être terminés avec une perte possible de données. WAPTexit peut ne pas réussir à mettre à jour un logiciel si un logiciel que vous mettez à jour se trouve dans la liste `impacted_process` du fichier `control` d'un des logiciels que vous essayez de mettre à jour. Voir *below* pour plus d'informations.

La méthode consistant à déclencher WAPTexit à une heure programmée est la moins recommandée pour les ordinateurs de bureau. Il est préférable de laisser WAPTexit s'exécuter à l'arrêt de la machine ou à l'initiative de l'utilisateur.

Empêcher l'interruption de l'installation des mises à jour

Pour désactiver l'interruption de l'installation des mises à jour vous pouvez lancer **`waptexit`** avec l'argument :

```
waptexit.exe -allow_cancel_upgrade = True
```

Sinon **`waptexit`** va prendre la valeur indiqué dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini`:

```
[global]
allow_cancel_upgrade = False
```

If this value is not indicated in `C:\Program Files (x86)\wapt\wapt\wapt-get.ini`, then the default value will be **10**.

Augmenter le temps de déclenchement dans waptexit

Pour préciser le temps d'attente avant le lancement automatique des installations vous pouvez lancer **`waptexit`** avec l'argument :

```
waptexit.exe -waptexit_countdown = 10000
```

Sinon **`waptexit`** prendra la valeur indiquée dans le fichier de configuration `C:\Program Files (x86)\wapt\wapt-get.ini` :

```
[global]
waptexit_countdown = 25
```

If this value is not indicated in `C:\Program Files (x86)\wapt\wapt\wapt-get.ini`, then the default value will be **1**.

Ne pas interrompre l'activité des utilisateurs

Pour indiquer à WAPT de ne pas lancer l'"**upgrade** de logiciels en cours d'exécution sur la machine (attribut *impacted_process* du paquet), vous pouvez lancer **waptexit** avec l'argument :

```
waptexit.exe -only_if_not_process_running=True
```

Sinon **waptexit** va prendre la valeur indiqué dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini`:

```
[global]
upgrade_only_if_not_process_running = True
```

If this value is not indicated in `C:\Program Files (x86)\wapt\wapt\wapt-get.ini`, then the default value will be **False**.

Lancer l'installation des paquets avec une prioritée spécifique

Pour indiquer à WAPT de lancer uniquement les installations des paquets avec une priorité élevée, vous pouvez lancer **waptexit** avec l'argument :

```
waptexit.exe -priorities = high
```

Sinon **waptexit** va prendre la valeur indiqué dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini`:

```
[global]
upgrade_priorities = high
```

If this value is not indicated in `C:\Program Files (x86)\wapt\wapt\wapt-get.ini`, then the default value will be **Empty** (no filter on priority).

Personnaliser waptexit

Il est possible de personnaliser waptexit en plaçant l'image que vous souhaitez dans `C:\Program Files (x86)\wapt\templates\waptexit-logo.png`.

Activer / Désactiver WAPTexit

Pour activer ou désactiver **waptexit** dans les stratégies locales d'arrêt de la machine, utilisez la commande suivante :

- pour activer **waptexit** à l'extinction de la machine :

```
wapt-get add-upgrade-shutdown
```

- pour désactiver **waptexit** à l'extinction de la machine :

```
wapt-get remove-upgrade-shutdown
```

6.8 Comment utiliser WAPT Enterprise

Cette section de la documentation couvre l'utilisation des fonctionnalités de WAPT Enterprise. Nouveau dans la version 1.7: Enterprise

6.8.1 Utilisation des paquets d'unités organisationnelles dans WAPT

Indication: Cette fonctionnalité est disponible uniquement dans la version **WAPT Enterprise**.

Principe de fonctionnement

WAPT Enterprise offre des fonctionnalités d'unités organisationnelles.

Cette méthode automatise les installations de logiciels basées sur votre infrastructure Active Directory.

L'agent WAPT est conscient de sa position dans l'arborescence Active Directory, il connaît donc la hiérarchie des Unités Organisationnelles qui le concerne, par exemple :

```
DC=ad,DC=domain,DC=lan
OU=Paris,DC=ad,DC=domain,DC=lan
OU=computers,OU=Paris,DC=ad,DC=domain,DC=lan
OU=service1,OU=computers,OU=Paris,DC=ad,DC=domain,DC=lan
```

Si un paquet d'Unité Organisationnelle est défini à chaque niveau, l'agent WAPT téléchargera automatiquement les paquets et les configurations qui sont attachés à chaque niveau, par héritage, et appliquera les paquets attachés et leurs dépendances.

Filtres et actions disponibles avec les Unités Organisationnelles

Indication: Vous pouvez voir sur l'image que les actions **vérifier les mises à jour** et **mettre à jour** peuvent être effectuées au travers de ce menu, vous pouvez sélectionner ainsi les hôtes rattachés à l'Unité Organisationnelle sélectionnée.

Dans la version **Enterprise**, vous pouvez filtrer la façon dont les hôtes sont affichés en fonction OU auquel ils appartiennent dans l'Active Directory.

La case à cocher *Inclure les hôtes dans les sous-dossiers* permet d'afficher les hôtes dans les sous-dossiers.

Créer des paquets d'Unités Organisationnelles dans la console WAPT

Vous pouvez créer des paquets *unit* en faisant *Clic droit sur une UO* → *Créer ou éditer le paquet UO*.

Une fenêtre s'ouvre et vous êtes invité à choisir les paquets à mettre en dépendance de l'UO.

Enregistrez le paquet et il sera chargé sur le serveur WAPT.

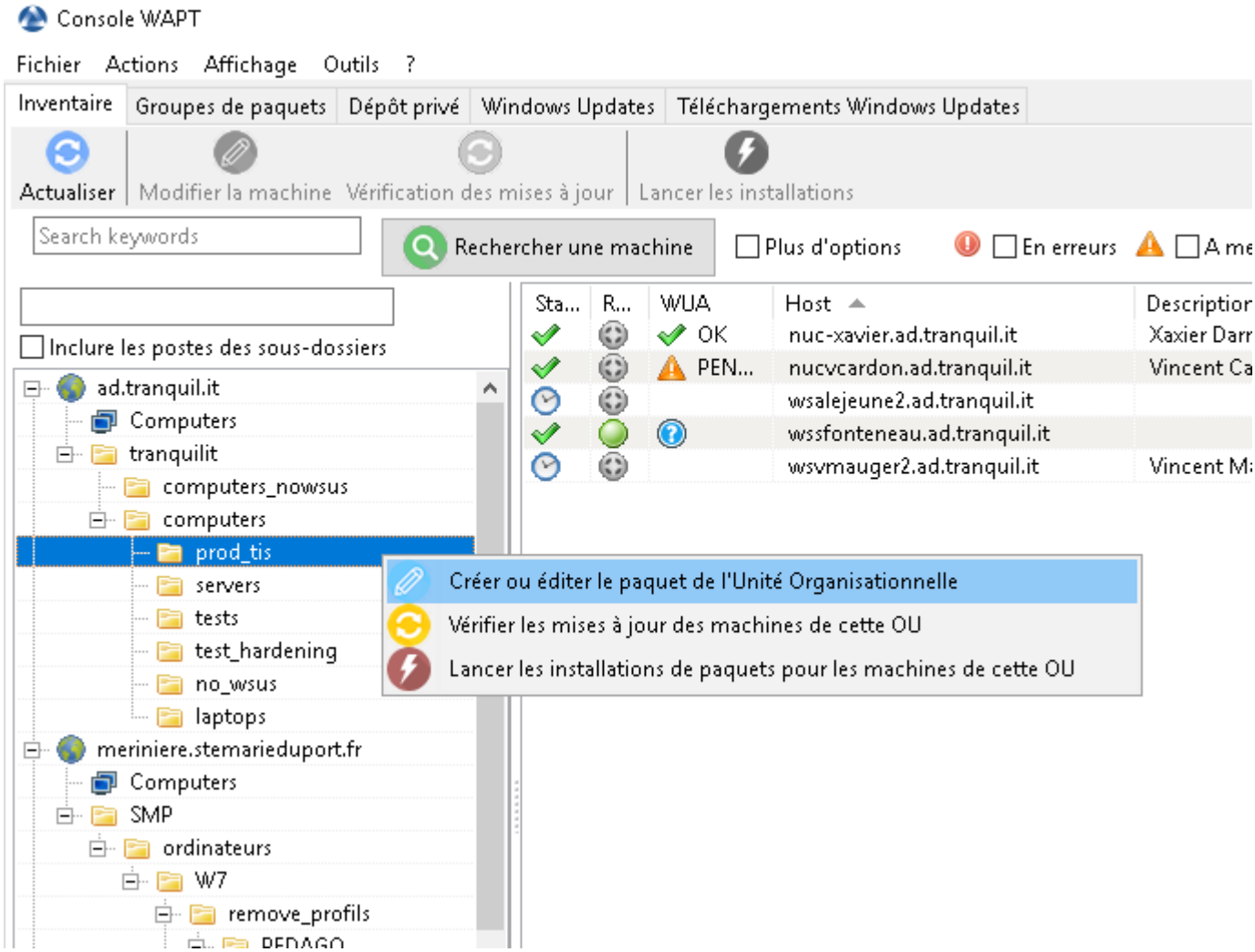


Figure121: Console WAPT montrant les options applicables à l'UO

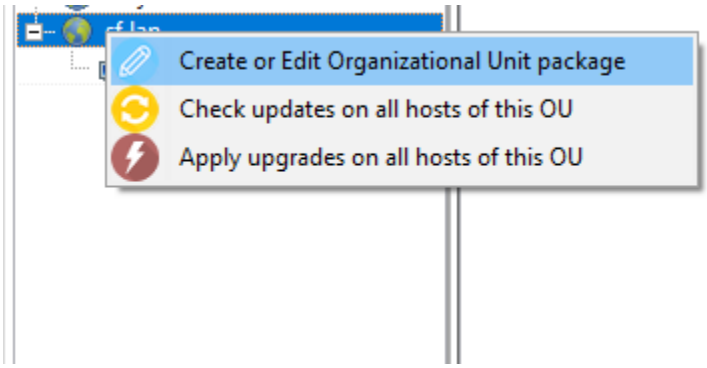


Figure122: Cliquez avec le bouton droit de la souris sur OU pour créer un paquet d'unités.

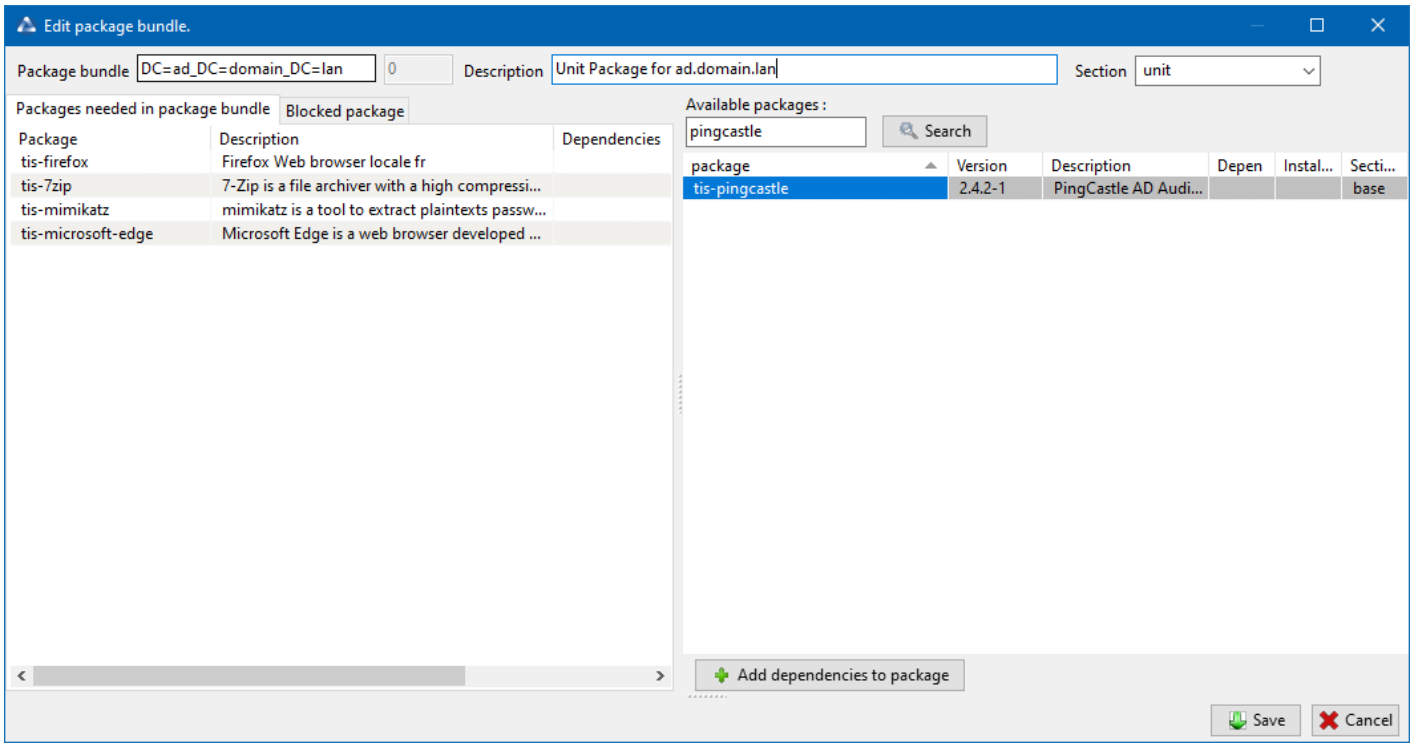


Figure123: Ajouter un paquet à l'UO.

Simuler une Unité Organisationnelle pour des machines en WORKGROUP

Il peut arriver que certains hôtes spécifiques ne puissent pas être joints à un domaine Active Directory.

Avec cette spécificité, ces hôtes n'apparaissent pas dans vos Unités Organisationnelles Active Directory dans votre console WAPT.

Pour que tous les hôtes apparaissent dans la console sous la bonne Unité Organisationnelle, qu'ils soient liés à un domaine AD ou non, WAPT vous permet de spécifier un fichier de configuration d'agent WAPT avec une Unité Organisationnelle *simulée*.

Les avantages de l'astuce sont :

- vous pouvez gérer ces machines avec WAPT comme si elles étaient jointes à l'AD ;
- Les machines hors domaine et les machines en mode groupes de travail apparaissent désormais dans l'arborescence AD ;
- les paquets *unit* fonctionnerons avec ces machines ;

Pour configurer une Unité Organisationnelle *simulée* sur les machines, créer un paquet WAPT vide ;

```
wapt-get make-template demo-configure-fake-ou
```

Utilisez ensuite le code suivant :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
```

(suite sur la page suivante)

(suite de la page précédente)

```
print('Setting Fake Organizational Unit')
fake_ou = "OU=TOTO,OU=TEST,DC=DEMO,DC=LAN"
inifile_writestring(WAPT.config_filename,'global','host_organizational_unit_dn',fake_ou)
```

Le paramètre `host_organizational_unit_dn` doit être comme ci-dessous dans `wapt-get.ini` :

```
[global]
host_organizational_unit_dn="OU=TOTO,OU=TEST,DC=DEMO,DC=LAN"
```

Note: Stick to a specific case with your `host_organizational_unit_dn` (don't mix « dc »s and « DC »s, « ou »s and « OU »s...). Follow the case used in the DN/`computer_ad_dn` fields in the hosts grid.

Nouveau dans la version 1.7: Enterprise

6.8.2 Utiliser des paquets profils dans WAPT

Indication: Cette fonctionnalité est disponible uniquement dans la version **WAPT Enterprise**.

Principe de fonctionnement

WAPT Enterprise permet d'utiliser la fonctionnalité des paquets profile Active Directory.

Cela permet d'automatiser l'installation de logiciels à partir des groupes de sécurité d'Ordinateurs Active Directory.

Important: **Les groupes de sécurité d'Ordinateur contiennent des Ordinateurs, pas des Utilisateurs**.

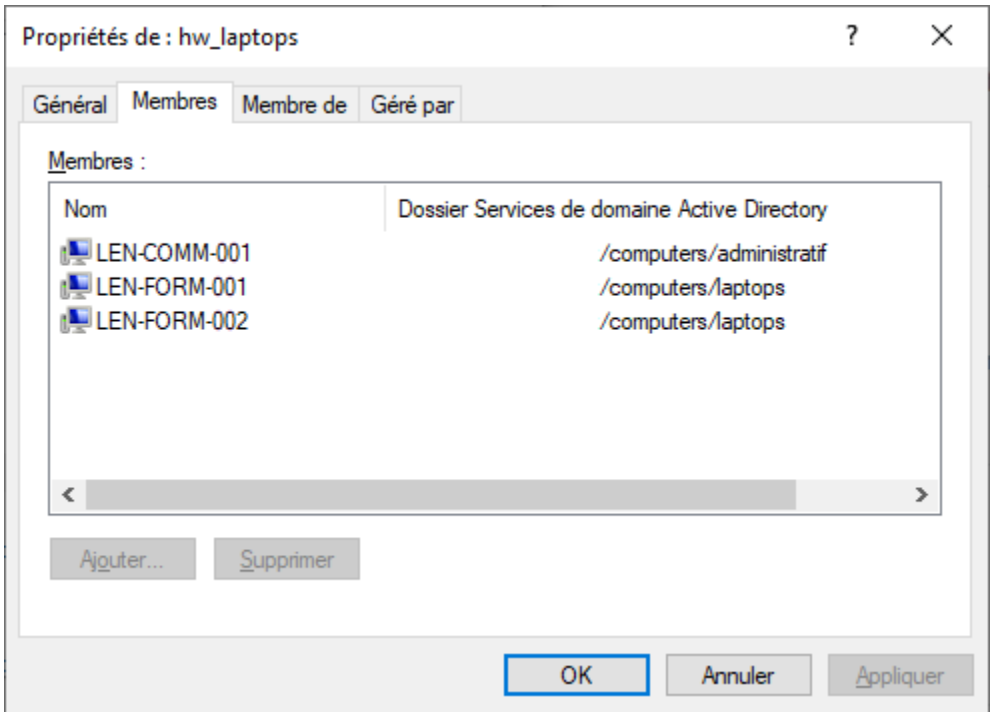


Figure124: Groupe d'ordinateurs Active Directory

L'installation automatique de logiciels et de configurations basées sur un utilisateur ou l'appartenance à un groupe d'utilisateurs n'est pas mise en œuvre avec WAPT. Ce cas d'utilisation est mieux servi par la fonction de *self-service différencié* qui est également disponible avec WAPT Enterprise.

Creation d'un paquet profile dans la console WAPT

Vous pouvez créer un paquet *profile* en cliquant sur *Groupes -> Nouveau profil AD*.

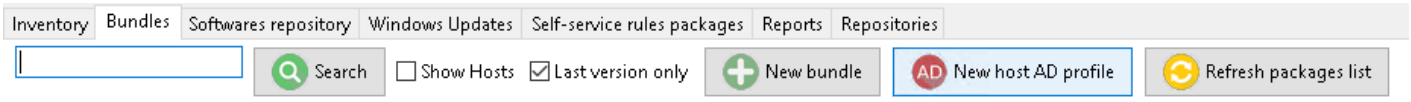


Figure125: Cliquer sur Nouveau paquet profile pour créer un paquet *profile*

Important: Pré-requis :

- le nom du paquet *profil* doit être **exactement** le même que le nom du groupe de sécurité AD ;
- le nom du paquet *profil* est sensible à la casse ;

Exemple :

- Groupe de sécurité AD : `HW_laptops` ;
- Paquet *profile* WAPT : `HW_laptops` ;

Une fenêtre s'ouvre et vous demande de choisir les paquets qui doivent être dans ce paquet **profile**.

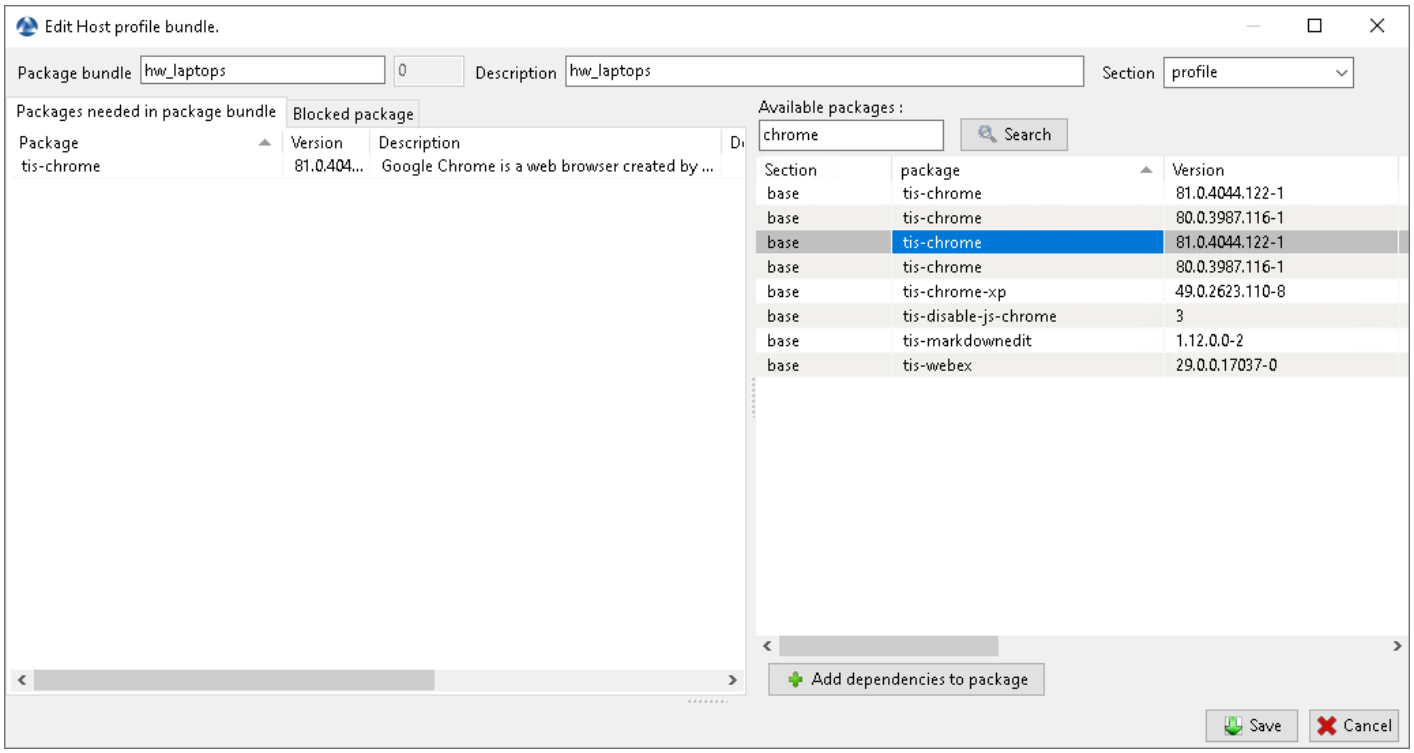


Figure126: Ajouter des paquets au paquet profile

Sauvegarder le paquet *profile* et il sera chargé sur le serveur WAPT. Nouveau dans la version 1.7: Enterprise

6.8.3 Utiliser l'agent de mise à jour Windows (WAPTWUA)

Note: Depuis la version 1.7, WAPT est capable de gérer les mises à jour Windows sur vos terminaux.

- the internals of WAPTWUA is based on the WUA (Windows Update Agent) API,
- pour plus d'informations : https://docs.microsoft.com/en-us/windows/win32/wua_sdk/using-the-windows-update-agent-api ;

Principe de fonctionnement

Régulièrement, le serveur WAPT télécharge un fichier `wsusscn2.cab` mis à jour à partir des serveurs Microsoft. Par défaut, les téléchargements ont lieu une fois par jour et aucun téléchargement n'est déclenché si le fichier `wsusscn2.cab` n'a pas été modifié depuis le dernier téléchargement.

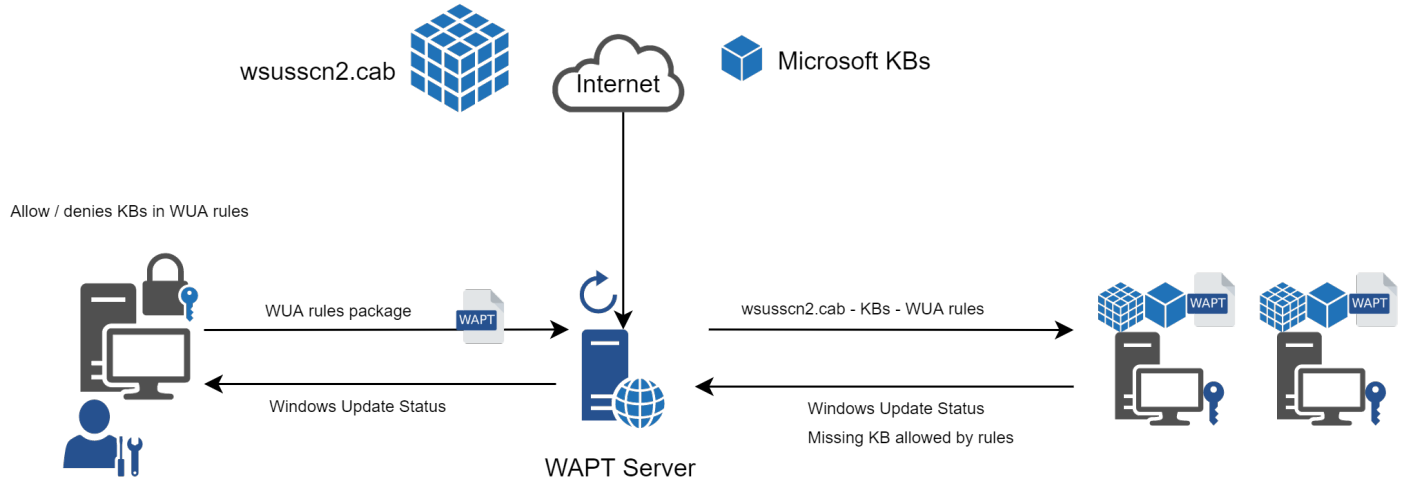


Figure127: WAPT Windows Update flow process.

Note: Dans certains cas, vous souhaiterez peut-être diffuser de nouveaux KB avant la prochaine sortie du *Patch Tuesday*.

Pour cela, vous pouvez suivre *cette documentation* pour packager des fichiers `.msu` pour ces mise à jour ponctuelles.

Le fichier `wsusscn2.cab` est ensuite téléchargé par l'agent WAPT à partir de son dépôt le plus proche et ensuite transmis à l'utilitaire standard *Windows Update Agent* Windows qui traitera l'arbre des mises à jour.

Régulièrement, la machine analysera les mises à jour disponibles en utilisant le fichier `wsusscn2.cab`. La liste des mises à jour nécessaires déterminée par l'agent WUA est ensuite envoyée par l'hôte au serveur WAPT.

Si une mise à jour est en attente sur la machine et si cette dernière n'est pas présente sur le serveur WAPT, le serveur téléchargera la mise à jour nécessaire depuis les serveurs Microsoft.

Indication: This mode of operation allows WAPT to download only the necessary updates on the computers, thus saving bandwidth, download time and disk space.

Note: Les mises à jour téléchargées sont stockées :

- sur des serveurs Linux dans `/var/www/waptwua` ;
- sur des machines Windows dans `C:waptwaptserverrepositorywaptwua` ;

L'URL de téléchargement des mise à jour Windows depuis le dépôt WAPT est basée sur le paramètre `repo_url` du fichier `wapt-get.ini` :

- la réplication des WAPT Windows Update fonctionne pour réduire l'usage de la bande passante ;

- n'oubliez pas de synchroniser le dossier `waptwua` si vous répliquez vos paquets avec des dépôts distants ;

Note: Si dans votre organisation, un proxy est nécessaire pour sortir sur Internet, alors assurez-vous de *définir le serveur proxy dans le fichier waptserver.ini*.

Différence de fonctionnement des mises à jour Windows entre WAPT et WSUS

WSUS downloads by default the updates for selected categories. This can lead to a very large update database and lots of storage used.

WAPT Windows Update ne télécharge que les mises à jour qui ont été demandées par au moins une machine. Cela permet de conserver une petite base de données (quelques dizaines de gigaoctets) et elle peut être facilement nettoyée si vous voulez récupérer de l'espace.

Mises à jour majeures du système d'exploitation

Major OS upgrades are upgrades from one OS version to another. That includes, for example, upgrades from Windows 7 to Windows 10, or from Windows 10 1803 to Windows 10 1903.

Major version upgrades are not handled in the same way as minor OS upgrades. Major upgrades are handled via the download of the new install ISO content (same content as for a fresh install) and running the **`setup.exe`** with the correct parameters. This process is the same for WSUS, SCCM and WAPT Windows Updates.

Dans le cas des mises à jour Windows WAPT, vous devez créer un paquet de mise à jour du système d'exploitation en utilisant un paquet modèle fourni sur https://store.wapt.fr.

Mises à niveau des pilotes

Les mises à niveau des conducteurs via le WSUS ne sont pas recommandées car il est difficile de gérer correctement les effets secondaires. Dans le cas des mises à jour dans WAPT, les **PILOTES NE SONT PAS TÉLÉCHARGÉS** car ils ne sont pas référencés dans les fichiers `wsusscn2.cab` fournis par Microsoft.

Il est recommandé de pousser les mises à jour des pilotes via un paquet WAPT personnalisé. Si le correctif pour un pilote est packagé sous la forme d'un *`msu`*, vous pouvez le conditionner sous la forme d'un paquet WAPT standard.

Just select the `msu` file and click :menuselection: »create package » in the WAPT console to launch the wizard for simplified package creation.

Si la mise à jour du pilote est présentée sous la forme d'un paquet *`zip`* contenant le fichier `exe`, vous pouvez créer un paquet WAPT contenant les fichiers nécessaires et le binaire **`setup.exe`** avec les paramètres silencieux corrects.

KB hors bande

Microsoft sometimes provides OOB (Out of Band) updates that are not contained in the `wsusscn2.cab` index. Those updates are not included in the main update because they may fix a very specific problem or may have drawbacks in some situations.

Si vous souhaitez déployer une mise à jour OOB KB, vous pouvez la télécharger à partir du catalogue microsoft https://www.catalog.update.microsoft.com/Home.aspx.

Just select the `msu` file and click *Create package* in the WAPT console to launch the wizard for simplified package creation.

Vous devez faire attention au fait que les mises à jour OOB peuvent casser votre système, assurez-vous de lire les conditions préalables sur le bulletin Microsoft correspondant à la mise à jour et de tester la mise à jour de manière approfondie.

Configurer WAPTWUA sur l'agent WAPT

WAPTWUA est configuré dans `wapt-get.ini`.

Ajouter une section `[waptwua]`.

Vous avez alors plusieurs options :

Table24: Options de configuration dans la section `[waptwua]` du `wapt-get.ini`

Options	Valeur par défaut	Description
`enabled`	False	Activez ou désactivez WAPTWUA sur cette machine.
`allow_direct_download`	False	Autoriser le téléchargement direct des mises à jour à partir des serveurs Microsoft si le serveur WAPT n'est pas disponible
`default_allow`	False	Définir si la mise à jour manquante est autorisée ou non par défaut
`filter`	Type="Software" ou Type="Driver"	Définir le filtre à appliquer pour l'analyse des mises à jour Windows
`download_scheduling`	None	Définir la fréquence d'analyse des mises à jour Windows (Ne fera rien si la règle du paquet *waptwua* ou le fichier `wsusscn2.cab` n'ont pas changé) (ex : 2h)
`install_scheduling`	None	Définir la fréquence d'installation des mises à jour Windows (Ne fera rien si aucune mise à jour n'est en attente) (ex : 2h)
`install_at_shutdown`	False	Installe les mises à jour à l'extinction de la machine
`install_delay`	None	Définir un délai d'installation différé avant la publication dans le dépôt (ex : 7d)
`allowed_severities`	None	Définit une liste de niveaux de gravité qui sera automatiquement acceptée lors d'un scan de mise à jour Windows avec WAPT Windows Update. ex : *Important*, *Critical*, *Moderate*

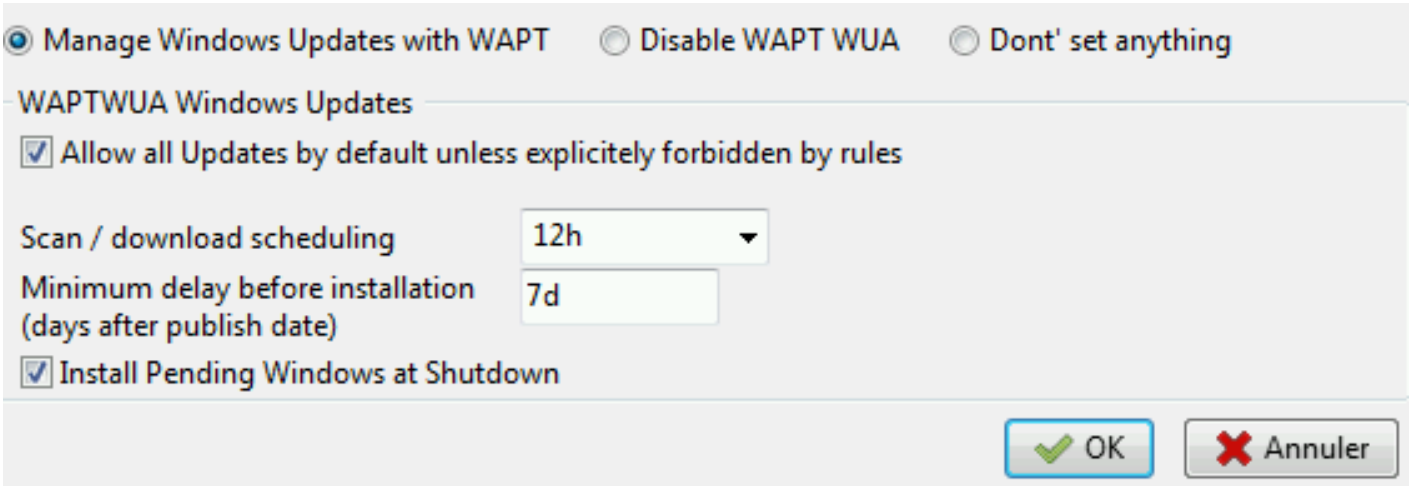
Indication: Ces options peuvent être définies lors de la génération de l'agent.

Exemple de section `[waptwua]` dans le fichier `wapt-get.ini` :

```
[waptwua]
enabled =true
offline =true
default_allow =false
allow_direct_download=false
download_scheduling=12h
install_at_shutdown=true
install_scheduling=12h
install_delay=7d
```

The *install_scheduling* option will try every 12 hours to install updates on the client. It is not in graphical options due to a potential danger. Indeed, trying to install updates on your IT infrastructure while working hours can impact your production.

Lorsque vous créer waptagent.exe depuis la console, ces options sont équivalentes à celles-ci :



Indication: si l'option *default_allow* est à `True` et que Wapt WUA est activé aussi,le client va contacter le serveur WAPT et va demander à télécharger les mises à jour manquantes. Le client installera ses mises à jour tout seul.

Exemple de code source d'un paquet pour modifier les paramètres `[waptwua]` :

```
def install():
 inifile_writestring(WAPT.config_filename,'waptwua','enabled','true')
 inifile_writestring(WAPT.config_filename,'waptwua','offline','true')
 inifile_writestring(WAPT.config_filename,'waptwua','filter',"Type='Software' or Type='Driver'")
 inifile_writestring(WAPT.config_filename,'waptwua','install_at_shutdown','true')
 inifile_writestring(WAPT.config_filename,'waptwua','download_scheduling','7d')
 inifile_writestring(WAPT.config_filename,'waptwua','allowed_severities','Critical,Important')
```

Utiliser de WAPTWUA depuis la console

L'onglet *WAPT Windows Update Agent* de la console WAPT est fourni avec deux sous-menus pour gérer WAPTWUA.

Paquet WAPTWUA

L'onglet *WAPTWUA Package* vous permet de créer des paquets de règles *waptwua*.

- lorsque ce type de paquet est installé sur une machine, il indique à l'agent WAPTWUA les KBs (Knowledge Base articles) autorisés ou interdits ;
- lorsque plusieurs paquets *waptwua* sont installés sur une machine, les différentes règles seront fusionnées ;
- lorsqu'un `cab` n'est ni mentionné comme autorisé, ni mentionné comme interdit, les agents WAPT prendront alors la valeur `default_allow` dans le fichier `wapt-get.ini` ;

Si une mise à jour Windows n'a pas encore été téléchargée sur le serveur WAPT, l'agent WAPT marquera la mise à jour comme *MISSING*.

Note:

- si la configuration de l'agent WAPTWUA est définie avec `default_allow = True`, alors il sera nécessaire de spécifier la cabine interdite ;
- si la configuration de l'agent WAPTWUA est définie avec `default_allow = False`, alors il sera nécessaire de spécifier le `cab` autorisé ;

Indication:

- pour tester les mises à jour sur un petit nombre d'ordinateurs, vous pouvez définir la valeur par défaut de WAPTWUA avec `default_allow = False` ;
- vous pouvez tester les mises à jour sur un petit échantillon de machines et si tout va bien, vous pouvez diffuser les mises à jour sur l'ensemble de la base des ordinateurs ;

Onglet Liste des mises à jour de Windows

L'onglet *Windows Update List* liste toutes les mises à jour Windows nécessaires.

Important: Le serveur ne scanne pas le fichier `wsussc2.cab` lui-même, il laisse les agents WAPT le faire. Si une mise à jour vous semble manquer dans la liste, vous devez lancer un scan sur l'une des machines présentes dans la console. Si vous lancez un scan WUA sur un agent Windows 7, les fichiers CAB et Windows 7 seront affichés sur l'onglet Liste des mises à jour de Windows.

Le volet de gauche affiche les catégories de mises à jour, vous permettant de filtrer par :

- la criticité ;
- le produit ;
- la classification ;

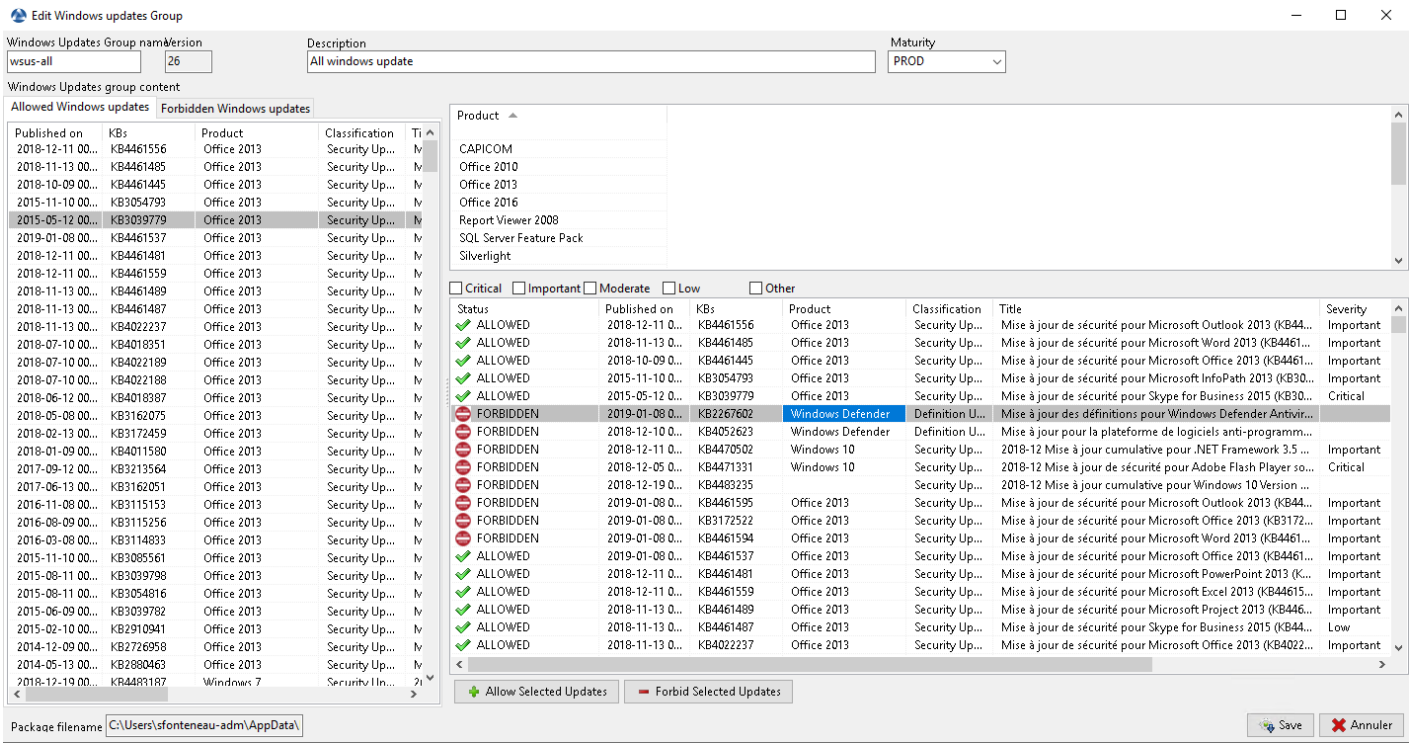


Figure128: Créer un paquet *waptwua*

Dans la grille de droite, si la colonne *Downloaded on* est vide, cela signifie que la mise à jour n'a pas encore été téléchargée par le serveur WAPT et n'est pas présente sur le serveur WAPT (Cette mise à jour ne manque sur aucun hôte).

- you can force the download of an update by *right-clicking* → *Download*;
- vous pouvez également forcer le téléchargement du fichier `wsusscn2.cab` avec le bouton *Download WSUSScan cab from Microsoft Web Site* ;
- vous pouvez voir le téléchargement des mises à jour Windows sur le serveur avec le bouton *Montrer les téléchargements* ;

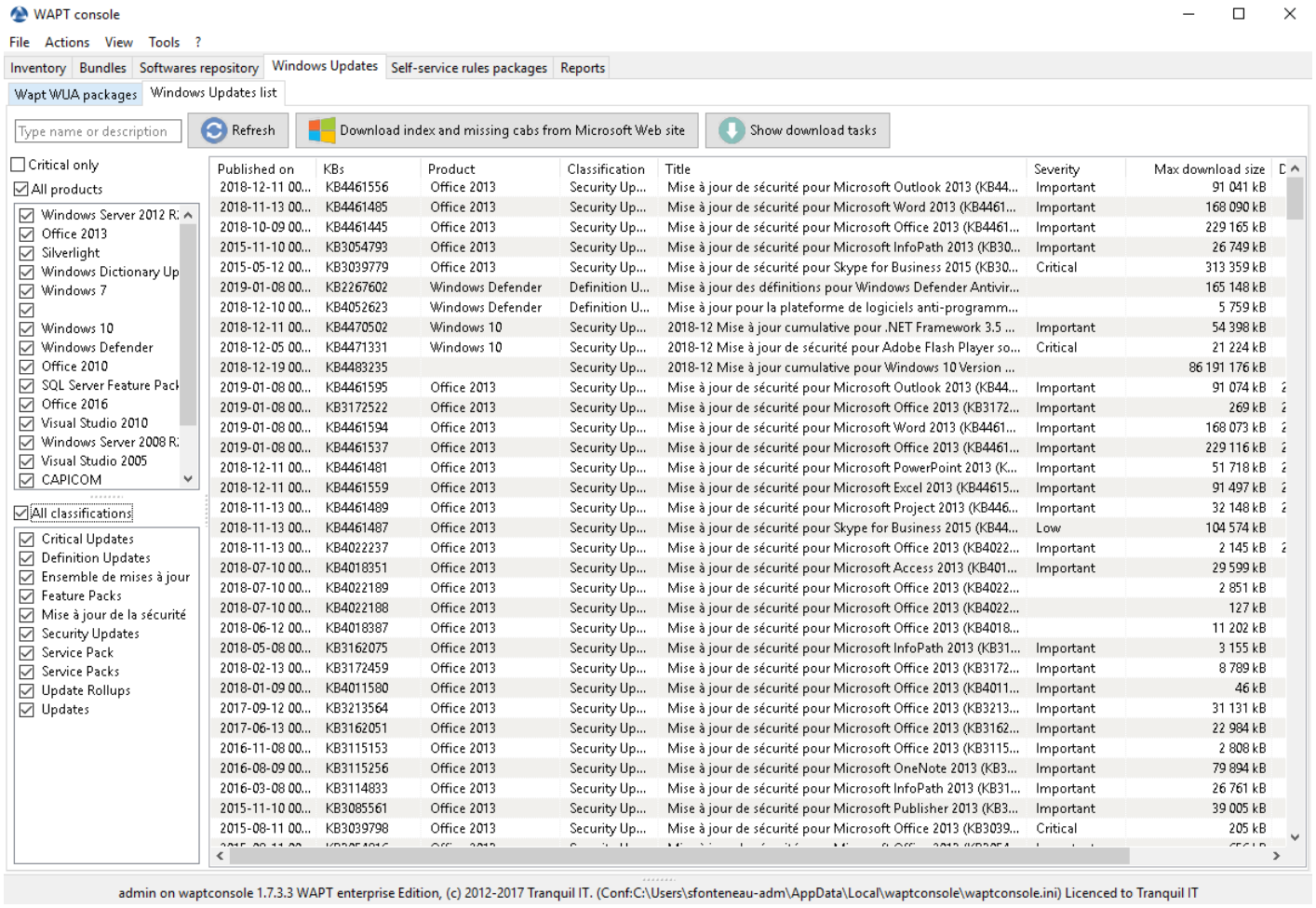
Indication: Pour nettoyer votre dossier `WAPTWUA`, vous pouvez supprimer les mises à jour Windows qui ne sont plus nécessaires. Le serveur WAPT ne re-téléchargera les mises à jour supprimées que si l'un des hôtes équipés de WAPT le demande ;

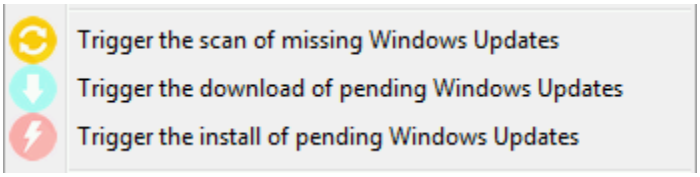
Lancer WAPT WUA sur le client

Vous avez alors trois options.

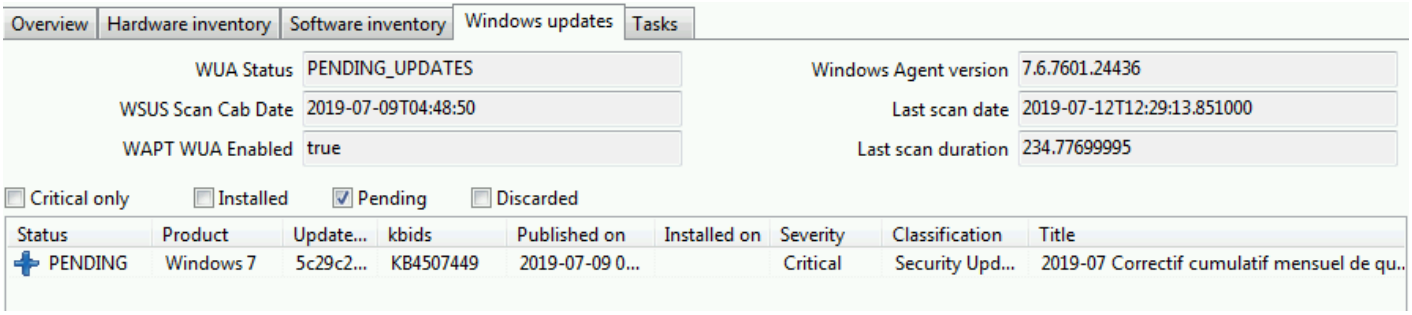
Le bouton *Rechercher des mises ç jour Windows* lance l'analyse sur le client WAPT et liste toutes les mises à jour signalées pour le système d'exploitation. Vous pouvez scanner le client depuis la console comme cela ou en utilisant la commande **`wapt-get waptwua-scan`** depuis la ligne de commande.

Indication: Toutes les 30 minutes, le serveur WAPT recherche les mises à jour qui ont été demandées au moins une fois par les clients WAPT et qui n'ont pas encore été téléchargées et mises en cache. Si une mise à jour est en attente, le serveur WAPT la téléchargera à partir des serveurs officiels de Microsoft.


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository Windows Updates Self-service rules packages Reports
Wapt WUA packages Windows Updates list
Type name or description
Refresh
Download index and missing cabs from Microsoft Web site
Show download tasks
Critical only
All products
Windows Server 2012 R:
Office 2013
Silverlight
Windows Dictionary Up
Windows 7
Windows 10
Windows Defender
Office 2010
SQL Server Feature Pack
Office 2016
Visual Studio 2010
Windows Server 2008 R:
Visual Studio 2005
CAPICOM
All classifications
Critical Updates
Definition Updates
Ensemble de mises à jour
Feature Packs
Mise à jour de la sécurité
Security Updates
Service Pack
Service Packs
Update Rollups
Updates
Published on KBs Product Classification Title Severity Max download size
admin on waptconsole 1.7.3.3 WAPT enterprise Edition, (c) 2012-2017 Tranquil IT. (Conf:C:\Users\sfonteneau-adm\AppData\Local\waptconsole\waptconsole.ini) Licenced to Tranquil IT


Trigger the scan of missing Windows Updates
Trigger the download of pending Windows Updates
Trigger the install of pending Windows Updates

Toutes les 30 minutes, le serveur va scanner tous les clients afin devérifier si une mise à jour en attente. Si elle n'est pas présente dans ses dépôts,le serveur wapt va la télécharger depuis les serveurs de Microsoft. Vous pouvezforcer ce scan avec le bouton *Télécharger l'index et les cabs manquant depuis le site de MS* dans l'onglet *Windows Updates → Liste des mises à jour Windows*



Si vous voulez télécharger depuis la console, cliquez sur le bouton *Lancer le téléchargement des Mises à jour Windows en attente*.

The command-line for downloading kb's from the client is **`wapt-get waptwua-download`**, it will scan the current status of Windows against current rules, download missing kb's and send the result to the server.

Si vous voulez installer la ou les mise(s) à jour en attente, utilisez **`wapt-get waptwua-install`** à partir de l'invite de ligne de commande.

Si vous voulez télécharger depuis la console, cliquez sur le bouton *Lancer le téléchargement des Mises à jour Windows en attente*.

Indication: Lorsque vous souhaitez installer les mises à jour en attente stockées dans le cache, le service WAPT déclenche le service WUA.

Le service WAPT activera et démarrera temporairement le service WUA pour installer les mises à jour. Lorsque les mises à jour seront installées, le service WAPT arrêtera et désactivera le service WUA jusqu'au prochain cycle.

Vidéo de démonstration

Nouveau dans la version 1.7: Enterprise

6.8.4 Utiliser les fonctions de reporting dans WAPT

Indication: Fonctionnalité disponible uniquement avec WAPT **Enterprise**.

Principe de fonctionnement

WAPT **Enterprise** offre des fonctionnalités avancées de reporting.

En effet, qui mieux que vous même pour savoir ce que vous voulez dans votre rapport.

Avec WAPT nous vous proposons d'écrire vos propres requêtes SQL pour afficher vos résultats souhaités dans votre console WAPT.

Concepteur de requêtes WAPT

Le concepteur de requêtes vous offre la possibilité d'éditer vos propres requêtes à exécuter sur la base de données PostgreSQL du serveur WAPT.

Pour créer un nouveau rapport, cliquez sur *Rapport* → *Mode Conception* → *Nouvelle Requête*.

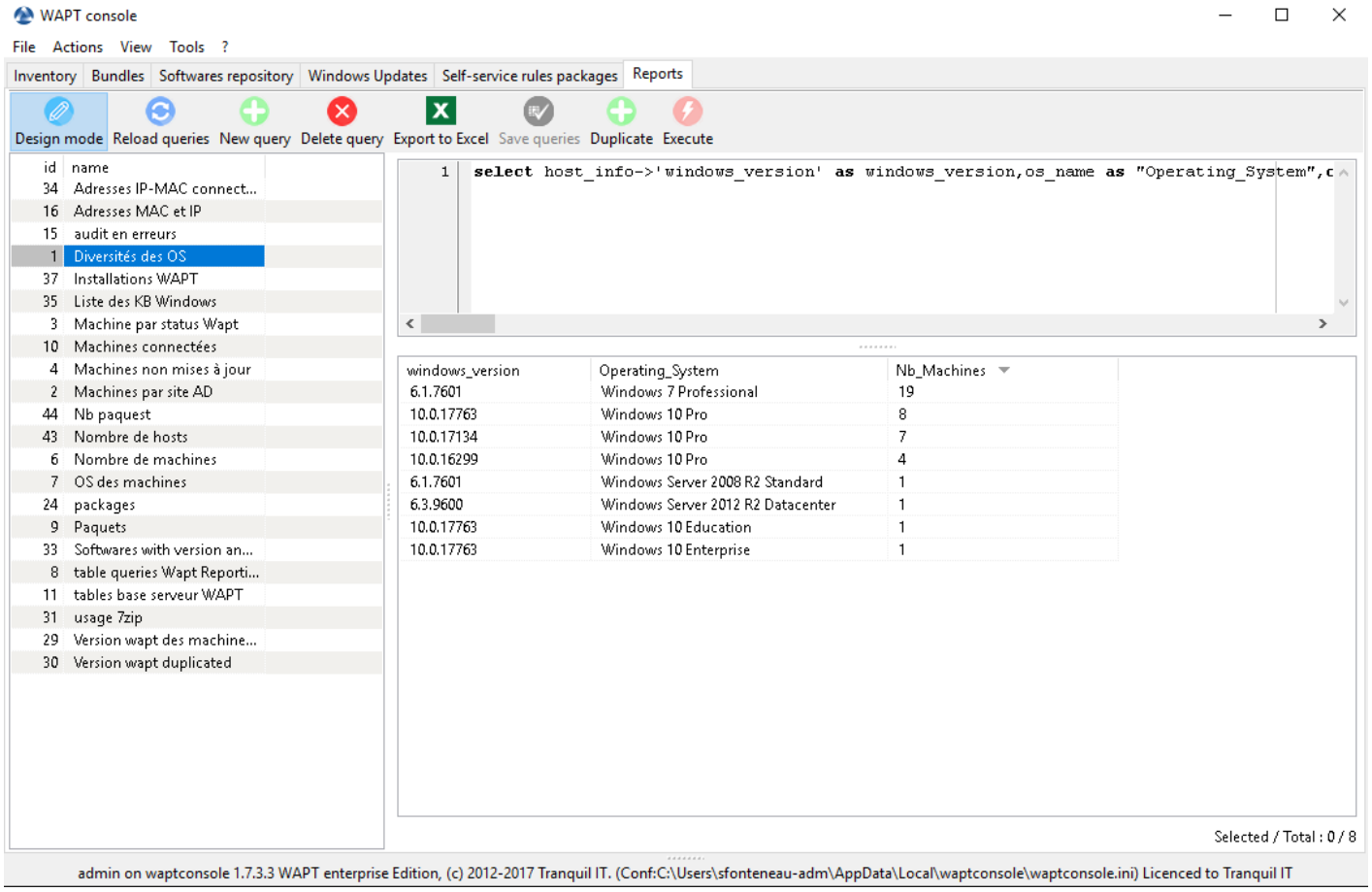


Figure129: Concevoir une requête pour le reporting WAPT

Indication:

- pour renommer une requête, appuyez sur la touche `F2` ;
- dans la bannière supérieure, vous pouvez écrire votre requête SQL ;

Pour éditer / modifier / sauvegarder vos rapports :

- le bouton *Recharger les requêtes* est utilisé pour recharger les requêtes enregistrées sur le serveur, par exemple, si un collègue vient de modifier une nouvelle requête ;
- le bouton *Nouvelle requête* ajoutera une nouvelle requête vide à la liste ;
- le bouton *Supprimer la requête* supprimera la requête sélectionnée du serveur WAPT ;
- le bouton *Exporter vers Excel* exportera le résultat de votre requête vers une feuille de calcul ;
- le bouton *Enregistrer la requête* enregistrera votre requête sur le serveur WAPT ;
- le bouton *Dupliquer* dupliquera une requête existante pour éviter d'écrire une requête à partir de zéro ;
- le bouton *Exécuter* exécute la requête sélectionnée ;

Note:

- les requêtes sont sauvegardées dans la base de données PostgreSQL du serveur WAPT ;
- le raccourci `CTRL+Espace` vous permet de construire vos requêtes plus efficacement en mode conception ;

Exemples de requêtes

Requêtes ordinateurs :

- Nombre de postes

```
select count(*) as "Nb_Machines" from hosts
```

- Liste des ordinateurs :

```
select computer_name,os_name,os_version,os_architecture,serialnr from hosts order by 4,3,1
```

- Adresses MAC et IP des ordinateurs :

```
select distinct unnest(mac_addresses) as mac,
unnest(h.connected_ips) as ipaddress,  computer_fqdn,h.description,
h.manufacturer||' '||h.productname as model,
h.serialnr,h.computer_type
from hosts h
order by 1,2,3
```

- versions de Windows:

```
select host_info->'windows_version' as windows_version,
os_name as operating_system,
count(os_name) as nb_hosts
from hosts
group by 1,2
```

- Lister les différents OS

```
select host_info->'windows_version' as windows_version,os_name as "Operating_System",count(os_
↪name) as "Nb_Machines" from hosts group by 1,2
```

- Lister les postes inactife depuis longtemps

```
SELECT h.uuid,h.computer_fqdn,install_date::date,version,h.listening_timestamp::timestamp,h.
↪connected_users from hostsoftwares s
left join hosts h on h.uuid=s.host_id
where
s.key='WAPT_is1'
and
h.listening_timestamp<'20190115'
```

- Filter les hotes par type de Chassis

```
select case
dmi->'Chassis_Information'->>'Type'
 when 'Portable' then '01-Laptop'
 when 'Notebook' then '01-Laptop'
 when 'Laptop' then '01-Laptop'
 when 'Desktop' then '02-Desktop'
 when 'Tower' then '02-Desktop'
 when 'Mini Tower' then '02-Desktop'
 else '99-'||(dmi->'Chassis_Information'->>'Type')
end as type_chassis,
string_agg(distinct coalesce(manufacturer,'?') ||' '|| coalesce(productname,''),', '),
count(*) as "Nb_Machines" from hosts
group by 1
```

- Lister les hotes avec leur numéro de série Windows

```
select computer_name,os_name,os_version,host_info->'windows_product_infos'->'product_key' as␣
↪windows_product_key from hosts order by 3,1
```

Requêtes WAPT

- Lister les paquets WAPT du serveur

```
select package,version,architecture,description,section,package_uuid,count(*)
from packages
group by 1,2,3,4,5,6
```

- Lister les hotes nécessitant une mise à jour

```
select
computer_fqdn, host_status, last_seen_on::date,h.wapt_status,string_agg(distinct lower(s.
↪package),' ')
from hosts h
left join hostpackagesstatus s on s.host_id=h.uuid and s.install_status != 'OK'
where (last_seen_on::date > (current_timestamp - interval '1 week')::date and host_status!='OK')
group by 1,2,3,4
```

Requêtes paquets

- Lister les paquets avec leur nombre d'installation

```
select package,version,architecture,description,section,package_uuid,count(*)
from hostpackagesstatus s
where section not in ('host','unit','group')
group by 1,2,3,4,5,6
```

Requêtes logicielles

- Agent WAPT Community

```
select h.uuid,h.computer_name,install_date::date,version,h.listening_timestamp::timestamp,name␣
↪from hostsoftwares s
left join hosts h on h.uuid=s.host_id
where
s.key='WAPT_is1'
AND (name ilike 'WAPT%%Community%%' OR name ilike 'WAPT %%')
```

- Lister les hotes avec la version de 7zip associée

```
select hosts.computer_name,hostsoftwares.host_id,hostsoftwares.name,hostsoftwares.version
from hosts,hostsoftwares
where hostsoftwares.name ilike '7-zip%%' and hosts.uuid=hostsoftwares.host_id
order by hosts.computer_name ASC
```

- Lister les hotes avec leur logiciels

```
select
n.normalized_name,s.version,string_agg(distinct lower(h.computer_name),' '),count(distinct h.
↪uuid)
from hostsoftwares s
left join normalization n on (n.original_name = s.name) and (n.key = s.key)
left join hosts h on h.uuid = s.host_id
where (n.normalized_name is not null) and (n.normalized_name<>'') and not n.windows_update and␣
↪not n.banned and (last_seen_on::date > (current_timestamp - interval '3 week')::date)
group by 1,2
```

- Normaliser les noms d'un logiciel

```
select
n.normalized_name,string_agg(distinct lower(h.computer_name),' '),count(distinct h.uuid)
from hostsoftwares s
left join normalization n on (n.original_name = s.name) and (n.key = s.key)
left join hosts h on h.uuid = s.host_id
where (n.normalized_name is not null) and (n.normalized_name<>'') and not n.windows_update and␣
↪not n.banned and (last_seen_on::date > (current_timestamp - interval '3 week')::date)
group by 1
```

Vous pouvez également trouver plusieurs autres exemples de requêtes sur le forum de Tranquil IT.

Feel free to post your own queries on the same forum with an explanation of what your query does, ideally with a screen capture or a table showing a sample of your query result.

Normaliser les noms d'un logiciel

Parfois, la version du logiciel ou son architecture font partie intégrante du nom du logiciel. Lorsqu'ils s'inscrivent dans l'inventaire du serveur WAPT, ils apparaissent comme des logiciels différents alors qu'ils ne sont qu'un seul logiciel pour nous, les humains.

Pour résoudre ce problème, nous proposons de standardiser le nom de certains logiciels.

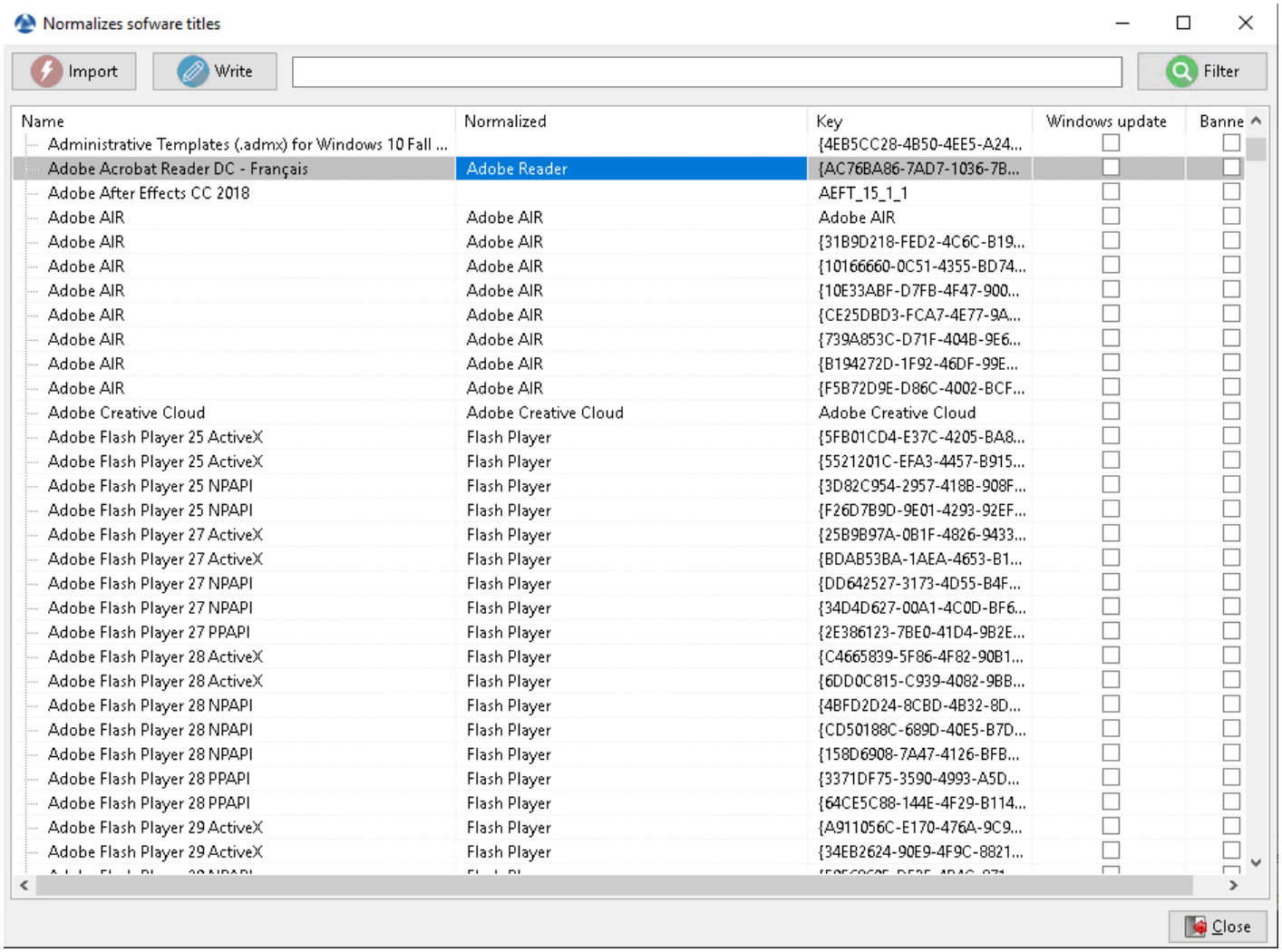


Figure130: Normaliser les noms d'un logiciel

- cliquez sur *Normaliser les noms de logiciels* dans le menu *Outils* ;
- sélectionner le logiciel à standardiser, par exemple, toutes les différentes versions d'Adobe Flash Player ;
- sur la colonne *normalized*, appuyez sur `F2` pour attribuer un nom standardisé au logiciel sélectionné. Puis appuyez sur `Entrée` ;

Note:

- pour sélectionner plusieurs programmes, sélectionnez-les avec la combinaison de touches `shift-up/down` ;

- vous pouvez aussi indiquer un logiciel comme *windows update* ou *banned* (Appuyez sur `Espace` dans la colonne correspondante) ;

- cliquez sur *Importer* pour charger les changements depuis le serveur ;
- cliquez sur *Ecrire* pour sauvegarder vos modifications ;

Vous pouvez maintenant exécuter vos requêtes avec ce nom standardisé.

Vidéo de démonstration

6.8.5 Utiliser le WAPT SelfService

Nouveau dans la version 1.7: Enterprise

Présentation

Avec WAPT 1.7, vous pouvez maintenant filtrer les paquets en libre-service disponibles pour vos utilisateurs.

Vos utilisateurs pourront installer une sélection de paquets WAPT sans nécessiter de droits *Administrateur Local*.

Les *Utilisateurs* gagnent en autonomie en déployant des logiciels et des configurations fiables et autorisées par l'*Organisation*. Il s'agit d'un gain de temps pour le service d'assistance informatique de l'Organisation.

Comment ça fonctionne ?

Avec WAPT 1.7 **Enterprise**, un nouveau type de paquet WAPT existe à côté des paquets *base*, *group*, *host*, *profile* et *unit* : ce sont les paquets **selfservice**.

Un package *self-service* peut maintenant être déployé sur les machines pour lister les différentes règles de self-service qui s'appliquent à la machine.

Comment utiliser la focntionnalité de *self-service* ?

Indication: La fonctionnalité **Self-Service** est uniquement disponible dans la version **WAPT Enterprise**.

In the **Community** version, only Local Administrators and members of the *waptselfservice* group can access self-service on the agent.

Dans la version **Community**, il n'est pas possible de filtrer les paquets rendus accessibles à l'utilisateur.

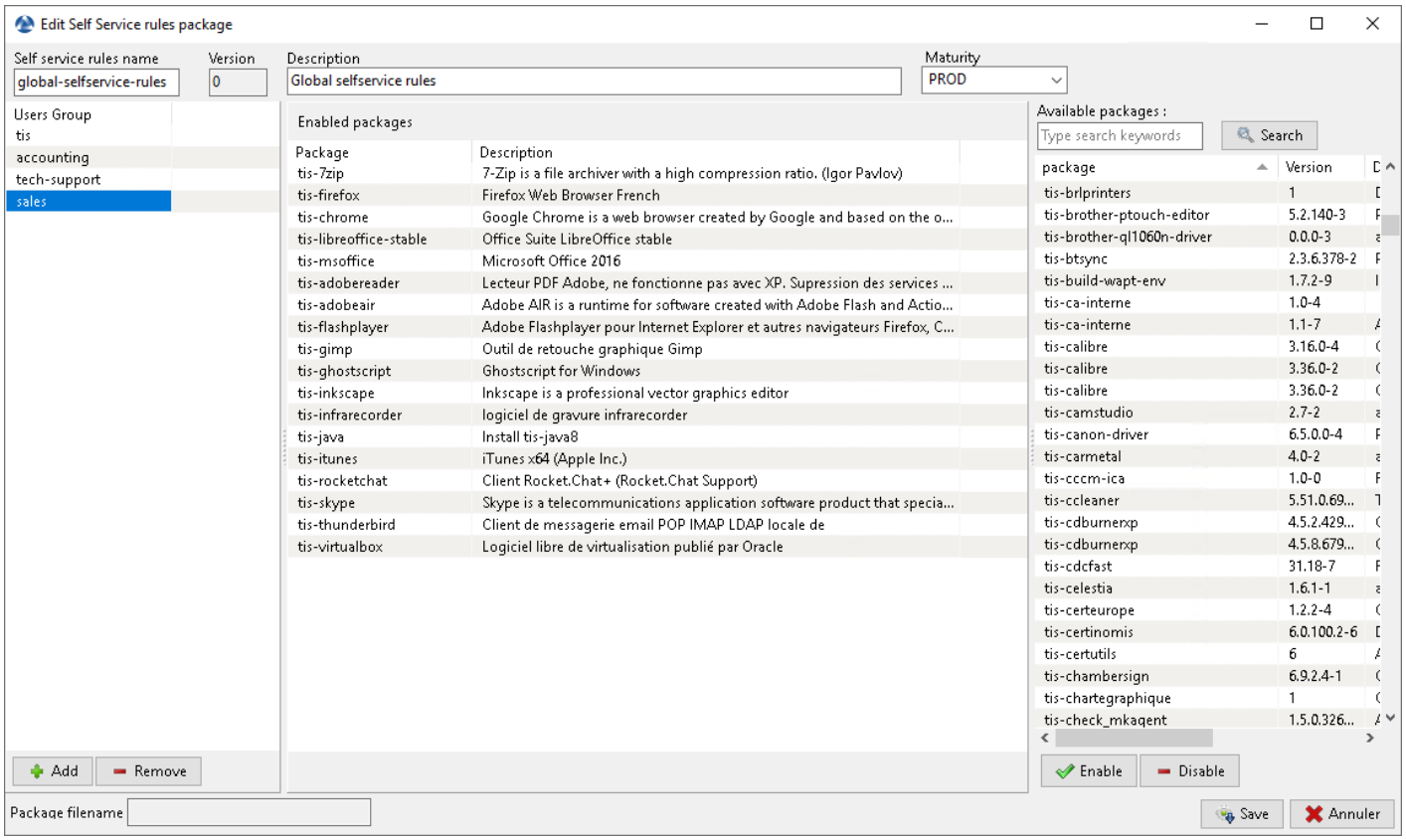


Figure131: Créer un paquet *self-service*

Dans la console, allez dans l'onglet *Règles de self-service*.

Vous pouvez maintenant créer votre premier paquet de règles *self-service*.

- donner un nom au nouveau paquet de *self-service* ;
- cliquer sur *Ajouter* pour ajouter un groupe Active Directory (en bas à gauche) ;
- nommer le groupe *self-service* (avec `F2` ou tapez directement dans la cellule) ;
- faire glisser les paquets logiciels et les paquets de configuration autorisés pour ce groupe *self-service* dans la colonne centrale ;
- ajouter autant de groupes que nécessaire dans le paquet ;
- enregistrer le paquet et le déployer sur une sélection de machines ;
- une fois le paquet déployé, seuls les paquets autorisés listés dans le(s) groupe(s) *self-service* dont l'*Utilisateur* est membre seront affichés pour l'*Utilisateur* actuellement connecté ;

Note:

- Si un groupe apparaît dans plusieurs paquets *selfservice*, alors les règles seront fusionnées ;
- l'authentification utilisée est l'authentification par le système des utilisateurs et des groupes locaux, mais si la machine se trouve dans un domaine, l'authentification et les groupes fonctionneront également avec les utilisateurs et les groupes du domaine ;

Comment utiliser la focntionnalité de *self-service* ?

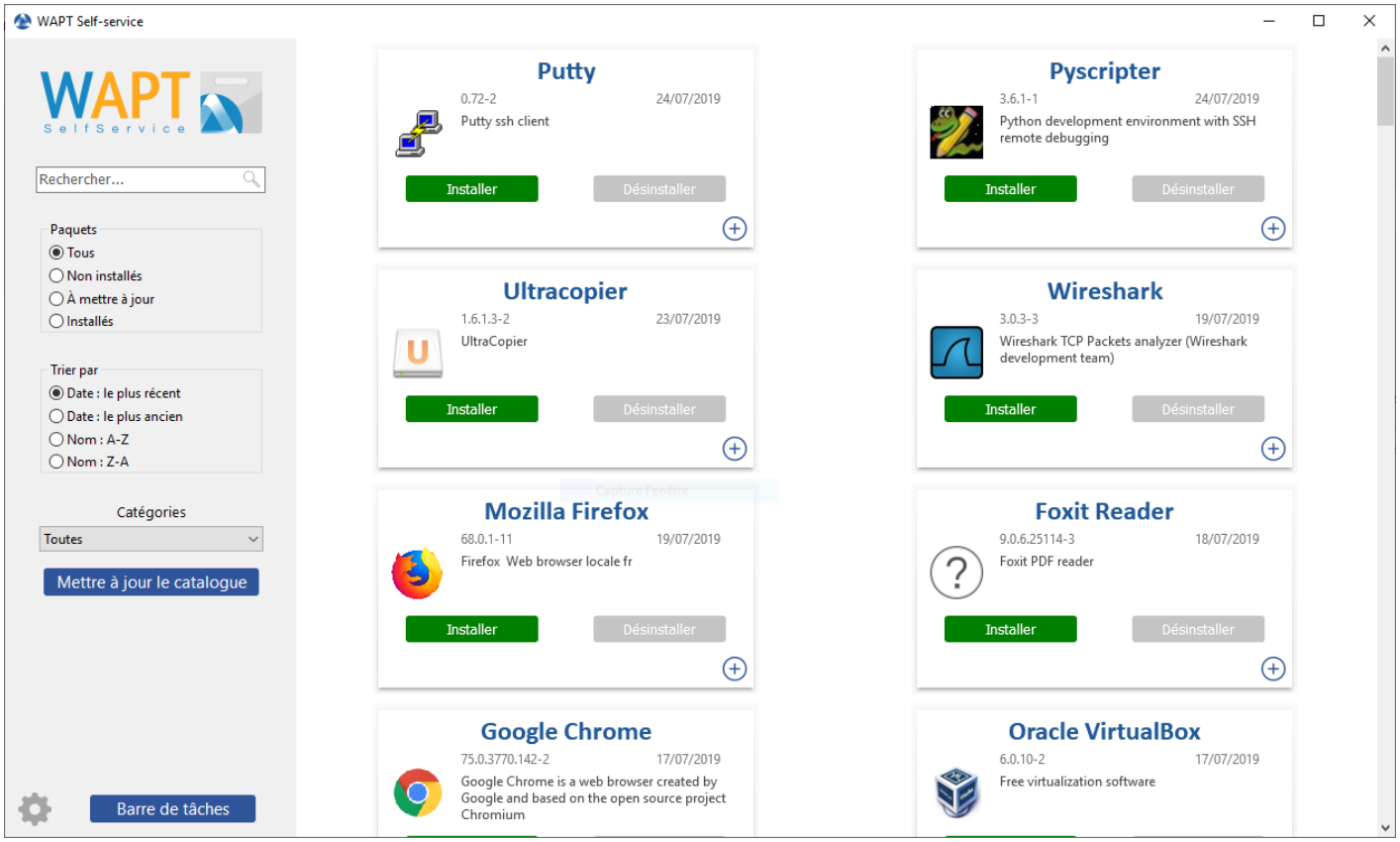
Le self service est accessible aux utilisateurs dans le menu démarrer sous le nom « Self service logiciels WAPT ».

Il est aussi disponible directement à `<base>\waptself.exe`

Les identifiants à entrer au démarrage du self service sont ceux de Windows (local ou active directory).

Le self service affiche ensuite une liste de paquets disponibles pour l'installation.

- L'utilisateur peut avoir plus de détails sur chaque paquets en cliquant sur le « + ».
- Différents filtres sont disponibles pour l'utilisateur sur le panneau de gauche.
- Le bouton « mettre à jour le catalogue » est utilisé pour forcer un « wapt-get update » sur l'agent WAPT.
- La liste des catégories de paquets est disponible pour l'utilisateur. Pour ajouter une catégorie à la liste, vous devez juste spécifier une nouvelle catégorie dans le fichier control de paquet souhaité. Précisément dans le champ « categories ».
- Les tâches actuellement en cours par l'agent wapt sont disponibles en cliquant sur le bouton « task bar ».
- Il est possible de changer le langage de l'interface en cliquant sur le bouton de configuration en bas à gauche de l'interface.


WAPT Self-service
WAPT
SelfService
Rechercher...
Paquets
Tous
Non installés
À mettre à jour
Installés
Trier par
Date : le plus récent
Date : le plus ancien
Nom : A-Z
Nom : Z-A
Catégories
Toutes
Mettre à jour le catalogue
Barre de tâches
Putty
0.72-2
24/07/2019
Putty ssh client
Installer
Désinstaller
Pyscripter
3.6.1-1
24/07/2019
Python development environment with SSH remote debugging
Installer
Désinstaller
Ultracopier
1.6.1.3-2
23/07/2019
UltraCopier
Installer
Désinstaller
Wireshark
3.0.3-3
19/07/2019
Wireshark TCP Packets analyzer (Wireshark development team)
Installer
Désinstaller
Mozilla Firefox
68.0.1-11
19/07/2019
Firefox Web browser locale fr
Installer
Désinstaller
Foxit Reader
9.0.6.25114-3
18/07/2019
Foxit PDF reader
Installer
Désinstaller
Google Chrome
75.0.3770.142-2
17/07/2019
Google Chrome is a web browser created by Google and based on the open source project Chromium
Oracle VirtualBox
6.0.10-2
17/07/2019
Free virtualization software

Personnaliser l'interface du libre-service

Ajouter le logo de votre organisation

Dans la version **Entreprise seulement de WAPT**, il est possible de changer le logo qui apparaît dans l'interface de libre-service et donc d'améliorer l'acceptation de la fonction de libre-service par vos utilisateurs.

Pour faire cela, placez simplement le logo que vous souhaitez à cet emplacement : `<wapt>\templates\waptself-logo.png`

Note: Il est hautement recommandé d'utiliser une image PNG de dimension 200*150px.

Gérer les catégories de paquets

Les catégories par défaut sont :

- Internet ;
- Utilities ;
- Messaging ;
- Security ;
- System and network ;
- Storage ;
- Media ;
- Development ;
- Office;

Vous pouvez créer vos propres catégories facilement en remplissant l'attribut `catégories` du fichier `control` de n'importe quel paquet WAPT et écrire une nouvelle catégorie de votre choix, WAPT affichera automatiquement le paquet dans la nouvelle catégorie.

Paramétrage de WAPT Self-Service dans l'agent WAPT

L'agent WAPT peut être configuré pour forcer le filtrage des paquets WAPT SelfService aux administrateurs locaux *Paramètres pour le libre-service WAPT et l'authentification Waptservice*.

Configurer une méthode d'authentification différente pour le libre-service

Comme mentionné ci-dessus, l'authentification sur le service WAPT est configurée par défaut en mode système.

Cela signifie que le service WAPT transmet l'authentification directement au système d'exploitation ; il récupère également les groupes en interrogeant directement le système d'exploitation.

Ce comportement est défini avec la valeur de `service_auth_type` dans `wapt-get.ini`. La valeur par défaut est *system*.

Dans ce mode, nous supposons que les administrateurs locaux peuvent voir tous les paquets. Pour changer ce comportement, modifiez la valeur de `waptservice_admin_filter` dans `wapt-get.ini`.

Vous pouvez consulter cet article décrivant les *réglages pour le libre-service WAPT et l'authentification Waptservice* pour plus d'options.

Deux modes supplémentaires sont disponibles à partir de la version 1.8.2 :

- « waptserver-ldap » : ce mode permet l'authentification au serveur WAPT. Le serveur WAPT effectuera une requête LDAP pour vérifier l'authentification et les groupes. **Attention** ! Pour que cela fonctionne, vous devez avoir configuré l'authentification LDAP sur le serveur WAPT, (la configuration du groupe d'administration sera ignorée) Voir *cet article sur la configuration de l'authentification par rapport à Active Directory* pour plus d'informations.
- « waptagent-ldap », ce mode permet l'authentification avec un serveur LDAP identifié dans le fichier `wapt-get.ini`. L'agent WAPT effectuera une requête LDAP pour vérifier l'authentification et les groupes.

  Vous pouvez consulter cet article décrivant les *réglages pour le libre-service WAPT et l'authentification Waptservice* pour plus d'options.

Note: Pour que l'authentification du système sous fonctionne correctement, assurez-vous de configurer correctement votre authentification pam et votre `nsswitch.conf`. La commande **`id username`** doit retourner la liste des groupes dont l'utilisateur est membre.

Vidéo de démonstration

Nouveau dans la version 1.5: Enterprise

6.8.6 Différencier les niveaux de rôle dans WAPT

Indication: Cette fonctionnalité est uniquement disponible avec la version **Enterprise**

Introduction

WAPT offre la possibilité de différencier les rôles des administrateurs en basant leurs identités sur une PKI pour signer les paquets et les actions.

Indication: La description suivante de la différenciation des rôles est temporaire car elle va évoluer dans un avenir proche.

Il y a trois cas :

Clé privée + types de certificats	Principaux usages
Clé privée simple + certificat	Permet l'authentification sur la console WAPT + les interactions avec les agents WAPT
Clé privée du développeur + certificat	Permet l'authentification sur la console WAPT + interactions avec les agents WAPT + signature de paquets WAPT
Clé privée de l'autorité de certification (CA) + certificat	Permet l'authentification + les interactions + la signature de paquets + la délivrance de nouvelles clés privées

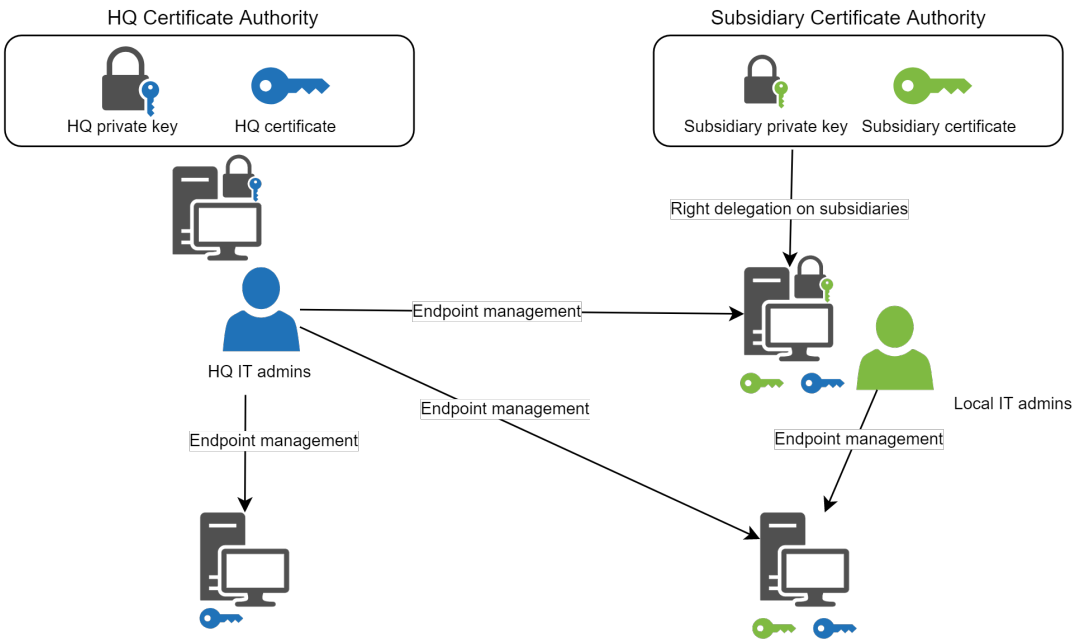


Figure132: Différenciation des rôles des utilisateurs administrateurs de WAPT

Une installation standard de WAPT générera une clé privée de CA par défaut, permettant l'émission de nouvelles clés privées pour les développeurs et les déployeurs de paquets WAPT.

Il est possible d'émettre une Autorité de Certification intermédiaire pour chaque filiale. Il est alors possible de délivrer une clé privée personnelle et son certificat correspondant à chaque administrateur informatique.

En examinant les schémas ci-dessus, nous pouvons en déduire la conclusion suivante :

- Les agents WAPT du siège de l'Organisation peuvent être gérés par l'équipe informatique du siège et ne peuvent pas être gérés par les équipes informatiques des filiales ;
- Les agents WAPT de la filiale ayant les deux certificats, celui du siège et celui de la filiale, peuvent être gérés par l'équipe informatique locale et par l'équipe informatique du siège ;

L'utilisation d'une PKI existante est possible, la console WAPT est livrée avec un simple générateur de certificats.

Générer un nouveau certificat

Générer l'Authorité de Certification (CA)

Lors de l'installation de WAPT, il vous est demandé de créer une paire `.pem` / `.crt` en cochant les cases *Certificat CA* et *Code Signing*.

Ce couple crt / pem permettra de signer des paquets WAPT et de nouveaux certificats.

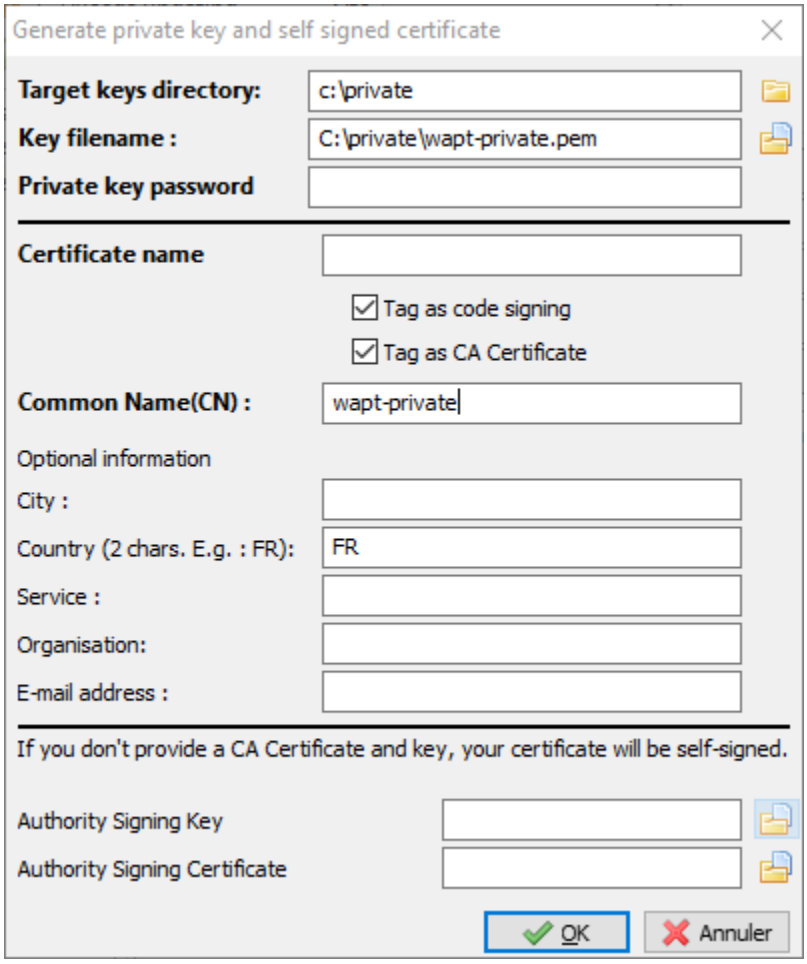


Figure133: Générer un certificat auto-signé

Générer un nouveau certificat avec la CA

Pour générer un nouveau couple crt / pem à partir de la clé privée, cliquer sur *Créer un certificat*.

Note: Le nouveau certificat ne sera pas un certificat auto-signé ;

Ce nouveau certificat sera signé par la CA (la clé générée lors de la première installation) ;

Vous devez donc renseigner le *Certificat de la CA`** et la :guilabel:`Clé de la CA*.

Lors de la génération de ce nouveau couple pem / crt vous pouvez choisir si le nouveau certificat sera de type **Code Signing** ou non.

Indication: Pour rappel, un certificat *Code Signing* est destiné dans le contexte de WAPT aux personnes avec le rôle *Administrateur* , alors qu'un certificat SSL non Code Signing est destiné aux personnes ayant le rôle de *Déployeur de Paquets* .

Les *Administrateurs* pourront signer des paquets qui **CONTIENNENT** un fichier `setup.py` exécutable (c.à.d. des paquets logiciels).

Les individus avec le rôle de *Déployeur de Paquets* pourront signer des paquets qui **NE CONTIENNENT PAS** de fichier `setup.py` (c.à.d. des paquets de type *machine*, *unit* et groupe*).

Les clés et certificats **Non Code Signing** pourront donc être confiées aux personnes en charge du déploiment des paquets sur le parc machine.

Une autre équipe disposant de certificats pourvus de l'attribut **Code Signing** pourra préparer les paquets WAPT contenant des applications configurées en accord avec la politique de sécurité de l'*Organisation* et les personnalisations souhaitées par elle.

Générer un nouveau couple pem / crt permettra également d'authentifier formellement le signataire d'un paquet avec le champ CN.

Indication: Les nouveaux certificats ne seront pas **Certificat CA**, ce qui signifie qu'il ne pourront pas contre-signer d'autres certificats.

En règle générale, vous n'avez qu'un seul couple crt / pem de type `Certificat CA` par *Organisation*.

Déployer les certificats d'administrateurs locaux sur les clients

Certaines organisations choisiront de laisser les administrateurs informatiques locaux effectuer des actions sur les appareils équipés de WAPT en leur délivrant des certificats personnels qui fonctionneront sur l'ensemble des appareils dont les administrateurs informatiques locaux sont responsables.

Les administrateurs informatiques du siège déploieront les certificats des administrateurs informatiques locaux sur les clients de leurs sites respectifs.

De cette façon, les administrateurs informatiques locaux ne pourront pas gérer les ordinateurs situés au siège, mais uniquement les ordinateurs situés sur leurs propres sites.

Copiez les certificats des administrateurs informatiques locaux autorisés sur les clients dans `C:\program files(x86)\wapt\ssl`.

Indication: N'oubliez pas de redémarrer le service WAPT sur les clients pour qu'ils puissent utiliser leur nouveau certificat. Ouvrez une ligne de commande **`cmd.exe`**:

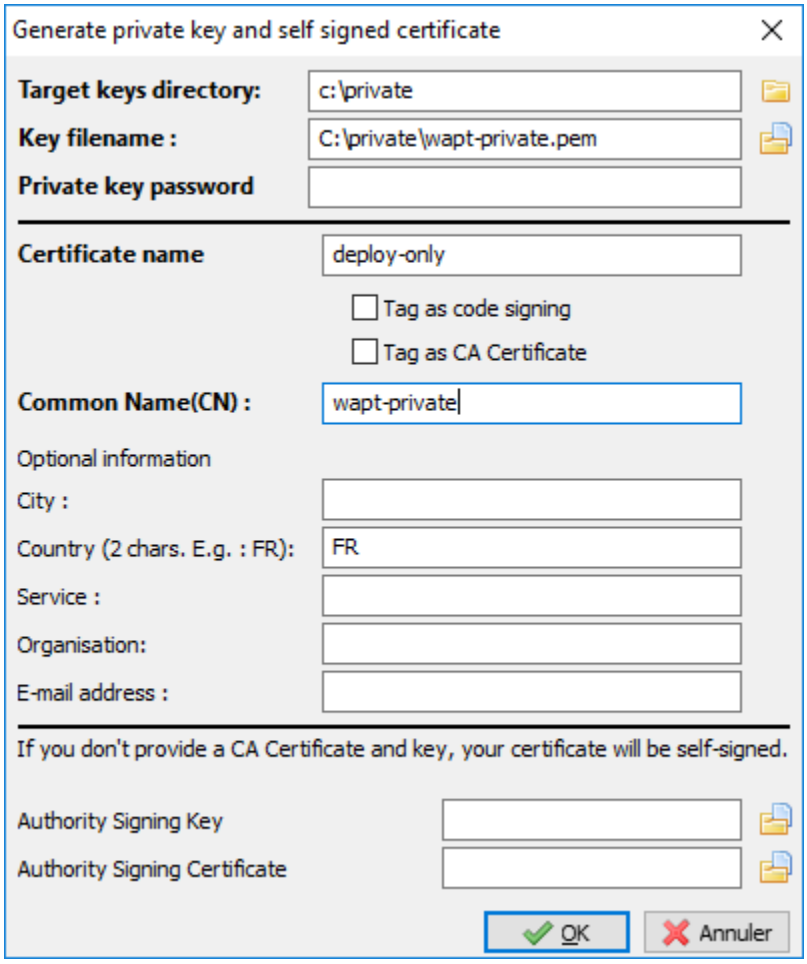


Figure134: Génération d'un certificat sans attribut Code Signing

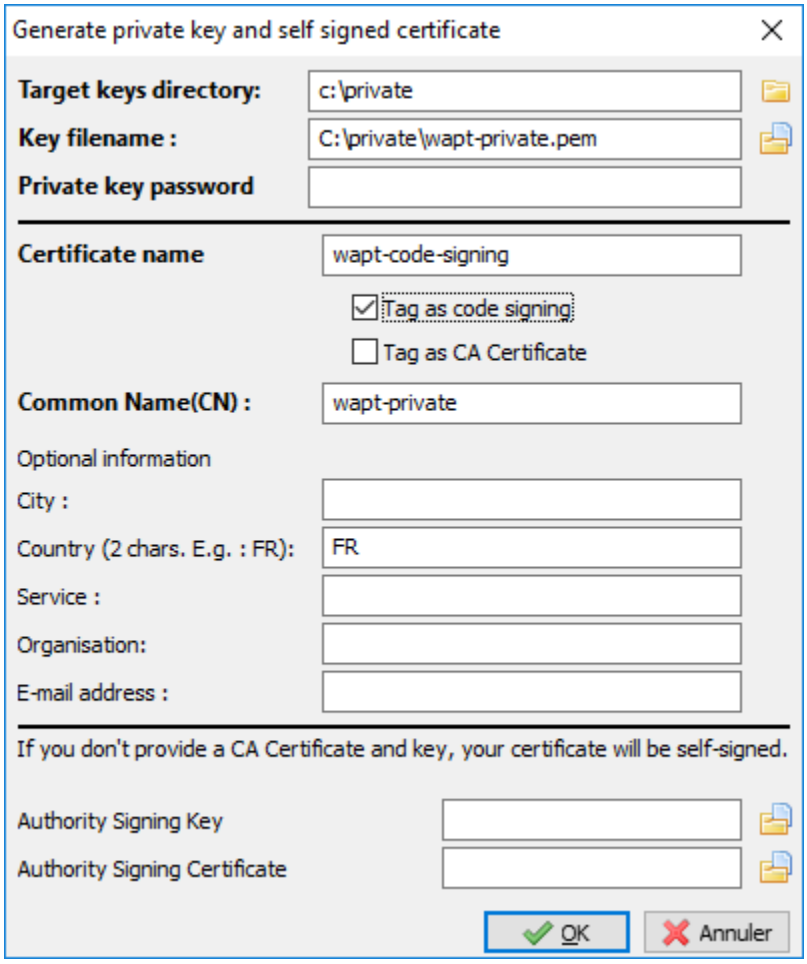


Figure135: Génération d'un certificat Code Signing

```
net stop waptservice
net start waptservice
```

Si vous souhaitez déployer les certificats en utilisant WAPT, vous trouverez ci-dessous un exemple de package permettant de déployer les certificats sur les ordinateurs clients.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
  print(ur"Copy of AC's distant site")
  filecopyto('ca_distant.crt',makepath(install_location('WAPT_is1'),'ssl',))

def audit():
  print('Auditing %s' % control.asrequirement())
  return "OK"

if __name__ == '__main__':
  update_package()
```

6.9 Mettre à jour le serveur WAPT

Indication: Si votre serveur wapt est installé en machine virtuelle, faites un snapshot. Cela vous permettera de revenir facilement en arrière en cas d'échec de la mise à jour.

Avant de mettre à niveau le serveur WAPT, veuillez consulter le tableau de compatibilité de mise à niveau suivant :

	Vers WAPT 1.5	Vers WAPT 1.6	Vers WAPT 1.7	Vers WAPT 1.8
Depuis WAPT 1.3	Oui	Oui	Non	Non
Depuis WAPT 1.5	•	Oui	Oui	Non
Depuis WAPT 1.6	•	•	Oui	Non
Depuis WAPT 1.7	•	•	•	Oui

6.9.1 Mettre à jour de 1.6/1.7 vers 1.8

La mise à jour s'effectue comme une mise à jour mineure :

- *mise à jour mineure pour Debian*;
- *mise à jour mineure pour CentOS*;
- *mise à jour mineure pour Windows*;

Attention:

- Debian Jessie est maintenant obsolète. **WAPT 1.8 ne fonctionnera pas avec cette ancienne version de Debian** ;
- envisagez de migrer votre installation WAPT existante vers *Debian Buster ou CentOS7*;

6.9.2 Mettre à jour de 1.5 vers 1.6

La mise à jour s'effectue comme une mise à jour mineure.

Note:

- si vous êtes sur Debian Jessie, il est recommandé de mettre à niveau vers Debian Buster 64 bits : *Mettre à jour le système d'exploitation* ;
- ceci est **OBLIGATOIRE** pour la version **Enterprise** avec prise en charge des Windows Updates ;
- lors de la mise à jour vers Debian10, la base de données PostgreSQL doit être mise à jour ;

6.9.3 Mettre à jour de 1.3 vers 1.6

La mise à jour de la version 1.3 vers 1.6 est une mise à jour majeure.

Changements entre 1.3 et 1.5/1.6 et conséquences

PostgreSQL a remplacé MongoDB dans WAPT

La base de données MongoDB a été remplacée par PostgreSQL avec le support JSON.

La conséquence est que **ceux et celles qui avaient développé du reporting s'appuyant sur MongoDB devront adapter leurs requêtes**.

Nginx est le seul serveur web aujourd'hui supporté dans WAPT

L'introduction des Websockets dans WAPT apporte de nombreux bénéfices :

- la remontée des changements d'état des clients dans la console est instantanée ;
- quand l'*Administrateur* force une mise à jour à effet immédiat, ce n'est plus le serveur qui établit une connexion avec l'agent ; l'agent WAPT maintient une connexion permanente avec le serveur ;
- les Websockets évitent de maintenir ouvert un port en écoute sur le poste, améliorant ainsi la sécurité du dispositif ;
- puisque l'agent WAPT initie et maintient une connexion avec le serveur, nous pouvons franchir plus facilement les proxys et les pares-feu ;

`Nginx` est le seul serveur web capable de gérer une grande quantité de Websockets. Cette technologie n'est pas quelque chose de trivial à implémenter et à maintenir.

En conséquence, **le support IIS et Apache dans WAPT est abandonné**.

Les hashs pour les signatures sont en sha256 au lieu de sha1

Les sommes de contrôle pour la liste des fichiers du paquet et la signature des attributs du fichier control sont maintenant des hashs sha256 au lieu de sha1 pour satisfaire les recommandations des organismes de sécurité.

En conséquence, **nous devons resigner tous les paquets du dépôt et regénérer l'index :file:`Packages` du dépôt**.

Un script permet de resigner en masse tous les paquets sur le serveur.

Les paquets machines sont nommés avec l'UUID du poste

Dans les versions <= 1.3.xx, les paquets machines sont nommés avec le nom complet du poste client.

Cela pose des problèmes lorsque la machine change de domaine, ou que la machine n'est pas dans un domaine mais change de réseau avec des noms de réseau DHCP différents.

Dans les versions >= 1.5, le paquet machine est nommé avec l'UUID du poste client, résolvant les problèmes décrit ci-avant.

En conséquence, **il faut recréer et res-signer les paquets machines**.

Un script sur le serveur permet de faire cette opération automatiquement.

Attribut Code Signing pour les certificats des paquets logiciels

Pour différencier les rôles de *Déployeur de Paquets* de celui de *Développeur de Paquets*, le client WAPT vérifie l'attribut **Code Signing** du certificat qui a signé le paquet.

Si le paquet contient du code python, c'est-à-dire qu'un fichier `setup.py` est présent, alors il doit être signé avec un certificat **Code Signing**. Sinon, il ne s'installera pas.

En conséquence, **il faut recréer et déployer un certificat code signing sur le parc**, et resigner les paquets logiciels avec un certificat de type **Code Signing**.

Le certificat est déployé lors de la mise à jour de l'agent WAPT.

Mettre à jour WAPT de 1.3 vers 1.6

Deux méthode sont à votre disposition :

- La *mise à jour minimale* qui consiste à ne migrer que les paquets logiciel sans conserver l'inventaire ;
- La *mise à jour complète* qui permet de migrer entièrement toutes les données de l'ancien serveur ;

Attention: Quelque soit votre choix, deux prérequis sont nécessaire :

- Pour que la mise à jour fonctionne correctement, les postes de votre parc doivent **tous** être en 1.3.13 avant de passer en version 1.6, dans le cas contraire, le paquet **`waptupgrade`** ne fonctionnera pas correctement. A défaut, il vous reste la possibilité de faire une upgrade à travers le déploiement par GPO ;
- Les machines à mettre à jour doivent toutes avoir également installé le paquet *waptupgrade* ;

Indication: Des forfaits de mise à jour WAPT 1.3 vers WAPT 1.6 sont disponibles auprès de notre équipe commerciale au +33 (0) 240 975 755.

Mettre à jour à minima

Note: Cette procédure permet également la bascule facile d'un serveur WAPT Windows à Linux ou de Linux à Windows.

Cette procédure est la procédure la plus simple.

La migration à minima consiste à ne migrer que les paquets WAPT logiciels et groupes sans conserver le contenu de la base de données ni les paquets machines.

- toutes les informations de la base de données viennent des postes, on ne perd aucune information d'inventaire ;
- la base de données d'inventaire se reconstruira au fur et à mesure que les postes redémarreront avec l'agent WAPT 1.6 et se ré-enregistreront ;

Migrer plus simplement de WAPT 1.3 à 1.6

Cette procédure est plus simple qu'une migration complète, mais certaines données non essentielles peuvent être perdues, cassant la chaîne de traçabilité.

- sauvegarder les paquets WAPT présents dans :
  - Linux : `/var/www/wapt/` ou `/var/www/html/wapt/` ;
  - Windows : `C:\wapt\waptserver\repository\wapt` ;

Installer WAPT et réimporter les paquets

Si vous venez d'une version Windows 1.3.13 de WAPT, vous devez déinstaller la version 1.3.13 avant d'installer la version 1.6.

Vous pouvez maintenant suivre la procédure d'installation normale pour *installer un serveur WAPT*.

Le serveur WAPT doit répondre au même nom DNS que l'ancienne installation (avoir la même IP si vos agents fonctionnent par IP).

Indication: Si vous voulez que le paquet **waptupgrade fonctionne**, il vous faudra suivre la procédure pour *regénérer les clés*.

Arrêter vous à la section **Re-signer les paquets du dépôt**.

Une fois l'installation terminée, vous pouvez ré-importer tous les paquets WAPT sauvegardés avec *Importer depuis un fichiers* dans l'onglet *Dépôt Privé*.

Cette opération va re-signer tous les paquets.

Installer le nouvel agent WAPT 1.6 sur les postes

- relancer maintenant la *create_WAPT_agent* ce qui uploadera automatiquement un paquet **waptupgrade** ;
- puis vous pouvez mettre à jour la Stratégie de Groupe de Déploiement de l'agent WAPT en suivant la procédure pour *déployer les agents WAPT* ;
- le paquet **`waptupgrade`** s'installera ensuite automatiquement sur les poste qui l'avait déjà et l'agent WAPT se mettra à jour automatiquement à l'arrêt du poste ;

Attention:

- il faudra réattribuer les logiciels aux machines (pas de panique, cela ne désinstallera pas les paquets WAPT déjà installés) ;
  - cette procédure fait perdre temporairement l'affichage des machines dans la console ;
  - cette procédure ne permet pas la mise à jour avec le paquet **`waptupgrade`** ;

Mise à jour majeure complète

Cette procédure est la procédure la plus complète car elle permet de migrer la base de donnée et les paquet WAPT machine, groupe et logiciels.

Mettre à jour le serveur WAPT 1.3 vers une version récente Linux x64

Sous Linux, vous devez sauvegarder toute votre base d'inventaire MongoDB avec la commande `mongodump --db wapt --archive=/root/wapt.dump`.

```
mongodump
```

Indication: Si la commande mongodump n'est pas disponible, vous pouvez l'installer avec la commande `apt install mongo-tools`.

Lorsque la commande **mongodump** est lancée, un fichier `dump` est créé, vous devez le sauvegarder.

Vous devez également sauvegarder vos paquets WAPT machine et logiciel. Ils sont stockés dans `/var/www/wapt/` et `/var/www/wapt-host/`.

Lorsque vous allez installer votre nouveau serveur WAPT, vous devrez ré-installer une version 1.3.13 de WAPT avant de faire l'upgrade en version 1.6 :

https://www.wapt.fr/fr/doc-1.3/Installation/waptserver/linux/index.html

Vous allez ensuite pouvoir restaurer les paquets WAPT sauvegardés précédemment dans les dossiers `/var/www/wapt/` et `/var/www/wapt-host/` de votre nouveau serveur.

Pour restaurer la base MongoDB créée précédemment vous devez exécuter la commande `mongorestore /root/dump`.

Vous pouvez maintenant suivre la procédure classique de mise à jour de WAPT 1.3.13 vers 1.6 :

- pour *Debian Linux* ;
- pour *CentOS / RedHat* ;

Mettre à jour WAPT de 1.3 vers 1.6 avec Debian

Prélude

Note: Nous supposons que votre serveur WAPT est installé sur un système Debian 8 minimum (x64). Si ce n'est pas le cas, vous pouvez suivre la documentation de *mise à jour*.

Cette procédure vise exclusivement la migration de WAPT 1.3 vers WAPT 1.6.

Elément	WAPT 1.3	WAPT 1.6
Base de données	**MongoDB**	**PostgreSQL**
Serveur Web	**Apache2**	**Nginx**
Agent WAPT	l'agent écoute en local sur le port 8088	l'agent initie et maintient une *websocket* avec le serveur.
Signature	Hashes sha1	un certificat *Code Signing* est désormais requis ; les attributs du fichier `control` sont signés en sha256.

Toutes ces différences induisent les manipulations décrites ci-après à scrupuleusement effectuer dans l'ordre.

Installer systemd et ca-certificates

- lancer l'installation de systemd :

```
apt install systemd
```

- installer ca-certificates :

```
apt install ca-certificates
```

- redémarrer le serveur WAPT :

```
reboot
```

Désinstaller WAPT 1.3 du serveur Debian

```
apt remove tis-waptrepo tis-waptsetup tis-waptserver
systemctl stop apache2
systemctl disable apache2
```

Configurer le serveur Debian

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo  "deb  https://wapt.tranquil.it/debian/wapt-1.6/ $(lsb_release -c -s) main"  > /etc/apt/
↪sources.list.d/wapt.list
apt update
```

Installer la nouvelle version sur le serveur Debian

```
apt install tis-waptserver tis-waptsetup
```

Note: L'installation peut vous demander le royaume Kerberos. Vous pouvez ignorer, appuyer sur `Entrer` pour passer l'étape.

Lancer le script de post-configuration

Note:

- il est recommandé de lancer la post-configuration après chaque mise à jour pour que le serveur utilise le format de configuration le plus sûr ;
- il n'est pas obligatoire de redéfinir un mot de passe pour la console WAPT lors de la post-configuration ;

```
/opt/wapt/waptserver/scripts/postconf.sh
```

Le postconf va vous proposer de changer le mot de passe ou de poursuivre, vous pouvez change le mot de passe si vous le souhaitez.

Le postconf va ensuite détecter que vous venez d'une version 1.3 et va tenter de lancer une migration de la base de donnée MongoDB vers PostgreSQL. Acceptez cette migration.

Le postconf va ensuite vous proposer de configurer **`Nginx`**, vous pouvez accepter.

Démarrer le serveur WAPT sur le serveur Debian

```
systemctl enable waptserver
systemctl start waptserver
```

Nettoyer le serveur Debian

A l'issue de l'installation, il faut absolument nettoyer le serveur WAPT.

En effet, WAPT utilise désormais **Nginx** pour son serveur Web et **PostgreSQL** pour la base de donnée d'inventaire.

```
apt remove apache2 mongodb
apt autoremove
apt clean
```

Installer la nouvelle console WAPT

- télécharger **waptsetup** : https://srvwapt.mydomain.lan/wapt/waptsetup-tis.exe ;
- lancer l'installation ; la configuration des url du repo et du serveur n'a pas changé ;
- C:\Program Files (x86)\wapt\
- vérifier que le serveur WAPT fonctionne correctement en cliquant sur les petites *clé à molette* et en cliquant sur le bouton *Vérifier* ;

Vous pouvez maintenant passer à la suite pour *générer les clés de signature* !!

Mettre à jour WAPT de 1.3 vers 1.6 avec Centos / RedHat

Note: Nous supposons que votre serveur WAPT est installé sur un système Centos 7 minimum (x64). Si ce n'est pas le cas, vous pouvez suivre la documentation de *mise à jour*.

Cette procédure vise exclusivement la migration de WAPT 1.3 vers WAPT 1.6.

Les principales différences sont :

Elément	WAPT 1.3	WAPT 1.6
Base de données	**MongoDB**	**PostgreSQL**
Serveur Web	**Apache2**	**Nginx**
Agent WAPT	l'agent écoute en local sur le port 8088	l'agent initie et maintient une *websocket* avec le serveur.
Signature	Hashes sha1	un certificat *Code Signing* est désormais requis ; les attributs du fichier `control` sont signés en sha256.

Toutes ces différences induisent les manipulations décrites ci-après à scrupuleusement effectuer dans l'ordre.

Désinstaller WAPT 1.3 du serveur CentOS / RedHat

```
yum remove tis-waptrepo tis-waptsetup tis-waptserver
systemctl stop httpd
systemctl disable httpd
```

Configurer le serveur CentOS / RedHat

```
localectl set-locale LANG=en_US.utf8
localectl status
yum update
yum install epel-release wget sudo unzip
wget https://wapt.tranquil.it/tools/mongo-tools_centos7_2.6.zip -O /tmp/mongo.zip
unzip -j mongo.zip -d /bin/
```

Mettre à jour le serveur CentOS / RedHat

```
cat > /etc/yum.repos.d/wapt.repo <<
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/centos7/wapt-1.6/
enabled=1
gpgcheck=0
EOL

yum install postgresql96-server postgresql96-contrib
```

Installer WAPT 1.6 sur le serveur CentOS / RedHat

```
yum install tis-waptserver
sudo /usr/pgsql-9.6/bin/postgresql96-setup initdb
sudo systemctl enable postgresql-9.6 waptserver nginx
sudo systemctl start postgresql-9.6 nginx
```

Note: L'installation peut vous demander le royaume Kerberos. Vous pouvez ignorer, appuyer sur `Entrer` pour passer l'étape.

Lancer le script de post-configuration

Note:

- il est recommandé de lancer la post-configuration après chaque mise à jour pour que le serveur utilise le format de configuration le plus sûr ;
- il n'est pas obligatoire de redéfinir un mot de passe pour la console WAPT lors de la post-configuration ;

```
/opt/wapt/waptserver/scripts/postconf.sh
```

Le postconf va vous proposer de changer le mot de passe ou de poursuivre, vous pouvez change le mot de passe si vous le souhaitez.

Le postconf va ensuite détecter que vous venez d'une version 1.3 et va tenter de lancer une migration de la base de donnée MongoDB vers PostgreSQL. Acceptez cette migration.

Le postconf va ensuite vous proposer de configurer **Nginx**, vous pouvez accepter.

Démarrer le serveur WAPT sur le serveur CentOS / RedHat

```
systemctl enable waptserver
systemctl start waptserver
```

Nettoyer le serveur CentOS / RedHat

A l'issue de l'installation, il faut absolument nettoyer le serveur WAPT.

En effet, WAPT utilise désormais **Nginx** pour son serveur Web et **PostgreSQL** pour la base de donnée d'inventaire.

```
yum remove httpd mongodb
```

Installer la nouvelle console WAPT

- télécharger **waptsetup** : https://srvwapt.mydomain.lan/wapt/waptsetup-tis.exe ;
- lancer l'installation ; la configuration des url du dépôt et du serveur n'a pas changé ;
- C:\Program Files (x86)\wapt\
- vérifier que le server WAPT fonctionne correctement en cliquant sur les petites *clé à molette* et en cliquant sur le bouton *Vérifier* !

Vous pouvez maintenant passer à la suite pour *générer les clés de signature* !!

Mettre à jour WAPT de 1.3 à 1.6 pour Windows

Attention: WAPT Serveur n'est plus installable sur une version x86 de Windows.

- télécharger la dernière version de **waptserversetup** 1.6 depuis https://wapt.tranquil.it/wapt/releases/latest/;
- lancer l'installation par dessus l'installation 1.3 déjà existante en suivant *cette documentation* ;

Note: A la fin de la post configuration de waptserver la configuration va détécter que vous venez d'un version 1.3 de WAPT et va vous proposer de lancer la migration de MongoDB à PostgreSQL.

- cliquer sur *Oui* ;
- lancer la console WAPT

Note: Si vous venez d'une version WAPT 1.3.13 installée avec IIS, le port d'écoute de WAPT doit être changé. Vous devez dans ce cas suivre la documentation *pour changer le port d'écoute*.

Vous pouvez maintenant passer à l'étape suivante pour *générer les clés de signature* !!

Passer WAPT 1.3 de Windows à Linux

Nous pouvons sinon vous proposer de passer à la version linux de **waptserver**.

- télécharger MongoDB pour Windows https://www.mongodb.com/download-center/community;
- extraire le zip puis exécuter **mongodump** ;

Note: Un dossier `dump` a été créé dans le même repertoire que le fichier `mongodump.exe`.

- sauvegardez tout le repertoire `C:\wapt` de votre server WAPT ;
- sauvegardez le dossier `C:\private` ;
- installez ensuite une version **1.3.13** de WAPT sous linux (Debian 8 x64) ou CentOS7/ RedHat7 (x64);

Indication: Pour installer une Debian 10 (Buster) neuve (machine physique ou virtuelle) sans interface graphique, référez vous à la documentation officielle : Debian GNU/Linux Installation Guide pour Strech.

- si les agents WAPT pointent sur une adresse IP, alors le nouveau serveur doit avoir la même adresse IP que l'ancien serveur WAPT sous windows ;
- si vos agents WAPT pointent vers un enregistrement *DNS*, alors vous pouvez modifier le champ dns *srvwapt* en indiquant la nouvelle adresse IP du serveur WAPT linux.
- mettre à jour les sources de téléchargement ;

```
apt update && apt upgrade -y
```

- installer le serveur WAPT ;

Note: Les paquets **tis-waptserver**, **tis-waptsetup** et **tis-waptrepo** sont signés et il est nécessaire de récupérer la clef gpg ci-dessous pour éviter les messages d'alerte lors de l'installation.

```
apt install apt-transport-https lsb-release systemd-sysv systemd
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo  "deb  https://wapt.tranquil.it/debian/wapt-1.3/ $(lsb_release -c -s) main"  > /etc/apt/
↪sources.list.d/wapt.list
apt update
apt install tis-waptserver tis-waptrepo tis-waptsetup
```

- lancer le script de configuration ;

```
/opt/wapt/waptserver/scripts/postconf.sh
```

Note: Le mot de passe demandé en étape #4 est utilisé pour accéder à la console WAPT.

- configurer le serveur WAPT ;
- démarrer le serveur WAPT ;

```
systemctl start waptserver
```

- restaurer les paquets WAPT de la machine windows sur le serveur Linux ;
  - uploader le contenu de `C:\wapt\waptserver\repository\wapt` dans `/var/www/wapt/` ;
  - uploader le contenu de `C:\wapt\waptserver\repository\wapt-host` dans `/var/www/wapt-host/` ;

    Indication: Vous pouvez charger le fichier sur le server linux avec le logiciel **winscp**.

  - ensuite, changer le propriétaire des fichiers à wapt :

    ```
    chown wapt:www-data /var/www/wapt*
    ```

- restaurer la base de données MongoDB sur le server linux :
  - uploader maintenant le dossier `dump` dans `/root/` avec WinSCP ou équivalent ;
  - restaurer la base MongoDB windows sur la base MongoDB linux :

    ```
    mongorestore /root/dump
    ```

Votre serveur WAPT fonctionne à nouveau en version 1.3.13 sous Linux.

Vous pouvez maintenant installer votre **waptagent** sur votre PC d'*Administrateur*, et restaurer le dossier `C:\private`.

Attention: Vous ne devez pas regénérer de clé privée, vous devez redonner le chemin vers votre clé privée dans les paramètres de la console. Vous devez également réindiquer le préfix.

Vous pouvez maintenant suivre la procédure classique de mise à jour de WAPT 1.3.13 vers 1.6 !!

Mettre à jour la console WAPT et les agents WAPT

Définir un mot de passe sur la clé privée

La console a perdu le chemin vers sa clé privée, c'est normal dans la version 1.3, la console pointait vers la clé privée. Depuis la version 1.5, la console pointe vers le certificat associé à la clé.

Rendez-vous donc dans *Outils* → *Préférences* → *Chemin du certificat personnel* et sélectionner le certificat issue de votre clé privée existante :

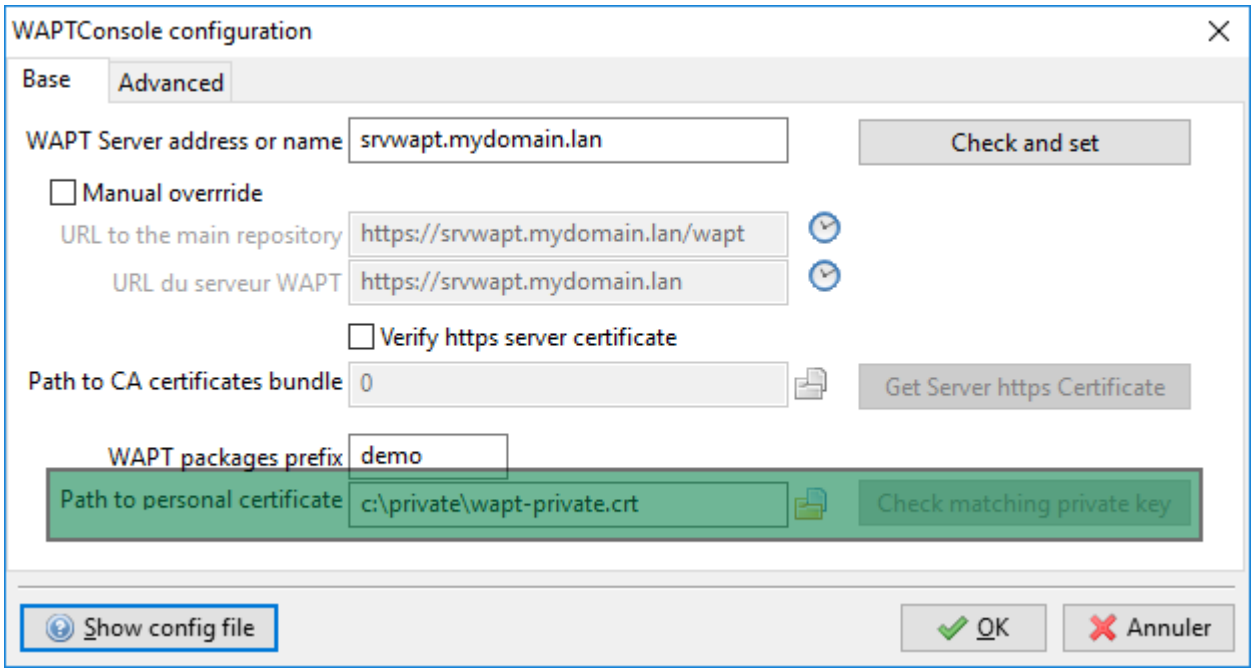


Figure136: Sélectionner le certificate de la clé privée

WAPT 1.6 nécessite un mot de passe sur la clé privée utilisée pour la signature des paquets et des actions.

Dans la console, aller dans *Outils* → *Changer le mot de passe de la clé privée* puis sélectionnez votre certificat et choisissez un mot de passe pour la clé privée.

Générer un certificat *Code Signing*

WAPT 1.6 différencie maintenant les certificats **Code Signing** des certificats **SSL** simples.

Il faut donc regénérer un nouveau certificat **Code Signing**.

Par défaut, WAPT 1.6 Community Edition génère des certificats Code Signing autosignés.

Vous devez vous assurer que l'ancien certificat `C:privatewapt-private.crt` et la clé `C:privatewapt-private.pem` sont présents dans `C:private`.

Attention: Pour permettre l'installation des paquets sur les ordinateurs de l'"*Organisation*, il faudra donc resigner tous les paquets en utilisant un certificat *Code Signing*.

Pour créer ce certificat, dans la console WAPT aller dans *Outils →Générer un Certificat*.

Note: Normalement, vous n'avez pas à recréer une nouvelle clé. Seul le certificat sera changé. Dans chemin de la clé, **vous devez indiquer** votre clé `C:privatewapt-private.pem`.

Attention: Ne modifiez pas les informations du certificat. Si cette information est modifée, la mise à jour 1.3.13 vers 1.6 ne pourra pas être faite avec WAPT et le paquet **`waptupgrade`** ne fonctionnera pas !

La console vous demande maintenant si ce nouveau certificat doit être ajouté dans le dossier `ssl` de l'agent WAPT, vous pouvez accepter.

Maintenant que vous avez généré votre nouvelle clé, vous devez à nouveau modifier le chemin vers le certificat personnel. La console doit maintenant pointer vers ce nouveau certificat.

Rendez-vous donc dans *Outils → Préférences → Chemin du certificat personnel* et sélectionner le certificat issue de votre clé privée existante :

Vous pouvez maintenant supprimer l'ancien certificat.

Re-signer les paquets du dépôt

Les paquets 1.3.13 doivent être re-signés avec la nouvelle procédure 1.6 qui intègre le certificat *Code-Signing* dans le paquet et utilise l'algorithme de hachage sha256.

Linux

Transférer temporairement votre clé privée (`.pem`) et le certificat **Code-Signing** (`.crt`) sur le serveur WAPT en utilisant **`WinSCP`** ou un utilitaire équivalent.

Se connecter en SSH sur le serveur, re-signer les paquets *base* en précisant le chemin du certificat :

```
PYTHONPATH=/opt/wapt PYTHONHOME=/opt/wapt python /opt/wapt/wapt-signpackages.py -i -s --message-
↪digest=sha256,sha1 -c /root/wapt-private-20180312-1522.crt /var/www/wapt/*.wapt
```

Indication: Le mot de passe *SuperAdmin* du serveur est demandé pour accéder à la base de données pour établir la correspondance entre les *FQDN* et les *UUID*.

Renommer les paquets machine au format de nommage UUID :

```
PYTHONPATH=/opt/wapt PYTHONHOME=/opt/wapt python /opt/wapt/waptserver/scripts/migrate-hosts.py  -
↪C /root/wapt-private-20180312-1522.crt  -K /root/wapt-private.pem
```

Figure137: Certificat Code Signing

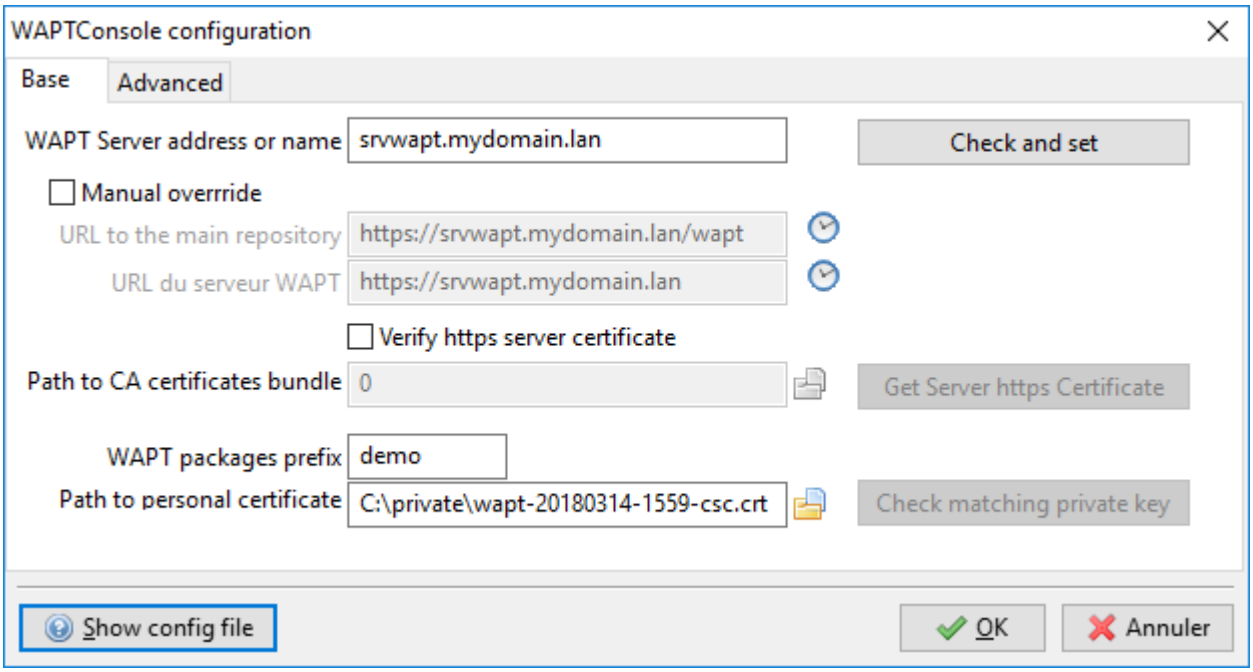


Figure138: Nouveau certificat Code Signing

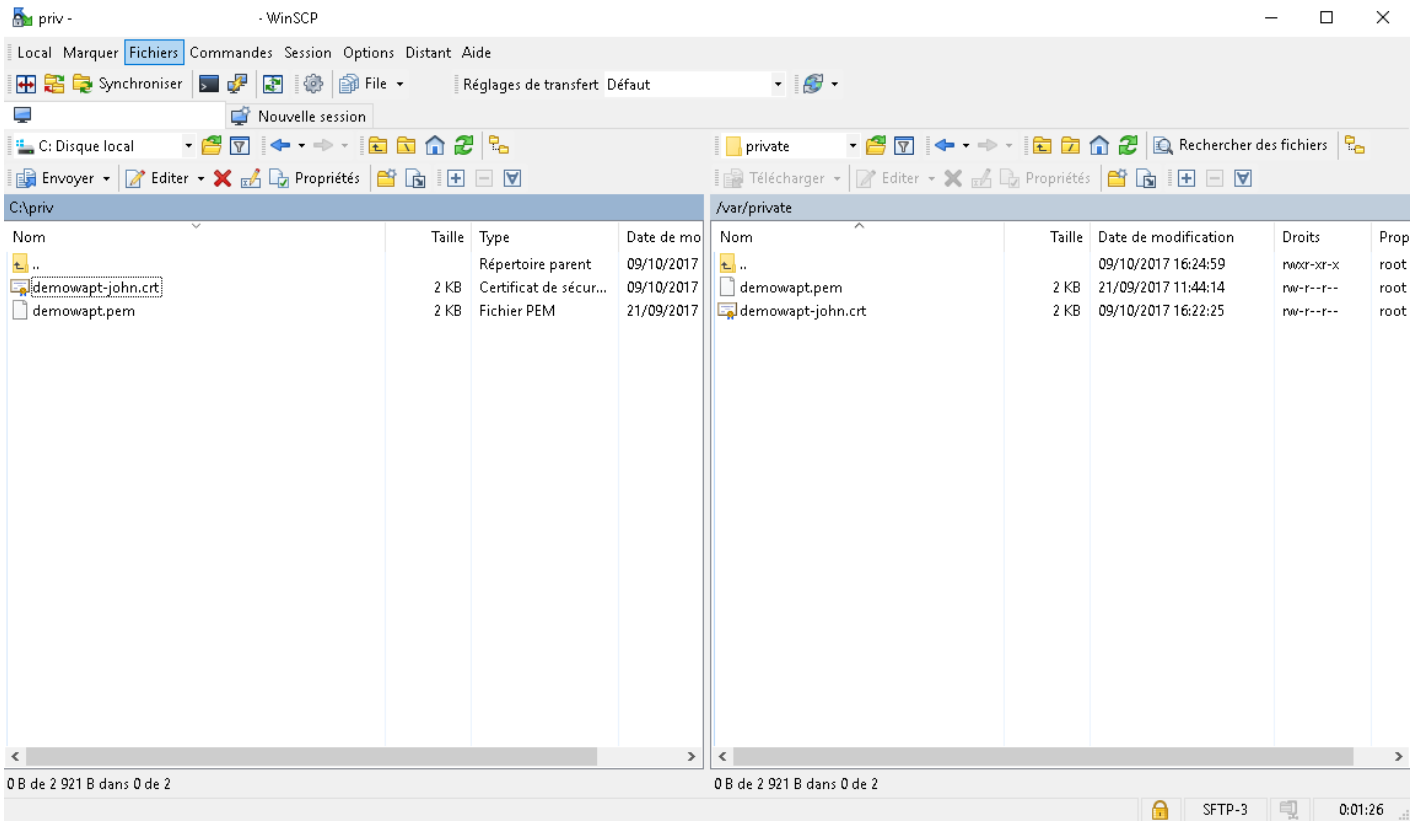


Figure139: Téléchargez temporairement votre certificat Code Signing sur le serveur WAPT

Note: La clé privée étant désormais protégée par un mot de passe, le mot de passe est demandé pour re-signer les paquets.

Attention: N'oubliez pas de **SUPPRIMER la clé** (`.pem`) du serveur après avoir re-signé les paquets !

Windows

Se connecter en SSH sur le serveur, re-signer les paquets *base* en précisant le chemin du certificat :

```
wapt-signpackages -i -s --message-digest=sha256,sha1 -c C:\private\wapt-private-20180312-1522.
↪crt C:\wapt\waptserver\repository\wapt\*.wapt
```

Indication: Le mot de passe *SuperAdmin* du serveur est demandé pour accéder à la base de données.

Renommer les paquets *host* au format de nommage UUID :

```
"C:\wapt\waptserver\scripts\migrate-hosts.bat" -C C:\private\wapt-private-20180312-1522.crt -K␣
↪C:\private\wapt-private.pem
```

Note: La clé privée étant désormais protégée par un mot de passe, le mot de passe est demandé pour re-signer les paquets.

Attention: N'oubliez pas de **SUPPRIMER la clé** (`.pem`) du serveur après avoir re-signé les paquets !

Générer la table des dépendances groupe

Dans la version 1.3 de WAPT, le filtrage par groupe était effectué en parcourant le fichier `Packages` du dossier `wapt-host` du serveur.

Pour des raisons de performance, ce filtrage est maintenant géré en parcourant une table PostgreSQL. Cette table sera alimentée au fur et à mesure en version WAPT 1.6.

Pour regénérer rapidement cette table, l'astuce consiste a créer un groupe *mig-temp* dans WAPT.

Vous pouvez ensuite appliquer le paquet *mig-temp* à tous les clients WAPT depuis la console WAPT (`CTRL+A` dans l'inventaire, puis *Clic droit* → *Ajouter en dépendance* et sélectionnez le paquet *mig-temp*).

Faire l'opération inverse. (`CTRL+A` dans l'inventaire, puis *Clic-droit* → *Supprimer des dépendances* et sélectionner *mig-tempo*).

Le filtrage groupe devrait à nouveau fonctionner.

Mettre à jour les clients WAPT

Vous pouvez maintenant suivre la *procédure pour créer l'agent WAPT* !!

> **Attention:** Le changement de branche modifie le mode de signature.
>
> Si vous mettez à jour depuis la version 1.3.13, le paquet **`waptupgrade`** devrait s'installer correctement si vous cochez la case *signer waptupgrade en sha256 ET sha1* lors de la génération de l'agent wapt.
>
> Si vos agents sont dans une version inférieure à 1.3.13, le paquet **`waptupgrade` ne fonctionnera pas**.

Vous pouvez mettre en place une GPO **`waptdeploy`** pour *déployer le nouvel agent WAPT sur votre parc*.

Lancer l'installation de waptupgrade sur les clients

Dans la version 1.6 il n'est plus possible de contacter les clients < 1.6 depuis la console WAPT.

Pour vous permettre de lancer l'installation du paquet *waptupgrade* sur les 1.3 nous avons prévu un petit script.

Linux

Vous pouvez donc lancer ce script :

```
/opt/wapt/waptserver/trigger_action.sh prefix-waptupgrade
```

Windows

Vous pouvez donc lancer ce script :

```
"C:\wapt\waptserver\trigger_action.bat" prefix-waptupgrade
```

6.9.4 Mise à jour mineure

Cette procédure concerne la mise à jour classique du serveur WAPT pour les versions mineures, par exemple de 1.7.4 à 1.8.

Le procédé de migration comprend :

- la mise à jour du serveur WAPT ;
- la mise à jour de la console WAPT ;
- la mise à jour de l'agent WAPT ;
- la mise à jour de la GPO de déploiement ;

Note: Pour la migration majeure WAPT 1.3 vers WAPT 1.6, suivez la documentation de *mise à jour majeure 1.3 vers 1.6*.

Effectuer des mises à jour mineures sur un serveur WAPT basé sur Debian

Attention: Les ports 80, 443 sont utilisés par le serveur WAPT et doivent être disponibles.

- d'abord mettre à jour le serveur Debian subjacent :

```
apt update && apt upgrade -y && apt dist-upgrade
```

- ajouter le dépôt de paquets Debian, importer la clé GPG de ce dépôt et installer les paquets pour le serveur WAPT :

WAPT Enterprise

Indication: Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **deb** pour accéder au dépôt WAPT Enterprise.

```
apt install apt-transport-https lsb-release
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo  "deb  https://user:password@srvwapt-pro.tranquil.it/entreprise/debian/wapt-1.8/ $(lsb_
↪release -c -s) main"  > /etc/apt/sources.list.d/wapt.list
apt update
apt install tis-waptserver tis-waptrepo tis-waptsetup
```

WAPT Community

```
apt install apt-transport-https lsb-release
wget -O - https://wapt.tranquil.it/debian/tiswapt-pub.gpg  | apt-key add -
echo  "deb  https://wapt.tranquil.it/debian/wapt-1.8/ $(lsb_release -c -s) main"  > /etc/apt/
↪sources.list.d/wapt.list
apt update
apt install tis-waptserver tis-waptsetup
```

Post-configuration

- lancer la post-configuration

Note:

- il est recommandé de lancer la post-configuration après chaque mise à jour pour que le serveur utilise le format de configuration le plus sûr ;
- il n'est pas obligatoire de redéfinir un mot de passe pour la console WAPT lors de la post-configuration ;
- si vous avez personnalisé la configuration de **`Nginx`**, ne pas répondre *Oui* lorsque postconf vous demande de configurer **`Nginx`** ;

Attention:

- avec WAPT 1.8, la post-configuration va déplacer tous les fichiers à la racine du dossier waptwua (`/var/www/waptwua`).
- si la réplication de dépôts à été mise en place, cela aura pour conséquencede resynchroniser toutes les KB/CAB sur vos dépôts distants.

```
/opt/wapt/waptserver/scripts/postconf.sh
```

Le mot de passe demandé en étape #4 est utilisé pour accéder à la console WAPT.

- démarrer le serveur WAPT :

```
systemctl restart waptserver
```

- mettre à jour la console WAPT en suivant la même procédure que pour *installer la console WAPT* ;
- puis *créer l'agent WAPT* :

  Cependant, il faudra conserver le même préfix et surtout ne rien changer concernant le couple clé privée / clé publique !

  Cela va générer un paquet **waptupgrade** dans le dépôt privé.

  Note: Il y a deux manières de déployer la mise à jour :

  - soit utiliser une GPO et **`waptdeploy`** ;
  - soit créer un paquet **waptupgrade** et l'affecter à l'ensemble du parc ;

- mettre à jour les clients WAPT

  La procédure à suivre pour la mise à jour des clients est la même que pour la première installation.

  Télécharger et installer la dernière version de l'agent WAPT qui a été mis à jour : https://wapt.mydomain.lan/wapt/waptagent.exe.

  Comme mentionné ci-dessus, cette procédure peut s'automatiser avec une GPO ou un paquet de mise à jour **waptupgrade**.

Effectuer des mises à jour mineures sur un serveur WAPT basé sur CentOS / RedHat

Attention: Les ports 80, 443 sont utilisés par le serveur WAPT et doivent être disponibles.

- d'abord mettre à jour le serveur CentOS / RedHat subjacent ;

```
yum update
```

WAPT Enterprise

Modifier l'adresse du dépôt puis lancer la mise à niveau.

Indication: Pour accéder au site de téléchargement WAPT Enterprise, vous devez utiliser le nom d'utilisateur et le mot de passe fournis par notre service commercial.

Remplacez **user** et **password** dans le paramètre **baseurl** pour accéder au dépôt WAPT Enterprise.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Enterprise Server Repo
baseurl=https://user:password@srvwapt-pro.tranquil.it/entreprise/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF

wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/centos7/RPM-GPG-KEY-TISWAPT-7"; rpm --
↪import /tmp/tranquil_it.gpg
yum install epel-release
yum install cabextract
yum install postgresql96-server postgresql96-contrib tis-waptserver tis-waptsetup
```

WAPT Community

- modifier l'adresse du dépôt puis lancer la mise à niveau :

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/centos7/wapt-1.8/
enabled=1
gpgcheck=1
EOF

wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/centos7/RPM-GPG-KEY-TISWAPT-7"; rpm --
↪import /tmp/tranquil_it.gpg
yum install postgresql96-server postgresql96-contrib tis-waptserver tis-waptsetup
```

Post-configuration

- lancer la post-configuration :

  Note:

  - il est recommandé de lancer la post-configuration après chaque mise à jour pour que le serveur utilise le format de configuration le plus sûr ;
  - il n'est pas obligatoire de redéfinir un mot de passe pour la console WAPT lors de la post-configuration ;

– si vous avez personnalisé la configuration de **Nginx**, ne pas répondre *Oui* lorsque postconf vous demande de configurer **Nginx** ;

Attention:

– avec la post-configuration WAPT 1.8, les paquets WAPT WUA seront déplacés de leur emplacement de stockage actuel vers le dossier racine de waptwua (`/var/www/waptwua`).

– si la réplication de dépôts à été mise en place, cela aura pour conséquence de resynchroniser toutes les KB/CAB sur vos dépôts distants.

```
/opt/wapt/waptserver/scripts/postconf.sh
```

- démarrer le serveur WAPT :

```
systemctl start waptserver
```

- mettre à jour la console WAPT en suivant la même procédure que pour *installer la console WAPT* ;
- puis *créer l'agent WAPT* :

  Cependant, il faudra conserver le même préfix et surtout ne rien changer concernant le couple clé privée / clé publique !

  Cela va générer un paquet **waptupgrade** dans le dépôt privé.

  Note: Il y a deux manières de déployer la mise à jour :

  – soit utiliser une GPO et **waptdeploy** ;

  – soit créer un paquet **waptupgrade** et l'affecter à l'ensemble du parc ;

- mettre à jour les clients WAPT :

  La procédure à suivre pour la mise à jour des clients est la même que pour la première installation.

  Télécharger et installer la dernière version de l'agent WAPT qui a été mis à jour : https://wapt.mydomain.lan/wapt/waptagent.exe.

  Comme mentionné ci-dessus, cette procédure peut s'automatiser avec une GPO ou un paquet de mise à jour **waptupgrade**.

Effectuer des mises à jour mineures sur un serveur WAPT basé sur Windows

Attention: En cas d'utilisation d'une GPO pour la mise à jour de l'agent WAPT, il faut exclure le serveur WAPT de l'OU où est appliqué la GPO.

Note: Le serveur WAPT peut s'installer sur Windows 7 64bit et Windows 10 64bit et Windows Server 2008 / 2008R2, 2012 / 2012R2, 2016 et 2019 64bit.

- sur la machine Windows hébergeant le serveur WAPT, téléchargez la dernière version de l'installateur depuis le site web de Tranquil IT WAPTServerSetup.exe et lancez-la en tant qu"*Administrateur Local* ;
- installer la mise à jour ;

Note: La procédure à suivre pour la mise à jour est la même que pour *installer un serveur WAPT sur Windows*.

Attention: Le préfix ne doit pas changer et vous NE DEVEZ PAS regénérer de clé !

- on the workstation that you use to build your packages, manually download WAPTSetup from
- Community: waptserversetup.exe;
- Enterprise: waptserversetup.exe;
- puis *créer l'agent WAPT* :

  Cependant, il faudra conserver le même préfix et surtout ne rien changer concernant le couple clé privée / clé publique !

  Cela va générer un paquet **waptupgrade** dans le dépôt privé.

  Note: Il y a deux manières de déployer la mise à jour :

  - soit utiliser une GPO et **`waptdeploy`** ;
  - soit créer un paquet **waptupgrade** et l'affecter à l'ensemble du parc ;

- mettre à jour les clients WAPT

  La procédure à suivre pour la mise à jour des clients est la même que pour la première installation.

  Télécharger et installer la dernière version de l'agent WAPT qui a été mis à jour : https://wapt.mydomain.lan/wapt/waptagent.exe.

  Comme mentionné ci-dessus, cette procédure peut s'automatiser avec une GPO ou un paquet de mise à jour **waptupgrade**.

Mettre à jour le système d'exploitation

Mettre à jour de Debian 9 Stretch vers Debian 10 Buster

Afin de mettre à niveau votre serveur WAPT de Stretch à Buster, vous devez suivre la procédure standard pour Debian. Vous devez d'abord modifier les fichiers sources apt `/etc/apt/sources.list` et `/etc/apt/sources.list.d/wapt.list`, puis lancer la mise à niveau.

Par défaut, PostgreSQL n'est pas mis à niveau vers PostgreSQL 11. Il faut demander manuellement la mise à niveau. Après la mise à niveau, il est possible de supprimer l'ancienne base de données PostgreSQL 9.6.

```
sed -i 's/stretch/buster/g'  /etc/apt/sources.list
sed -i 's/stretch/buster/g'  /etc/apt/sources.list.d/wapt.list
apt update
apt update && apt dist-upgrade
pg_dropcluster --stop 11 main
```

(suite sur la page suivante)

(suite de la page précédente)

```
pg_upgradecluster -v 11 9.6 main
apt remove postgresql*-9.6
apt autoremove
/opt/wapt/waptserver/scripts/postconf.sh
```

Mettre à niveau de CentOS 7 à CentOS 8

WAPT ne soutient pas encore CentOS 8. Cette section sera mise à jour en conséquence lorsque le support sera ajouté.

6.10 Sauvegarder et restaurer le serveur WAPT

6.10.1 Sauvegarder le serveur WAPT sur Linux

Attention: Cette procédure n'est valable que pour WAPT 1.5 et supérieure.

- stopper les services liés à WAPT sur la machine ;

```
systemctl stop nginx
systemctl stop waptserver
systemctl stop wapttasks
```

- sauvegarder les répertoires suivants à l'aide d'un outil de sauvegarde (ex : **rsync** ou **WinSCP**) ;

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

- sauvegarder la base PostgreSQL à l'aide de l'outil **pg_dumpall** (adapter le nom du fichier et la date) ;

```
sudo -u postgres pg_dumpall > /tmp/backup_wapt_$(date +%Y%m%d).sql
```

- relancer les services liés à WAPT sur la machine ;

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

6.10.2 Restaurer le serveur WAPT sur Linux

En cas de crash, relancer une installation standard du serveur WAPT pour Linux.

- stopper les services liés à WAPT sur la machine ;

```
systemctl stop nginx
systemctl stop waptserver
systemctl stop wapttasks
```

- restaurer les répertoires :

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

- restaurer la base de données (adapter votre nom de fichier). La première commande **supprime** ci-dessous la base WAPT (si elle existe), vérifier que votre dump est correct avant de la supprimer :

```
sudo -u postgres psql -c "drop database wapt"
sudo -u postgres psql < /tmp/backup_wapt_20180301.sql
```

- appliquer les droits sur les répertoires restaurés :

```
chown -R wapt:www-data /var/www/wapt/
chown -R wapt:www-data /var/www/wapt-host/
chown -R wapt:www-data /var/www/waptwua/
chown -R wapt /opt/wapt/conf/
chown -R wapt /opt/wapt/waptserver/ssl/
```

- scanner les répertoires de paquets :

```
wapt-scanpackages /var/www/wapt/
```

- relancer les services liés à WAPT sur la machine ;

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

6.10.3 Sauvegarder le serveur WAPT sur Windows

- backup the WAPT repository folder `C:\wapt\waptserver\repository` and `C:\wapt\waptserver\conf` and `C:\wapt\waptserver\nginx\ssl` on a remote backup destination;

Sauvegarder la base de données PostgreSQL avec pg_dump.exe :

```
"C:\wapt\waptserver\pgsql-9.6\bin\pg_dumpall.exe" -U postgres -f C:\backup_wapt.sql
```

- relancer les services liés à WAPT sur la machine ;

6.10.4 Restaurer le serveur WAPT sur Windows

- restore the following directories `C:\wapt\waptserver\repository` and `C:\wapt\waptserver\conf` and `C:\wapt\waptserver\nginx\ssl`
- Autoriser tous les droits sur le répertoire `C:\wapt\waptserverepository` pour le groupe « Network Service »

Restaurer la base de données PostgreSQL avec pg_restore.exe :

```
"C:\wapt\waptserver\pgsql-9.6\bin\psql.exe" -f c:\backup_wapt.sql -U postgres
```

6.11 Répliquer des dépôts et fonctionner avec plusieurs dépôts

Les grandes organisations ayant des sites et des filiales à distance exigent parfois que certains services soient reproduits localement pour éviter l'encombrement de la bande passante (*Edge Computing*).

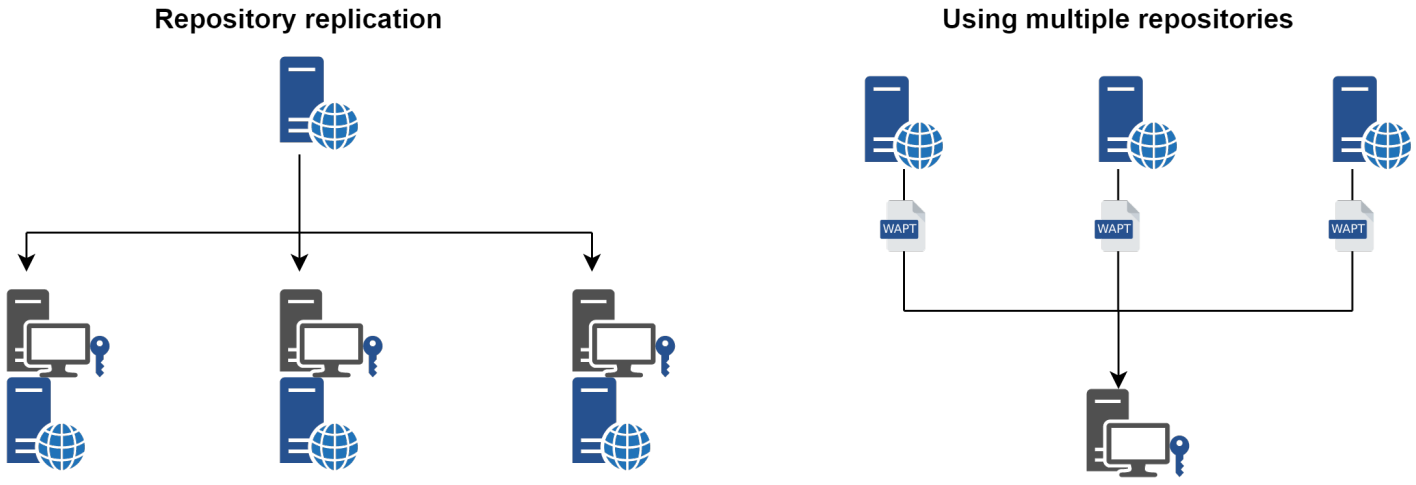


Figure140: Répliquer des dépôt et fonctionner avec plusieurs dépôts

Nouveau dans la version WAPT: Enterprise 1.8

WAPT Enterprise offre la possibilité de transformer des agents distants pour qu'ils servent de dépôts distants pouvant être gérés directement à partir de la console WAPT. Tous les agents WAPT peuvent alors être configurés de manière centralisée pour sélectionner automatiquement leur meilleur dépôt en fonction d'un ensemble de règles.

Vous trouverez ci-dessous les documentations pour mettre en place ces architectures.

6.11.1 Répliquer un dépôt pour préserver la bande passante des sites secondaires

Lorsque WAPT est utilisé sur plusieurs sites distants avec des liens Internet à faible débit, il devient pertinent de mettre en place un système de réplication pour alléger la charge réseau liée aux déploiement des mises à jour WAPT sur les machines de votre parc.

Ainsi, WAPT reste une solution à faible coût d'exploitation car vous n'aurez pas à passer à la fibre pour bénéficier des avantages de WAPT.

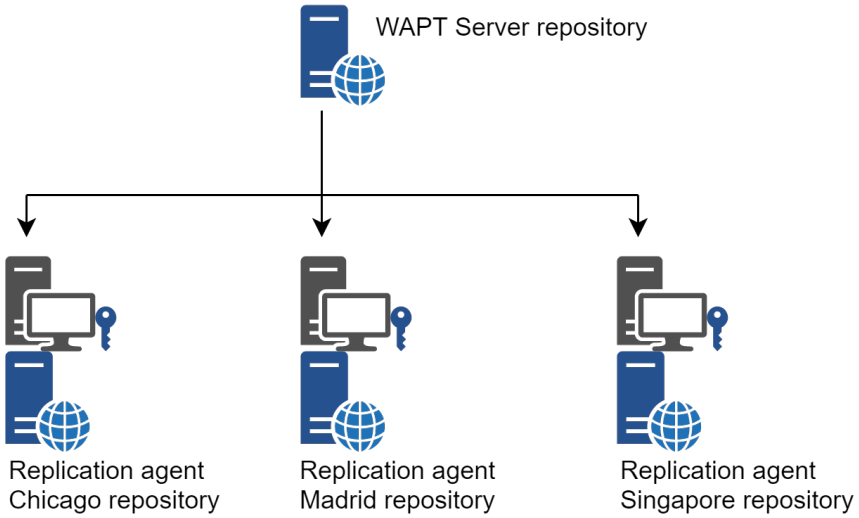


Figure141: Répliquer un dépôt

Le principe est le suivant :

- un appareil dédié de petite taille sans maintenance ayant le rôle de dépôt secondaire est déployé sur le réseau local de chaque site distant ; un poste de travail peut également être utilisé, bien qu'il puisse ne pas être allumé quand vous souhaiteriez vous y connecter ;
- le dépôt distant réplique les paquets depuis le dépôt principal WAPT et les éventuels dépôts secondaires ;
- les clients WAPT consultent en priorité le dépôt le plus proche ;

Indication: L'ancienne méthode utilisée pour synchroniser les dépôts était Syncthing et cette méthode est disponible à la fois pour la version communautaire et la version Entreprise du WAPT.

Vous pouvez trouver l'ancienne documentation ici : *(Déclassé) Réplication de dépôts avec Syncthing*.

La méthode expliquée ci-dessous s'applique uniquement à la version Enterprise.

Rôle de réplication de l'agent WAPT

Nouveau dans la version WAPT: 1.8 Enterprise

À partir de WAPT 1.8, la réplication du dépôt peut être activée en utilisant un agent WAPT installé sur une machine existante, une *appliance* dédiée ou une machine virtuelle.

Le rôle de réplication est déployé par le biais d'un paquet WAPT qui active le serveur web **`Nginx`** et configure le déclenchement des mises à jour, les types de paquets, la synchronisation des paquets, et bien plus encore.

Cette fonctionnalité permet aux agents WAPT de trouver dynamiquement leur dépôt WAPT disponible le plus proche à partir d'une liste de règles stockées sur le serveur WAPT.

Comportement de réplication

La réplication du dépôt dans WAPT est maintenant nativement gérée par des agents WAPT spéciaux (**Versions entreprise uniquement**).

Il est basé sur un fichier `sync.json` qui indexe tous les fichiers présents dans ces dossiers :

- wapt ;
- waptwua ;
- wapt-host ;

Permettre la réplication a les effets suivants :

- une fois `enable_remote_repo` activé sur un agent WAPT, il synchronisera les paquets localement dans le dossier `local_repo_path` ;
- il ajoute l'agent WAPT dans l'onglet *Dépôts* comme un dépôt distant, permettant de nouvelles actions telles que *Forcer la Syncro* ou *Vérifier les fichiers* ;
- par défaut, seul le dossier `wapt` est synchronisé, vous pouvez sélectionner le dossier à synchroniser en ajoutant des éléments dans les paramètres `remote_repo_dirs` ;
- La tranche horaire de synchronisation peut être configurée avec les paramètres `local_repo_time_for_sync_start` et `local_repo_time_for_sync_stop` ;
- La bande passante allouée à la synchronisation peut être configurée avec `local_repo_limit_bandwidth` ;

Tous les paramètres de la synchronisation du dépôt WAPT doivent être définis dans la section `[repo-sync]` du fichier de configuration `wapt-get.ini` de l'agent WAPT.

Permettre la réplication sur l'agent WAPT

Pour permettre la réplication sur un agent existant (Linux/Windows), vous devez déployer un paquet WAPT. Son rôle est de :

- installer et activer le programme **`Nginx web server`** ;
- configurer l'hôte virtuel nginx ;
- activer la configuration du dépôt distant dans `wapt-get.ini` ;

Un paquet WAPT prêt à l'emploi est disponible dans notre dépôt public pour permettre la réplication du dépôt sur des agents WAPT Windows ou Linux : https://store.wapt.fr/store/tis-remote-repo-conf.

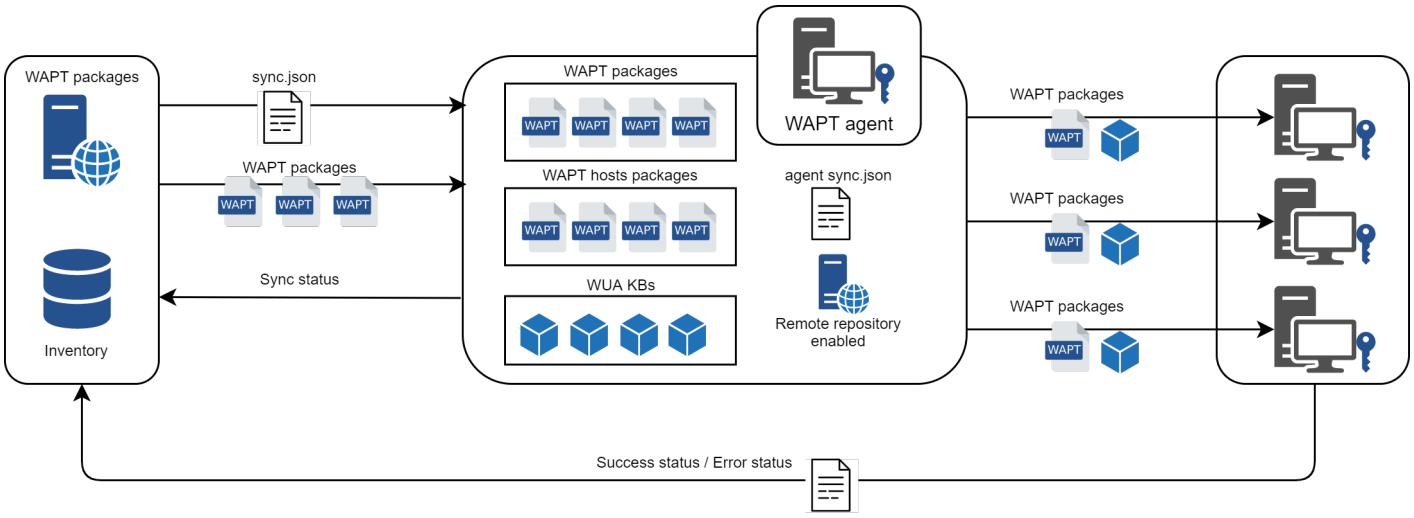


Figure142: Comportement de réplication

Configurer la réplication de l'agent WAPT

La configuration de la réplication de l'agent WAPT est définie dans la section `[repo-sync]` dans le fichier de configuration `wapt-get.ini` de l'agent WAPT :

Options	Obligatoire	Valeur Exemple	Définition
`enable_remote_repo`	Yes	`True`	Déclenche les connexions de synchronisation des dépôts à distance.
`local_repo_path`	Yes	`/var/www/`	Définit le chemin racine des paquets locaux
`local_repo_time_for_sync_start`	No	`22:30`	Set sync start time (HH:MM / 24h format)
`local_repo_time_for_sync_end`	No	`05:30`	Définit l'heure d'arrêt de la synchronisation (HH:MM / format 24h)
`local_repo_sync_task_period`	No	`25`	Définit la fréquence de synchronisation (minutes)
`local_repo_limit_bandwidth`	No	`2`	Définit la bande passante maximale allouée aux synchronisations (Mbits/s)
`remote_repo_dirs`	No	`wapt,waptwua, wapt-host`	Définit les dossiers synchronisés (par défaut : wapt,waptwua)

Vous trouverez ci-dessous un exemple d'un `wapt-get.ini` :

```
[global]
...
use_repo_rules = True

[repo-sync]
enable_remote_repo = True
local_repo_path = D:\WAPT\
local_repo_time_for_sync_start = 20:30
local_repo_time_for_sync_end = 05:30
local_repo_sync_task_period = 25
local_repo_limit_bandwidth = 4
remote_repo_dirs = wapt,waptwua,wapt-host
```

Configuration de la réplication du serveur WAPT

Le serveur WAPT doit connaître les dépôts à synchroniser dans la section `[options]` de son fichier `waptserver.ini` présent dans `/opt/wapt/conf/` :

Options	Valeur Exemple	Définition
`remote_repo_support`	True	ajouter la machine distante :

Then we must restart both *waptserver* and *wapttask*:

```
systemctl restart waptserver wapttask
```

Règles relatives aux dépôts

Comportement des règles de dépôt

En activant la prise en charge des règles du dépôt, les agents WAPT récupéreront automatiquement leur fichier `rules.json` sur le serveur WAPT.

Le fichier `rules.json` est un fichier *.JSON* signé qui contient une liste de règles triées à appliquer à l'agent WAPT, afin de rediriger ses demandes de téléchargements vers le dépôt le plus approprié.

Si aucune règle ne s'applique, alors l'agent WAPT se rabat sur le paramètre `repo_url` du fichier de configuration `wapt-get.ini` du serveur WAPT.

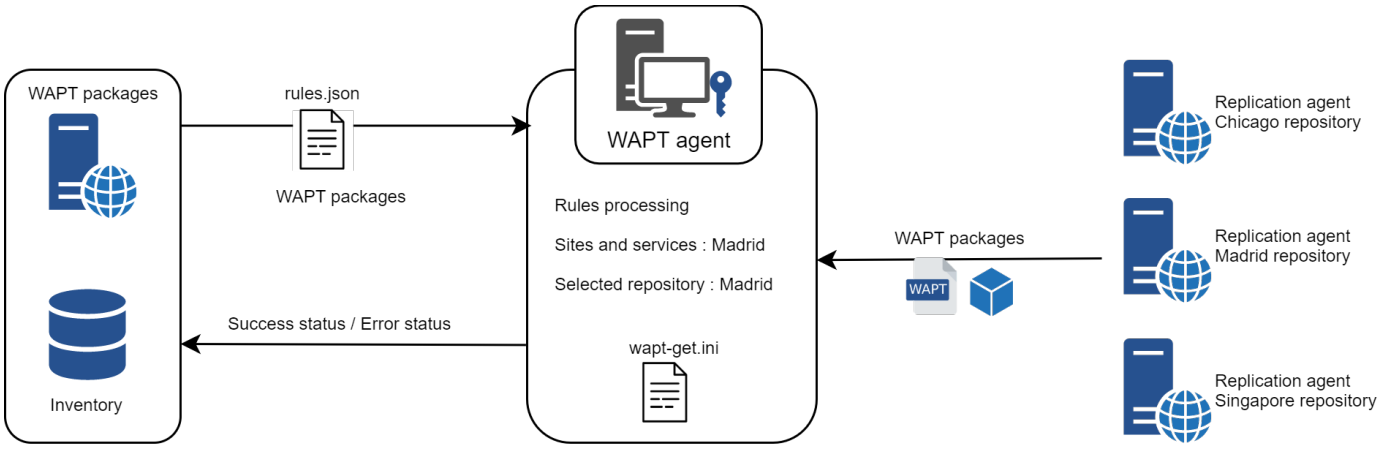


Figure143: Comportement des règles de dépôt

Répliquer un dépôtActiver les règles des dépôts

Les règles de dépôt sont configurées dans la console WAPT.

Les règles peuvent être basées sur plusieurs paramètres :

Options	Valeur Exemple	Définition
Le nom du domaine	`ad.domain.lan`	Règle basée sur le nom de domaine Active Directory
Sites et services du domaine	Sites et services du domaine	Règle basée sur les sites et services Active Directory
IP de l'agent	`192.168.85.0/24`	Règle basée sur le sous-réseau IP de l'agent WAPT
IP publique	`256.89.299.22/32`	Règle basée sur l'IP publique de l'agent WAPT (hôtes NATés)
Nom d'hôte	`desktop-04feb1`	Règle basée sur le nom d'hôte

Ajouter une règle dans la console WAPT

In *Repositories*, click on the *Add rule* button. The following window appears:

..figure:: create_new_rule.png :align:center

You can then choose from the different above parameters and affect values to a specific secondary WAPT repository. The option *No Fallback* will prevent from falling back to the main WAPT server and will avoid potential network congestion.

The rules are applied from top to bottom, and the first rule that matches the conditions overrides all the other rules below.

Utiliser des règles de dépôt sur les agents WAPT

Avertissement: Si vous avez configuré des redirections GeoIP sur Nginx, vous devriez les désactiver car elles pourraient entrer en conflit avec les règles du dépôt.

Pour activer les règles du dépôt de l'agent WAPT, vous devez activer ce paramètre dans la section `[global]` du fichier de configuration `wapt-get.ini` de l'agent WAPT :

Options	Obligatoire	Valeur Exemple	Définition
`use_repo_rules`	No	`True`	Permettre l'utilisation des règles du dépôt

Vous trouverez ci-dessous un exemple d'un `wapt-get.ini` :

```
[global]
...
use_repo_rules = True
```

En parlant de dépôts, il est parfois utile de configurer deux ou plusieurs dépôts pour différentes utilisations (prod, dev, licensed, selfservice).

6.11.2 Fonctionner avec plusieurs dépôts

Le multi-dépôt est maintenant géré par WAPT ; cette fonctionnalité est utile dans différents cas :

- pour utiliser un dépôt privé servant à héberger les applications métier propres à la structure en complément d'un autre dépôt ;
- pour avoir des dépôts privés à proximité des utilisateurs dans un scénario multi-établissements ;
- pour permettre l'usage de dépôts hébergeant des applications en libre service et des dépôts avec restrictions d'accès (logiciels soumis à licences . . .) ;

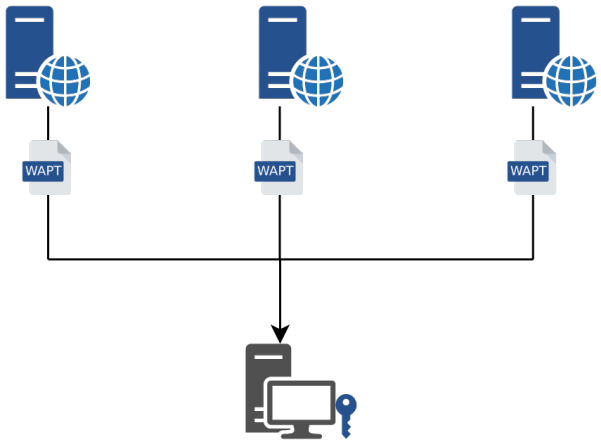


Figure144: Le multi-dépôts avec WAPT

Attention: C:\Program Files (x86)\wapt\

Voir la documentation *pour créer l'agent WAPT*.

Paramétrer les agents WAPT

- paramètre *repositories* :

  Le paramètre *repositories*, dans la section `[global]` permet de consulter plusieurs dépôts, par exemple `[private]` et `[tranquilit]` ici, dont les paramètres sont à indiquer dans des sections supplémentaires en fin du fichier `wapt-get.ini`.

```
repositories=private,tranquilit
```

- paramètres des dépôts secondaires :

```
[private]
repo_url=https://srvwapt.mydomain.lan/wapt

[tranquilit]
repo_url=https://wapt.tranquil.it/wapt
```

  Avec cette configuration, le client verra maintenant les paquets disponibles sur le dépôt secondaire en plus du dépôt principal.

  L'agent WAPT ira rechercher des mises à jours dans les deux dépôts.

```
wapt-get search
```

Les paquets du dépôt secondaire sont également visibles sur l'interface web http://127.0.0.1:8088 du poste équipé de l'agent WAPT.

Paramétrer la console WAPT avec plusieurs dépôts privés

Après avoir paramétré les agents WAPT pour utiliser deux dépôts, on peut faire apparaître ces deux dépôts simultanément dans la console.

Pour cela, modifier le fichier `%appdata%..localwaptconsole\waptconsole.ini` :

```
[private]
repo_url=https://srvwapt.mydomain.lan/wapt

[tranquilit]
repo_url=https://wapt.tranquil.it/wapt
```

6.11.3 Configurations et caractéristiques dépréciées

(Déclassé) Réplication de dépôts avec Syncthing

Indication: La synchronisation des dépôts WAPT est désormais native dans WAPT Enterprise.

Vous pouvez trouver la nouvelle documentation ici : *Répliquer un dépôt pour préserver la bande passante des sites secondaires*.

Avertissement: Cette partie de la documentation n'est plus maintenue. Vous pouvez l'utiliser comme source d'idées pour faire votre réplication tout en utilisant la version Community.

Présentation de Syncthing

Syncthing est un logiciel open-source multiplateforme de synchronisation de fichiers pair à pair.

Il permet de synchroniser des répertoires sur plusieurs machines en garantissant la sécurité et l'intégrité des fichiers.

La documentation officielle de Syncthing est *disponible en ligne <https://docs.syncthing.net/>_*.

Mettre en place de la réplication

Indication: La documentation suivante concerne des dépôts et serveurs WAPT basés sur Linux Debian ou CentOS / RedHat.

Installer le dépôt WAPT distant

Linux Debian

```
echo "deb https://wapt.tranquil.it/debian/ ./ " > /etc/apt/sources.list.d/wapt.list
apt update
apt upgrade -y
apt install tis-waptrepo
```

Linux CentOS / RedHat

Le dépôt WAPT distant est installé, on doit mettre en place la réplication Syncthing.

Configurer le service web Apache

```
a2enmod ssl
a2ensite default-ssl.conf
```

- modifier ensuite les fichiers de configuration Apache afin d'indiquer les racines correcte aux *VirtualHosts* :

```
/etc/apache2/sites-available/default-ssl.conf
/etc/apache2/sites-available/000-default.conf
```

- modifier la valeur de *DocumentRoot* dans chaque fichier de configuration :

```
- DocumentRoot /var/www/html
+ DocumentRoot /var/www
```

- redémarrer **Apache** afin de prendre en compte la configuration :

```
/etc/init.d/apache2 restart
```

Note: Il est préférable d'indiquer dans la configuration Apache des certificats SSL valides pour les dépôts distants.

- vider le contenu des dossiers `/var/www/wapt` et `/var/www/wapt-host`, Syncthing ira peupler ces dossiers avec les données du dépôt principal :

```
rm -rf /var/www/wapt/*
rm -rf /var/www/wapt-host/*
```

Installer Syncthing sur les dépôts

Note: Cette procédure est à appliquer sur le dépôt principal et sur les dépôts secondaires.

```
## Debian
apt update
apt install sudo curl apt-transport-https
curl -s https://syncthing.net/release-key.txt | apt-key add -
echo "deb https://apt.syncthing.net/ syncthing stable" | tee /etc/apt/sources.list.d/syncthing.
↪list
apt update
apt install syncthing

## CentOS 7
wget https://github.com/mlazarov/syncthing-centos/releases/download/v0.14.7/syncthing-0.14.7-0.
↪el7.centos.x86_64.rpm --no-check-certificate
yum install syncthing-0.14.7-0.el7.centos.x86_64.rpm
```

Configurer Syncthing

Opérations à effectuer :

- ajout du service dans **`systemd`** ;
- modifier le port d'écoute du service sur **0.0.0.0** ;
- création d'un compte utilisateur administrateur et définition d'un mot de passe fort ;
- activation du protocole https pour l'accès Web ;
- créer le fichier de définition du service **`waptsync`** en éditant `/etc/systemd/system/waptsync.service` :

```
[Unit]
Description=WAPT respository sync with syncthing
Documentation=http://docs.syncthing.net/
After=network.target
;Wants=syncthing-inotify@.service

[Service]
User=wapt
ExecStart=/usr/bin/syncthing -logflags=0 -home=/opt/wapt/.config/syncthing/ -no-restart
Restart=on-failure
SuccessExitStatus=3 4
RestartForceExitStatus=3 4

[Install]
WantedBy=multi-user.target
```

- créer l'arborescence nécessaire au lancement du service :

```
mkdir /opt/wapt/.config/
mkdir /opt/wapt/.config/syncthing/
```

- changer le propriétaire de cette arborescence de fichiers :

```
chown -R wapt:www-data /opt/wapt/.config/
```

- activer le service puis le lancer, les fichiers de configuration vont apparaître dans le dossier `/opt/wapt/.config/syncthing/` :

```
systemctl enable waptsync
systemctl start waptsync
systemctl stop waptsync
```

- modifier le port d'écoute dans le fichier `/opt/wapt/.config/syncthing/config.xml` :

```
<gui enabled="true" tls="true" debugging="false">
    <address>0.0.0.0:8384</address>
    <apikey>4jvEiL24UbFddsdsAQxqsfixNaLt</apikey>
    <theme>default</theme>
</gui>
```

- démarrer le service :

```
systemctl start waptsync
```

Configurer le service Web de Syncthing

L'interface Web Syncthing est maintenant accessible à l'adresse `http://srvwapt.mydomain.lan:8384`.

Opérations à effectuer :

- modifier le nom de l'appareil ;
- ajouter un *Utilisateur autorisé GUI* ;
- ajouter un mot de passe d'authentification pour l'*Utilisateur autorisé GUI* ;
- cocher « utiliser l'HTTPS pour le GUI » ;
- cliquer sur *Enregister* ;
- en ssh sur le serveur, redémarrer le service Syncthing :

```
systemctl restart waptsync
```

L'interface Web Syncthing est maintenant uniquement accessible en https à l'adresse : https://srvwapt.mydomain.lan:8384.

- dans la liste des partages, supprimer le répertoire par défaut : *Modifier* → *Enlever* ;
- configurer la réplication :

> **Note:** Ces actions sont à effectuer sur le serveur WAPT.

Dans la liste des dossiers partagés (*Partages*) :

- ajouter un répertoire partagé avec *Ajouter un appareil* ;
- renseigner le chemin du dossier à partager, ex : `/var/www/wapt/` ;
- dans le menu déroulant *Type de partage → Envoyer seulement* ;
- dans le menu déroulant *Ordre de récupération des fichiers → Les plus anciens en premier* ;
- recommencer l'opération pour wapt-host: `/var/www/wapt-host/` ;
- ajouter la machine distante :

  Une fois Syncthing installé sur les deux appareils, récupérer l'ID de l'appareil distant (*Actions → afficher mon ID*).

  Cet identifiant unique est de la forme

  ```
  DSINDDC-23ORDNM-PAK6FCL-ZJAKNCH-61GWXAT-77PC3JM-RZ4PPYP-K1QERAV
  ```

  Sur le serveur maître, dans la liste des appareils (*Autres appareils*) :

  * *Ajouter un appareil* ;
  * renseigner l'ID du dépôt distant ;
  * cocher les partages *wapt* et *wapt-host* ;

  Sur le dépôt distant effectuer les actions suivantes :

  * l'appareil distant reçoit une notification d'approbation d'ajout par l'appareil maître ;
  * le client reçoit ensuite une popup pour accepter les partage `wapt` et `wapt-host` ;

La réplication est maintenant mise en place.

- sécuriser la réplication :

Par défaut, les paramètres suivant sont activés pour Syncthing :

Table25: Paramètres Syncthing

Options	Description
Activer la translation d'adresses (*NAT*)	Utiliser un mappage de port UPnP pour les connexions de synchronisation entrantes.
Découverte locale	Syncthing fonctionnera alors avec un system de broadcast pour s'annoncer auprès des autres Syncthing.
Découverte globale	Syncthing s'inscrit sur un serveur **mondial**, et peut utiliser celui-ci pour rechercher d'autres périphériques.
Relais possible	L'utilisation de relais permet d'utiliser des serveurs externes pour relayer la communication. Le relais est activé par défaut mais ne sera utilisé que si deux périphériques ne peuvent pas communiquer directement entre eux.

Ce mode de fonctionnement permet une mise en place facilitée mais ce n'est pas le mode de fonctionnement le plus conseillé pour la sécurité.

- décocher toutes les cases dans la configuration réseau ;
- définir le port d'écoute du protocole (par défaut 22000) ;
- remplacer *default par *tcp://0.0.0.0:22000* ;

Se rendre ensuite sur l'interface de dépôts distants, cliquer sur *Modifier*, puis définir l'adresse de la machine distante :

- remplacer *dynamic* par *tcp://ip-du-syncthing-distant:22000*

Cette configuration permet de limiter les connexions externes du service Synthing.

Configurer les agents WAPT

Les clients des sites distants doivent maintenant être configurés pour pointer vers leurs dépôts les plus proches.

Deux solutions existent :

- utiliser la détection automatique via les champs *SRV DNS* ;
- modifier le champs *repo_url* dans le `wapt-get.ini` des agents WAPT manuellement ou à l'aide d'un paquet WAPT de configuration ;

Exemple de configuration de l'agent WAPT - dépôt local renseigné :

```
[global]
waptupdate_task_period=120
waptserver=https://srvwapt.mydomain.lan
repo_url=https://localrepo.mydomain.lan/wapt/
use_hostpackages=1
```

Exemple de paquet pour modifier le champ *repo_url* de `wapt-get.ini` :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
  print('Modifier la configuration agent pour le site de Colmar')
  inifile_writestring(WAPT.config_filename,'global','repo_url',
                      'https://wapt.city02.mydomain.lan/wapt/')
```

6.12 Améliorer la sécurité du serveur WAPT

Par défaut, tous les paquets WAPT sont signés avec votre clé privée, ce qui offre déjà un haut niveau de sécurité. Cependant, vous pouvez améliorer davantage la sécurité de WAPT.

Pour pleinement sécuriser la configuration de votre WAPT; vous aurez besoin de mettre en place les choses suivantes:

- activer les enregistrements authentifiés pour filtrer qui est authorisé à enregistrer l'appareil sur le serveur WAPT;
- activer la vérification du certificat https sur les agents et la consoleafin de s'assurer que les agents WAPT et la console WAPT soient connectés au bon serveur WAPT;
- configurer l'authentification par Active Directory pour permettre l'accès àla console WAPT uniquement pour les administrateurs WAPT;
- activer l'Authentification par Certificat Coté Client pour uniquement authoriser les appareils authentifiés à accéder au serveur WAPT (Note: c'est d'autant plus important si vous souhaitez exposer votre serveur WAPT vers l'extérieur dans une DMZ (De-Militarized Zone));

6.12.1 Configurer le pare-feu sur le serveur WAPT

La configuration du pare-feu du serveur WAPT est essentielle et devrait être la première étape vers une meilleure sécurité.

Comme WAPT vise à être sécurisé *by design*, seul un ensemble minimal de ports ouverts est nécessaire sur le serveur WAPT par rapport aux autres solutions.

Vous trouverez dans la documentation suivante des conseils autour des configurations de pare-feu pour renforcer la sécurité du serveur WAPT.

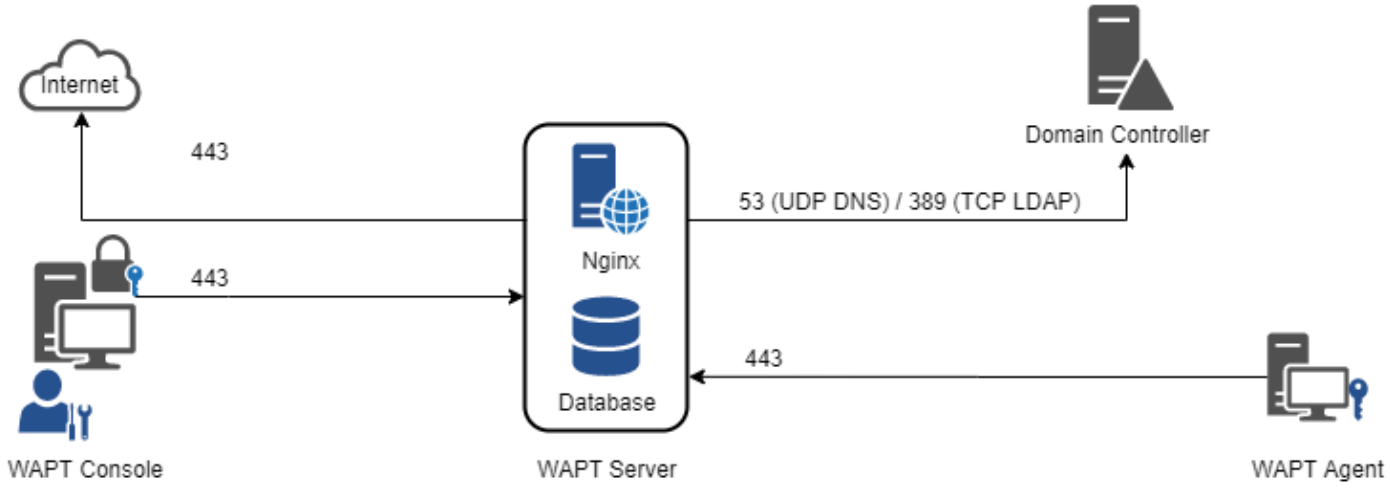


Figure145: Diagramme de flux de données de WAPT

Comme vous pouvez le voir, seuls les ports 80 et 443 doivent être ouverts pour les connexions entrantes car WAPT fonctionne avec des *webockets* initiées par les agents WAPT.

Configurer le pare-feu pour le serveur WAPT sous Debian Linux

Par défaut sur Debian Linux, aucune règle de pare-feu ne s'applique.

- désactiver **`ufw`** et installer **`firewalld`** à la place :

```
ufw disable
apt update
apt -y install firewalld
```

- appliquez simplement cette configuration **`firewalld`** :

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --zone=public --add-port=443/tcp --permanent
systemctl restart firewalld
```

Configurer le pare-feu pour le serveur WAPT sur CentOS

- appliquez simplement cette configuration **firewalld** :

```
systemctl start firewalld
systemctl enable firewalld
firewall-cmd --zone=public --add-port=80/tcp --permanent
firewall-cmd --zone=public --add-port=443/tcp --permanent
systemctl restart firewalld
```

6.12.2 Configurer l'authentification par Kerberos

Note:

- sans l'authentification par Kerberos, vous devez soit faire confiance lors de l'enregistrement initial ou bien entrer un mot de passe pour chaque poste lors de l'enregistrement;
- pour plus d'information, consultez la documentation *enregistrer une machine sur le serveur WAPT* et *signer les mises à jour d'inventaire*;
- l'authentification par Kerberos ne sera uniquement utilisé que lors de l'enregistrement de l'appareil;

Installer les composants pour Kerberos et configurer le fichier krb5.conf

```
#Debian
apt install krb5-user msktutil libnginx-mod-http-auth-spnego

#CentOS
yum install krb5-workstation msktutil nginx-mod-http-auth-spnego
```

Note: Cette fonctionnalité n'est pas disponible avec un serveur WAPT Windows

Modifiez le fichier `/etc/krb5.conf` et **remplacez tout le contenu par les 4 lignes suivantes** en remplaçant **MYDOMAIN.LAN** par votre nom de domaine Active Directory (i.e. *<MYDOMAIN.LAN>*).

Attention: Le `default_realm` doit absolument être écrit en **lettres MAJUSCULES** !!

```
[libdefaults]
  default_realm = MYDOMAIN.LAN
  dns_lookup_kdc = true
  dns_lookup_realm=false
```

Récupérez un ticket keytab. Utilisez les commandes **kinit** et **klist**. Vous pouvez utiliser compte *Administrator* ou n'importe quel autre compte avec une délégation de droit permettant de joindre un poste au domaine dans la bonne OU de destination (par défaut *CN=Computers*).

Dans le shell retranscrit ci-dessous, les commandes sont en noir et le texte retouné est commenté en gris clair:

```
sudo kinit administrator
## Password for administrator@MYDOMAIN.LAN:
## Warning: Your password will expire in 277 days on lun. 17 sept. 2018 10:51:21 CEST
sudo klist
## Ticket cache: FILE:/tmp/krb5cc_0
## Default principal: administrator@MYDOMAIN.LAN
##
## Valid starting       Expires              Service principal
## 01/12/2017 16:49:31  02/12/2017 02:49:31  krbtgt/MYDOMAIN.LAN@MYDOMAIN.LAN
## renew until 02/12/2017 16:49:27
```

Si la requête d'authentification est réussie, vous pouvez créer votre HTTP Keytab avec la commande **msktutil**.

Assurez-vous de modifier l'entrée *<DOMAIN_CONTROLER>* avec le nom de votre contrôleur de domaine (eg: **srvads.mydomain.lan**).

```
sudo msktutil --server DOMAIN_CONTROLER --precreate --host $(hostname) -b cn=computers --service␣
↪HTTP --description "host account for wapt server" --enctypes 24 -N
sudo msktutil --server DOMAIN_CONTROLER --auto-update --keytab /etc/nginx/http-krb5.keytab --
↪host $(hostname) -N
```

Attention: Assurez-vous d'avoir correctement configuré le nom du serveur WAPT avant de lancer ces commandes;

Pour faire une double vérification de votre *hostname*, vous pouvez lancer `echo $(hostname)` et cela doit vous retourner le nom qui sera utilisé par l'agent WAPT lancé sur les postes clients.

- appliquez les bons droits d'accès au fichier `http-krb5.keytab`:

```
#Debian
sudo chmod 640 /etc/nginx/http-krb5.keytab
sudo chown root:www-data /etc/nginx/http-krb5.keytab

#CentOS
sudo chown root:nginx /etc/nginx/http-krb5.keytab
sudo chmod 640 /etc/nginx/http-krb5.keytab
```

Post-configuration

Vous pouvez désormais utiliser le script de post-configuration pour configurer Kerberos sur le serveur WAPT.

Le script de post-configuration va configurer **Nginx** et l'utilisation de l'authentification par Kerberos sur le serveur WAPT.

Indication: Ce script de post-configuration doit être lancé en tant que **root**.

```
/opt/wapt/waptserver/scripts/postconf.sh --force-https
```

L'authentication par Kerberos va maintenant être configuré.

Cas spéciaux d'usage

Mon serveur WAPT n'a pas accès à un Active Directory en écriture

- connectez-vous à votre Active Directory (Pas un RODC);
- créez un compte machine *srvwapt*;
- ajoutez un champ SPN (Service Principal Name) sur le compte *srvwapt$*;

```
setspn -A HTTP/srvwapt.mydomain.lan srvwapt
```

- créez un keytab pour le serveur WAPT:

```
ktpass -out C:\http-krb5.keytab -princ HTTP/srvwapt.mydomain.lan@MYDOMAIN.LAN rndpass -
↪minpass 64 -crypto all -pType KRB5_NT_PRINCIPAL /mapuser srvwapt$@MYDOMAIN.LAN
Reset SRVWAPT$'s password [y/n]?  y
```

Note: Si l'adresse de votre serveur WAPT est différente de celle de votre domaine Active Directory, remplacez *HTTP/srvwapt.mydomain.lan@MYDOMAIN.LAN* par *HTTP/srvwapt.othername.com@MYDOMAIN.LAN*.

- transférez ce fichier vers `/etc/nginx/` (avec **`winscp`** par exemple);
- appliquez les bons droits d'accès au fichier `http-krb5.keytab`:

```
#Debian
sudo chmod 640 /etc/nginx/http-krb5.keytab
sudo chown root:www-data /etc/nginx/http-krb5.keytab

#CentOS
sudo chown root:nginx /etc/nginx/http-krb5.keytab
sudo chmod 640 /etc/nginx/http-krb5.keytab
```

L'agent WAPT a uniquement un accès vers un contrôleur de domaine RODC

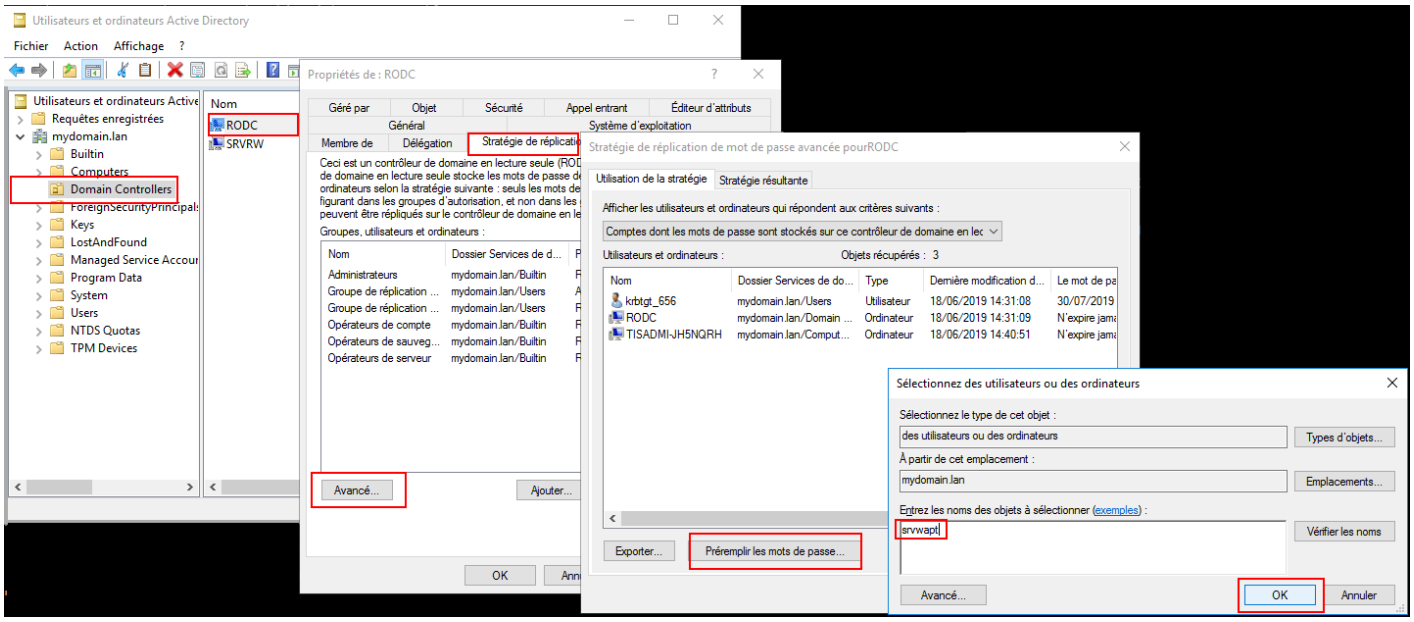
- pour un RODC (Read-Only Domain Controller), ajoutez le compte *srvwapt* » au groupe autorisé à répliquer les mots de passe;
- n'oubliez pas de pré-charger le mot de passe du serveur WAPT avec les différents serveurs RODC;

Vous avez de multiples domaines Active Directory avec ou sans relations

Si vous avez des domaines Active Directory multiples, vous devez créer un `keytab` par domaine en suivant la procédure ci-dessus, ex :

- `http-krb5-domain1.local.keytab` ;
- `http-krb5-domain2.local.keytab` ;
- `http-krb5-domain3.local.keytab` ;

Vous allez alors fusionner tous ces `keytabs` en un unique `keytab`:




```
ktutil
read_kt http-krb5-domain1.local.keytab
read_kt http-krb5-domain2.local.keytab
read_kt http-krb5-domain3.local.keytab
write_kt http-krb5.keytab
```

6.12.3 Activer la vérification du certificat SSL / TLS

Lorsque le script post-configuration est lancé sur le serveur WAPT, le script va générer un certificat auto-signé afin d'activer les communications HTTPS.

L'agent WAPT vérifie que le certificat du serveur renseigné dans l'option `verify_cert` de la section `[global]` dans le fichier `C:\Program Files (x86)\wapt\wapt-get.ini` corresponde.

Table26: Options pour « verify_cert »

Options pour « verify_cert »	Principe de fonctionnement de l'agent WAPT
`verify_cert` = 0	l'agent WAPT ne vérifiera pas le certificat HTTPS du serveur WAPT
`verify_cert` = 1	l'agent WAPT vérifiera le certificat HTTPS du serveur avec le *bundle* de certificats `C:\Program Files (x86)\wapt\lib\site-packages\certifi\cacert.pem`
`verify_cert` = C:\Program Files (x86)\wapt\ssl\srvwapt.mydomain.lan.crt	l'agent WAPT vérifiera le certificat HTTPS du serveur WAPT avec le *bundle* de certificats `C:\Program Files (x86)\wapt\ssl\srvwapt.mydomain.lan.crt`

Indication: Pour activer rapidement et simplement la vérification du certificat htts, vous pouvez utiliser la méthode d'*Épinglage*

Épingler le certificat

L'*épinglage de certificat* consiste à vérifier le certificat SSL/ TLS à l'aide de la définition d'un paquet bien défini et restrictif.

Indication: Cette méthode est la plus simple concernant les certificats auto-signés.

Pour cela, vous aurez besoin de lancer la commande suivante dans la console Windows **cmd.exe** (avec élévation de privilèges si les UAC (User Account Control) sont activées).

Si vous avez déjà une console Windows **cmd.exe** d'ouverte, fermez-là et ouvrez-en une nouvelle afin de prendre en compte la mise à jour des variables d'environnement:

```
wapt-get enable-check-certificate
net stop waptservice
net start waptservice
```

Validez le certificat avec **wapt-get update**

Lorsque vous avez exécuté la commande **update**, assurez-vous que tout ce soit bien déroulé et en cas de doute, vérifiez *Problème lors du enable-check-certificate*.

Note:

- La commande *enable-check-certificate* va télécharger le certificat `srvwapt.mydomain.lan.crt` dans le dossier `C:\Program Files (x86)\WAPT\ssl\server`;
- cela va alors modifier le fichier `wapt-get.ini` avec de spécifiée la valeur `verify_cert = C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt`;
- l'agent WAPT va désormais vérifier les certificats en utilisant le certificat épinglé;

Attention: Si vous utilisez la méthode d'*épinglage de certificat*, n'oubliez pas de sauvegarder le dossier `/opt/wapt/waptserver/ssl` sur votre serveur WAPT.

Le fichier devra être restauré sur votre serveur si vous migrez ou mettez à jour votre seveur WAPT, si vous voulez que les agents WAPT continuent d'avoir une connexion HTTPS sécurisée.

Comment utiliser le ou les certificat(s) founi(s) par votre organisation ?

Si la méthode d'épinglage ne vous convient pas, vous pouvez remplacer le certificat auto-signé généré pendant l'installation de **WAPT**.

Remplacez l'ancien certificat avec le nouveau dans le dossier `/opt/wapt/waptserver/ssl/` (linux) ou `c:\wapt\waptserver\ssl\` (windows).

La nouvelle paire de clé doit être une PEM encodée en Base64

Note: Cas spécifique où votre certificat a été signé par un Certificat d'Authorité interne

Les certificats issus d'un *Autorité de Certification* interne doit avoir la chaîne complète jusqu'au *Autorité de Certification*.

Vous pouvez ajouter la chaîne du Certificat d'Authorité manuellement au certificat qui sera utilisé par **Nginx**. »

Exemple : `echo srvwapt.mydomain.lan.crt ca.crt > cert.pem`

Pour les serveurs Linux, il est nécessaire d'aussi réinitialiser les ACLs (Access Control List):

```
#Debian:
chown root:www-data /opt/wapt/waptserver/ssl/*.pem

#Centos:
chown root:nginx /opt/wapt/waptserver/ssl/*.pem
```

- redémarrer **Nginx** pour qu'il prenne en compte les nouveaux certificats;
  - Linux:

    ```
    systemctl restart nginx
    ```

  - Windows:

    ```
    net stop waptnginx
    net start waptnginx
    ```

Configurer l'agent WAPT

Pour un certificat commercial, vous pouvez paramétrer `verify_cert` = 1 dans `wapt-get.ini`.

Pour un certificat issu d'un Certificat d'Authorité interne, vous devez placer le certificat dans le dossier `C:\Program Files (x86)\wapt\ssl\server\ca.crt` et renseigner le chemin du certificat dans `verify_cert` dans le `wapt-get.ini`. de l'agent.

Pour appliquer la nouvelle configuration à tout votre parc, vous pouvez regénérer unagent WAPT avec les options appropriées.

Vérifier les certificats dans la console WAPT

Lorsque la console WAPT se lance pour la première fois, elle va lire le contenu de `C:\Program Files (x86)\WAPT\wapt-get.ini` et va créer son fichier de configuration dans `C:\Users\admin\AppData\Local\waptconsole\waptconsole.ini`.

Cela va configurer correctement l'attribut `verify_cert` pour les communications HTTPS entre la console WAPT et le serveur WAPT.

6.12.4 Configurer l'authentification par Active Directory

Nouveau dans la version 1.5: Enterprise

Indication: Fonctionnalité uniquement disponlible dans le version **Enterprise** de WAPT

Par défaut, le serveur WAPT est configuré avec un seul compte *SuperAdmin* dont le mot de passe est configuré lors du premier post-configuration.

Sur un vaste réseau sécurisé, ce compte *SuperAdmin* ne devrait pas être utilisé puisqu'il ne peut fournir une traçabilité nécessaire sur les actions d'administration menées sur le réseau.

Il est donc nécessaire de configurer l'authentification par l'Active Directory de l'*Organisation* pour les *Administrateurs* et les *Déployeurs de Paquets*; cela permettra d'utiliser des comptes nominatifs pour les tâches d'administration.

Note:

- L'authentification par Active Directory est utilisé pour accéder à l'inventaire via la console WAPT;
- cependant, toutes les actions menées sur les appareils équipés de WAPT sont basées sur les signatures X.509, un *Administrateur* aura donc besoin d'un compte dans l'Active Directory **ET** d'une clé privée dont le certificat est reconnu par l'appareil avec WAPT qui va piloter le parc;
- seul le compte *SuperAdmin* et les membres du groupe **waptadmins** dans l'Active Directory pourra téléverser des paquets dans le répertoire principal (le mode d'authentification se fait par login et mot de passe);

Activer l'authentification par Active Directory

- pour activer l'authentification par Active Directory sur le serveur WAPT, configurez les fichier `/opt/wapt/conf/waptserver.ini` comme ceci:

```
wapt_admin_group_dn=CN=waptadmins,OU=groupes,OU=tranquilit,DC=mydomain,DC=lan
ldap_auth_server=srvads.mydomain.lan
ldap_auth_base_dn=DC=mydomain,DC=lan
ldap_auth_ssl_enabled=False
```

Réglages	Valeur	Description
wapt_admin_group_dn	CN=waptadmins,OU=groups,	DN du nom du groupe. Tous les membres de ce groupe pourra se connecter à WAPT
ldap_auth_server	srvads.mydomain.lan	Le serveur LDAP qui sera utilisé par WAPT
ldap_auth_base_dn	DC=mydomain,DC=lan	DN pour la recherche
ldap_auth_ssl_enable	True/False	Valeur par défaut : True

- redémarrez **`waptserver`** avec `systemctl restart waptserver`;

Avertissement: Pour Microsoft Active Directory, Microsoft a annoncé que l'authentification *SimpleBind* sur MS-AD sans SSL/TLS sera bloquée par défaut à partir d'avril 2020. Si vous n'avez pas de certificat installé, vous devrez modifier une clé de registre pour que l'authentification fonctionne.

Note: Par défaut, Samba-AD ne permet pas l'authentification *SimpleBind* sans SSL/TLS. Si vous ne disposez pas d'un certificat valide, vous devrez modifier le paramètre « Serveur d'authentification forte » dans le fichier `/etc/samba/smb.conf`. Pour plus d'informations, vous pouvez consulter la documentation de Tranquil IT sur https://dev.tranquil.it/samba/fr/index.html.

Activez le support SSL/ TLS pour les connexions LDAP dans le Contrôleur de Domaine Active Directory

Par défaut, l'authentification d'un Active Directory se repose sur du LDAP SSL (port par défaut 646).

SSL /TLS n'est pas activé par défaut sur un Microsoft Active Directory jusqu'à ce qu'un certificat SSL soit configuré sur le Contrôleur de Domaine.

Note: Le serveur WAPT utilise des *paquets* de Certificat d'Authorité du système d'exploitation (CentOS) pour valider la connexion SSL /TLS vers l'Active Directory.

Si le certificat Active Directory est auto-signé ou a été signé par un CA interne, vous aurez besoin d'ajouter ces certificats au magasin de certificat de CentOS.

Ajouter un *Autorité de Certification* dans le dossier `/etc/pki/ca-trust/source/anchors/` et mettez à jour le magasin des CA.

```
cp cainterne.pem /etc/pki/ca-trust/source/anchors/cainterne.pem
update-ca-trust
```

- une fois que le LDAP SSL/ TLS sur votre Active Directory est configuré (veuillez vous référer à la documentation Microsoft), vous pouvez activer le support pour la sécurité SSL /TLS pour AD dans `/opt/wapt/conf/waptserver.ini`:

  ```
  ldap_auth_ssl_enabled = True
  ```

- redémarrez **`waptserver`** avec `systemctl restart waptserver`;

6.12.5 Configurer l'authentification par Certificat sur le Client

Nouveau dans la version 1.7: Enterprise

Indication: Cette fonctionnalité n'est disponible qu'en version **Enterprise** de WAPT.

Si votre entreprise a besoin d'un serveur WAPT ouvert sur Internet, il peut être securisé grâce à l'**authentification par Certificat sur le Client**.

Cette configuration restreint la visibilité du serveur WAPT aux clients enregistrés uniquement. Cela se base sur la clé privée de l'agent généré lors de son enregistrement. Il marche comme ce qui suit:

- l'agent WAPT envoie un CSR (Certificate Signing Request) au serveur WAPT que ce dernier va signer et renvoyer à l'agent WAPT;
- en utilisant le certificat signé, l'agent peut accéder à l'arborescence protégée du serveur web **`Nginx`**;

Note: Nous vous recommandons d'activer Kerberos ou bien l'enregistrement par identifiant / mot de passe dans le post-configuration du serveur WAPT.

Activer l'authentification par Certificat sur le Client

- assurez-vous de désactiver les en-têtes personnalisées relatives aux résultats de l'authentification côté client lorsque les demandes sont passées au proxy sans être vérifiées par le module ssl de nginx. Ces en-têtes sont jugées fiables par le waptserver si *X-Ssl-Authenticated* est marqué *SUCCESS* et que le paramètre `waptserver.ini` `use_ssl_client_auth` est réglé sur **True** :

```
location / {
    ...
    proxy_set_header X-Ssl-Authenticated "";
    proxy_set_header X-Ssl-Client-DN "";
```

- Ajoutez une configuration **`Nginx`** dans le fichier `/etc/nginx/certificate-auth.conf`. Ce fichier est utilisé pour restreindre l'accès aux actions propres au Certificat d'Authentification:

```
proxy_set_header X-Ssl-Authenticated $ssl_client_verify;
proxy_set_header X-Ssl-Client-DN $ssl_client_s_dn;
if ($ssl_client_verify != SUCCESS) {
    return 401;
}
```

Exemple d'un fichier de configuration:

```
server {
    listen                      80;
    listen                      443 ssl;
    server_name                 _;

    ssl_certificate             "/opt/wapt/waptserver/ssl/cert.pem";
    ssl_certificate_key         "/opt/wapt/waptserver/ssl/key.pem";
    ssl_protocols               TLSv1.2;
    ssl_dhparam                 /etc/ssl/certs/dhparam.pem;
    ssl_prefer_server_ciphers   on;
    ssl_ciphers                 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
    ssl_stapling                on;
    ssl_stapling_verify         on;
    ssl_session_cache           none;
    ssl_session_tickets         off;

    gzip_min_length     1000;
    gzip_buffers        4 8k;
    gzip_http_version   1.0;
    gzip_disable        "msie6";
    gzip_types          text/plain text/css application/json;
    gzip_vary           on;

    ssl_client_certificate "/opt/wapt/conf/wapt-serverauth-ca.crt";
    ssl_verify_client optional;

    index index.html;

    location /static {
        alias "/opt/wapt/waptserver/static";
    }
```

(suite sur la page suivante)

(suite de la page précédente)

```
    location / {
        proxy_set_header X-Real-IP  $remote_addr;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        # be sure we ignore these headers if they are coming from clients
        proxy_set_header X-Ssl-Client-Dn  "";
        proxy_set_header X-Ssl-Authenticated  "";

        client_max_body_size 4096m;
        client_body_timeout 1800;

        location ~ ^/(wapt|wapt-host|waptwua)/(.*)$ {
            proxy_set_header Cache-Control "store, no-cache, must-revalidate, post-check=0, pre-
↪check=0";
            proxy_set_header Pragma "no-cache";
            proxy_set_header Expires "Sun, 19 Nov 1978 05:00:00 GMT";

            include /etc/nginx/certificate-auth.conf;

            rewrite ^/(wapt|wapt-host|waptwua)/(.*)$ /$1/$2 break;
            root "/var/www";
        }

        # kerberos auth
        location /add_host_kerberos {
            auth_gss on;
            auth_gss_keytab  /etc/nginx/http-krb5.keytab;
            proxy_pass http://127.0.0.1:8080;
        }

        # basic auth
        location ~ ^/(add_host|ping)$ {
            proxy_pass http://127.0.0.1:8080;
        }

        location /wapt-host/Packages {
                return 403;
        }

        location / {
            include /etc/nginx/certificate-auth.conf;
            proxy_pass http://127.0.0.1:8080;

        }

        location /socket.io {
            include /etc/nginx/certificate-auth.conf;
            proxy_http_version 1.1;
            proxy_buffering off;

            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
            proxy_pass http://127.0.0.1:8080/socket.io;
```

(suite sur la page suivante)

(suite de la page précédente)

```
            }
        }

}
```

Attention: Soyez prudent, en date du 2023-01-10, WAPT ne supporte pas les CRL, ce qui signifie que lorsque vous supprimez une machine dans la console WAPT, la machine pourra toujours accéder au répertoire WAPT.

6.13 Améliorer votre productivité avec WAPT

Vous trouverez dans cette section de la documentation quelques méthodes inspirantes qui vous feront gagner du temps en vous permettant de tirer le meilleur de votre installation WAPT.

6.13.1 Simplifier la préparation de vos postes de travail

Nous observons que de nombreuses entreprises et administrations intègrent des logiciels et des configurations dans les images Windows qu'elles déploient sur leurs parcs de machines.

Si vous utilisez WAPT, perdez cette habitude maintenant et pour toujours ! Pourquoi ?

- Chaque fois que vous créez une nouvelle image, vous perdez beaucoup de temps à installer des logiciels et à les configurer. Vous êtes très limité dans les configurations utilisateur que vous pourrez inclure dans votre image.
- chaque fois que vous créez une nouvelle image, si vous êtes sérieux, vous devrez suivre les changements dans un document texte, un tableur ou un outil de gestion des changements. C'est une charge très lourde et ingrate. Et vous savez aussi bien comme moi que ce qui est ingrat est généralement mal fait !
- enfin, si vous introduisez dans votre image des configurations de sécurité, des configurations réseau, ou des configurations pour limiter l'intrusion de la télémétrie Windows, ces configurations peuvent perturber le fonctionnement normal du WAPT, cela compliquera les diagnostics futurs, et cela vous découragera d'utiliser un outil efficace très capable de vous libérer de grandes quantités de temps et d'efforts.

Que proposez-vous de faire alors ?

Tranquil IT recommande :

- pour ne faire qu'une seule image brute par type de système d'exploitation avec MDT ou Fog (win10, win2016, etc) sans aucune configuration ni logiciel. **Déposez uniquement les pilotes système** dont vous avez besoin pour le déploiement de votre image dans les répertoires MDT ou Fog prévus à cet effet ;
- configurer votre serveur WAPT afin d'enregistrer les hôtes avec un UUID aléatoire pour éviter les conflits *UUID Bios ou FQDN* ;
- créer autant d'Unités Organisationnelles que vous avez de types de machines dans l'UO *CN=Computers* (ex : *standard_laptop*, *hardened_laptop*, *workstations*, *servers*, etc) dans votre Active Directory ;
- configurer votre Active Directory afin de distribuer la GPO de l'agent WAPT aux différentes Unités Organisationnelles machine ; de cette façon, vous pouvez opter pour des configurations fines de votre `waptagent.ini` pour les hôtes rattachés à chaque UO.

Note: Alternativement, vous pouvez inclure un agent WAPT générique dans l'image de votre système d'exploitation.

- configurer correctement votre DHCP afin de rediriger le PXE vers les images des systèmes d'exploitation ;
- configurer correctement votre MDT ou votre Fog afin d'enregistrer la machine dans la bonne Unité Organisationnelle de votre Active Directory ;
- créer autant de paquets de configuration de sécurité WAPT que vous avez d'Unités Organisationnelles créées ci-avant. Ainsi, vous pourrez appliquer différents profils de sécurité selon le type de machine. Ces paquets comprendront les configurations de sécurité souhaitées (suppression de la télémétrie, configuration du pare-feu, etc) ;

Indication: Pour vous faire gagner du temps, vous pouvez baser votre stratégie de configuration de sécurité sur les paquets de sécurité WAPT déjà disponibles dans le store WAPT, il vous suffira de les compléter en fonction des exigences de sécurité spécifiques de votre organisation.

- créer dans l'UO *CN=Computers* autant d'Unités Organisationnelles qu'il y a de typologies d'usage des ordinateurs dans votre organisation (*accounting*, *point_of_sale*, *engineering*, *sedentary_sales*, etc) ;
- créer des paquets WAPT génériques de vos applications logicielles avec leurs configurations associées ;

Note: Pour vous faire gagner du temps et des efforts, vous pouvez importer de nombreux paquets WAPT éprouvés depuis les stores publics de Tranquil IT ou vous abonner aux stores privés de Tranquil IT.

- associer les paquets WAPT créés ci-dessus aux Unités Organisationnelles des typologies d'utilisation des ordinateur ;

Comment le scénario fonctionne-t-il ?

- vous recevez ou le responsable informatique du site distant reçoit une nouvelle machine dans sa boîte ;

Indication: Alternativement, vous choisissez ou le responsable informatique du site distant choisit de faire passer une machine existante de win7 à win10. Vous aurez, ou il aura préalablement sauvegardé le(s) répertoire(s) de l'utilisateur sur un lecteur réseau ou un autre support de stockage pratique.

À cette fin, vous pouvez construire un paquet WAPT qui, à l'exécution, zippera le fichier `C:\Users` sur l'ordinateur win7, le nommera avec le FQDN de l'ordinateur, protégera par mot de passe le fichier compressé en utilisant *cette procédure* et le chargera sur un serveur web ou un partage réseau. Ce même paquet WAPT peut effectuer le processus inverse et réinstaller les fichiers utilisateur après que l'hôte ait été ré-imagé.

- vous configurez le MDT ou le Fog avec l'adresse MAC de la machine afin qu'elle obtienne la bonne image du système par DHCP et qu'elle se positionne dans la bonne Unité Organisationnelle à la fin du processus de clonage ;
- l'image système attendue est téléchargée sur la machine en temps masqué, la machine est placée dans la bonne Unité Organisationnelle ;
- l'agent WAPT enregistre la machine auprès du serveur WAPT, la machine apparaît dans la console WAPT ;

Indication: Si vos machines proviennent d'une mise à jour de win7 à win10, alors vous retirerez les anciennes machines win7 de l'inventaire WAPT car elles seront dupliquées en raison de votre choix de configuration UUID aléatoire ; ces machines seront faciles

à trouver dans la console WAPT car elles seront marquées comme win7 avec la même adresse MAC ou le même FQDN que votre nouvelle machine win10 ; après avoir retiré le win7, votre inventaire sera propre et à jour dans votre console WAPT ;

- l'agent WAPT détecte qu'il se trouve dans une Unité Organisationnelle qui nécessite un ensemble de logiciels et une configuration de sécurité particuliers ;
- l'agent WAPT télécharge et exécute des progiciels et des configurations de sécurité en temps masqué ; l'agent WAPT supprime automatiquement les droits délégués qui sont rendus inutiles après avoir rejoint le domaine pour éviter qu'ils ne soient ensuite exploités de manière non autorisée ;
- soit par groupe de machines, soit machine par machine, vous finalisez la configuration des machines en leur attribuant des paquets WAPT spécifiques ;

Indication: Si vous le souhaitez, vous pouvez même laisser l'étape de configuration finale à vos utilisateurs en configurant le libre-service WAPT pour eux (configuration des imprimantes, besoins logiciels spéciaux, etc).

Conclusion

Avec peu d'efforts, vous avez maintenant le contrôle total d'une flotte de plusieurs centaines, voire milliers de machines géographiquement dispersées. Toutes vos installations sont documentées, vos utilisateurs travaillent avec des droits adéquats et vous bénéficiez d'une visibilité claire sur les outils et les usages de vos utilisateurs. Ainsi, le passé n'est plus un fardeau impondérable pour vous et un obstacle à vos projets futurs.

6.14 Créer des paquets WAPT

Attention: L'installation correcte de *tis-pyscripter*, le paquet qui installe l'*IDE* **`PyScripter`**, est **primordiale** avant d'entreprendre toute création ou modification de paquet.

Veillez à consulter préalablement la documentation relative à la *mise en place de l'environment de développement PyScripter*.

Pour créer vos paquets et les personnaliser, suivez cette documentation et vous maîtriserez très vite les subtilités du fonctionnement de WAPT.

6.14.1 Créer son environment de développement de paquets WAPT

Pré-requis

Attention:

- il est **impératif** d'être en possession d'un compte *Administrateur Local* de la machine pour cette opération ;
- nous vous conseillons de créer / éditer vos paquets dans un environment maîtrisé, sain et *jetable* ;
- l'utilisation d'une machine virtuelle autonome (type Virtualbox ou équivalent) est vivement recommandée ;

Vérifier la présence de la clé privée sur le poste de développement

Dans la console, les champs *clé privé* et *prefix* doivent être renseignés.

Installer le paquet *tis-pyscripter*

- importer le paquet *tis-pyscripter* dans votre dépôt local et l'installer sur votre machine de développement ;

Préconisations concernant l'environment de test

La méthode préconisée pour tester correctement vos paquets est d'utiliser un échantillon de machines représentatif de votre parc. Donc plus votre parc est hétérogène, plus votre échantillon devra être large.

Cette démarche vise à confronter le paquet WAPT à une multitude de plateformes et d'environments afin qu'il devienne le plus abouti possible en régime de test, avant d'être basculé en production.

Démarche de test

Systèmes d'exploitation et architectures

- Windows XP ;
- Windows 7 ;
- Windows 10 ;
- Windows Server 2008 R2 ;
- Windows Server 2012 ;
- x86 ;
- x64 ;
- Machine physique et virtuelle ;
- PC portable ;

Indication: On testera si possible les versions RC / Beta des OS si elles sont disponibles (exemple : Windows 10 Creators Update).

État des mises à jour Windows Update

- **Poste Microsoft Windows sans aucune mise à jour Windows Update** : l'objectif est de détecter les mises à jour indispensables au bon fonctionnement du logiciel et adapter le paquet en conséquence ;
- **Poste Microsoft Windows à jour avec les toutes dernières MàJ Windows Update** : l'objectif est de détecter les mise à jour en conflit avec le logiciel et d'adapter le paquet en conséquence ;

État des installations logicielles

- **Poste avec peu de logiciels déjà installés** : l'objectif est de détecter une dépendance possible à Java ou autre framework applicatif ;
- **Poste avec beaucoup de logiciels déjà installés** : l'objectif est de détecter un conflit avec une application existante ;
- **Installer les anciennes versions du logiciel** : il est possible que l'installeur ne supporte pas l'écrasement d'une installation précédente, dans ce cas il faudra prévoir la désinstallation des anciennes versions avant d'installer la nouvelle version ;

6.14.2 Créer un paquet WAPT depuis la console

Nouveau dans la version 1.3.12.

Indication: Pré-requis : Pour créer des paquets à partir de la console, il faut d'abord avoir installé l'environment de développement WAPT *tis-waptdev*.

Créer un paquet WAPT depuis la console

Dans cet exemple, l'installeur de 7zip est utilisé au format MSI.

- obtenir l'installeur MSI de 7zip ;
  - télécharger 7-zip MSI x64 ;
  - télécharger 7-zip MSI x86 ;
- créer le modèle de paquet depuis l'installeur ;

  Dans la console WAPT, cliquer sur *Outils* → *Créer un modèle de paquet depuis un installeur*.

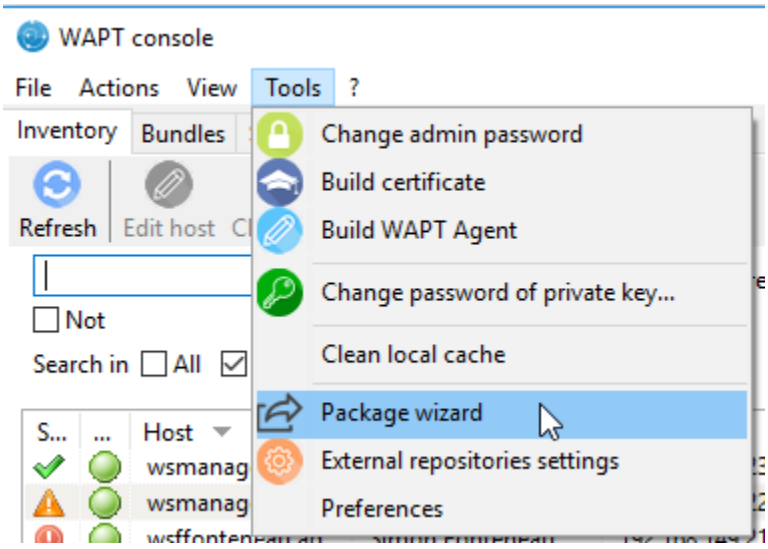


Figure146: PyScripter - Menu pour la création de modèle de paquets depuis la console

Sélectionner l'installeur MSI téléchargé et renseigner les différentes informations demandées. Veillez bien à ce que le nom du paquet ne contienne pas de numéro de version.

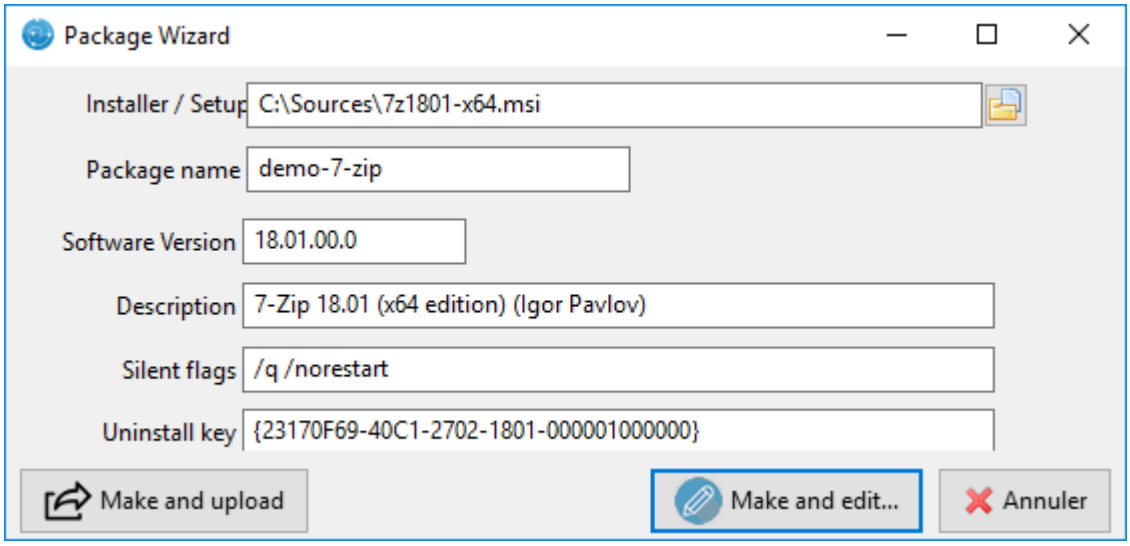


Figure147: PyScripter - Renseignements nécesssaires pour la création du paquet

- deux solutions sont proposées :
  - cliquer sur *OK* (recommandée) pour lancer la personnalisation du paquet ;
  - cliquer sur *Build and upload* pour lancer la création et le chargement direct du paquet sur le serveur WAPT (non recommandé) ;

> **Attention:** Le bouton *Build and upload* envoi directement le paquet dans le dépôt privé sans tester l'installation.
>
> Cette méthode fonctionne assez bien avec les MSI car leur installation est standard.
>
> Cependant la deuxième méthode qui consiste à tester localement le paquet d'abord puis à l'uploader est la méthode recommandée.

Personnaliser le paquet avant build-upload

La méthode conseillée avant l'"**`upload`** d'un paquet est de personnaliser son comportement en l'éditant avec **`PyScripter`**.

Lors de la création du modèle de paquet, cliquer sur *OK*.

L'IDE **`PyScripter`** se lance et permet d'éditer les fichiers du paquet.

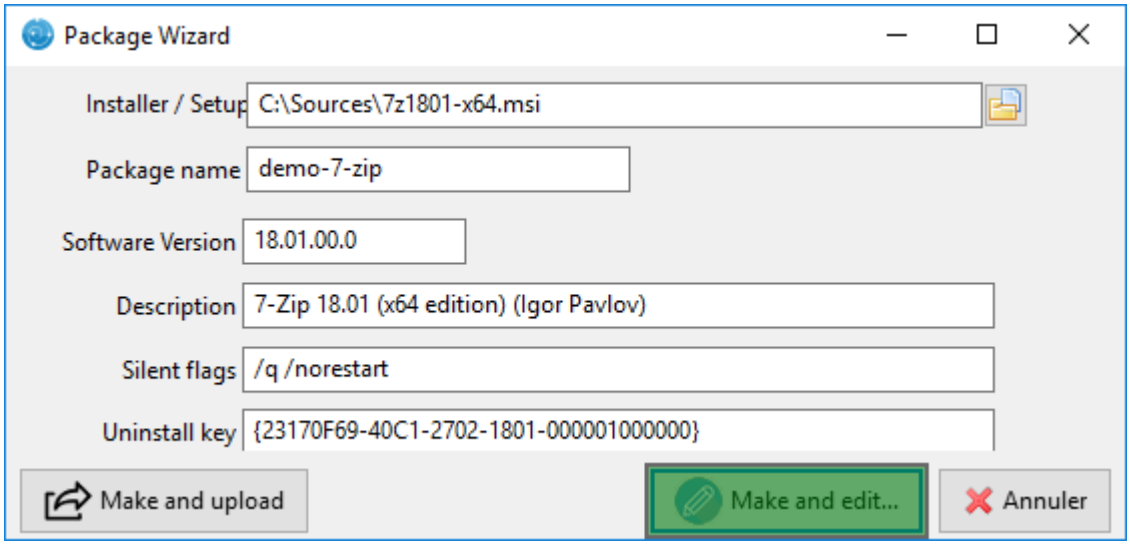


Figure148: PyScripter - Renseignements nécésssaires pour la création du paquet

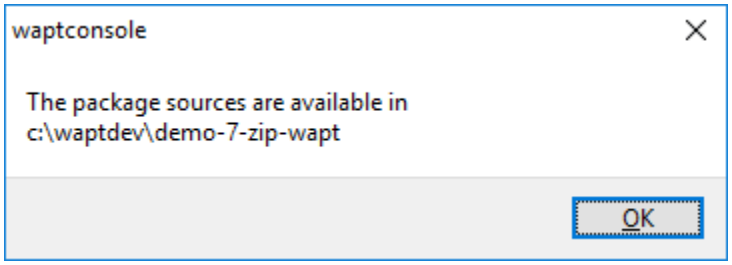


Figure149: PyScripter - Validation de la création du modèle et ouverture de PyScripter

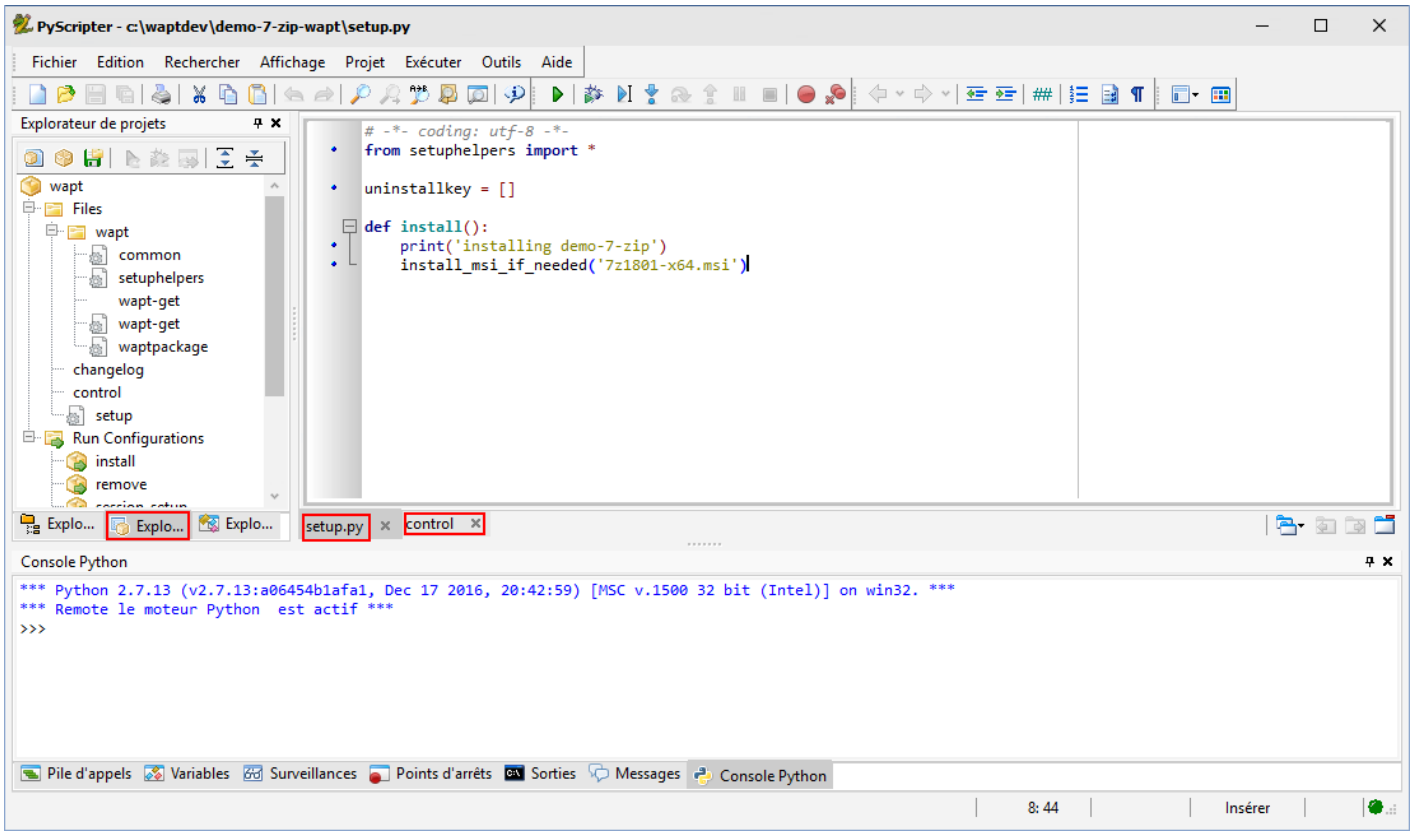


Figure150: PyScripter - Personnalisation du paquet avec PyScripter

Présentation de PyScripter

L'explorateur de projets PyScripter

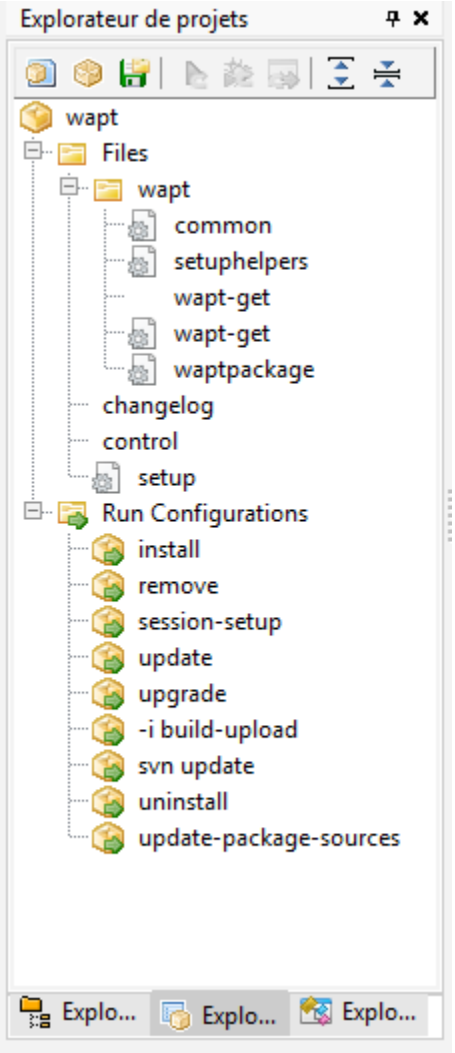


Figure151: PyScripter - Explorateur de projets PyScripter

L'explorateur de projets PyScripter liste les différents fichiers dont vous pouvez avoir besoin, notamment le fichier `control` et le fichier `setup.py`.

Run Configurations

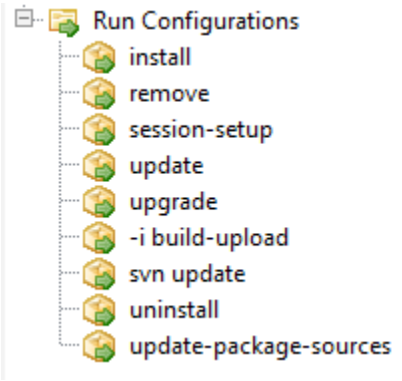


Figure152: PyScripter - Commandes Run dans l'explorateur de projets PyScripter

Les options de **Run** dans l'explorateur de projets de **PyScripter** vont vous permettre de lancer des actions de votre paquet en cours d'édition.

Zone d'édition

```
from setuphelpers import *

uninstallkey = []
vlcrc = makepath(programfiles,'VideoLAN','VLC','vlcrc')

def install():
  print("installing VLC")
  versionpaquet = control['version'].split('-',1)[0]
  if iswin64():
          install_exe_if_needed('vlc-%s-win64.exe' % versionpaquet,silentflags='/S --no-qt-privacy-ask --no-qt-updates-notif'
  else:
          install_exe_if_needed('vlc-%s-win32.exe' % versionpaquet,silentflags='/S --no-qt-privacy-ask --no-qt-updates-notif'
  filecopyto("vlcrc",vlcrc)
  remove_desktop_shortcut('VLC media player')
```

Figure153: PyScripter - zone d'édition de PyScripter

La Zone d'édition de **PyScripter** permet d'éditer le fichier `setup.py` ainsi que le fichier `control`.

Console Python

C'est la console python visible dans **PyScripter**, elle va vous permettre d'afficher la sortie python lorsque vous exécuterez des commandes **run**.

Vous pouvez également l'utiliser pour tester / déboguer des portions de votre script `setup.py`.

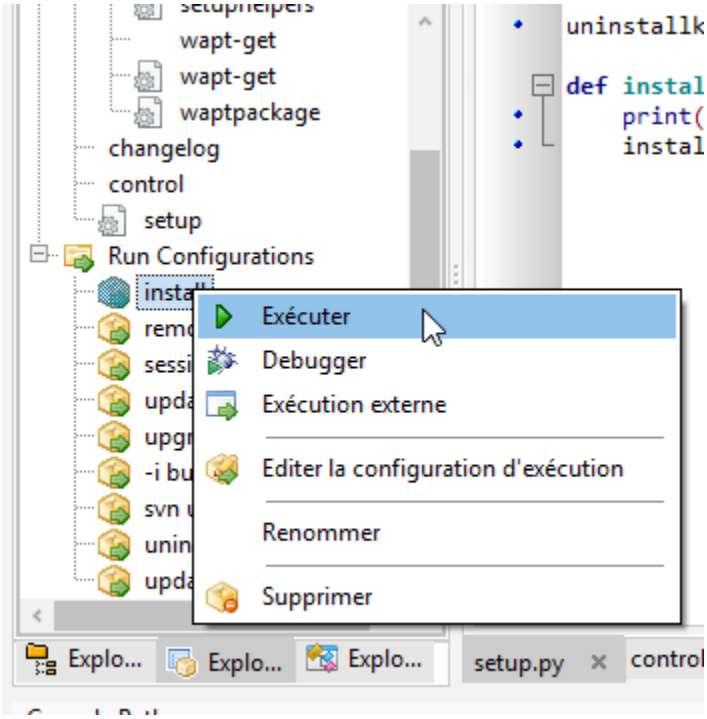
Pour en savoir plus sur la composition d'un paquet wapt, visitez la documentation sur la *structure d'un paquet WAPT*.

Si vous voulez parfaire vos paquets WAPT, visitez la documentation pour *personnaliser vos paquets*.

Figure154: PyScripter - console python de PyScripter

Tester localement l'installation du paquet WAPT

Vous pouvez ensuite tester le lancement d'une installation sur votre station de développement.



La Console PyScripter vous permet de vérifier si l'installation s'est bien déroulée.

Construire le paquet et l'envoyer au serveur WAPT

- une fois que le paquet est prêt, le construire et l'envoyer au serveur WAPT ;

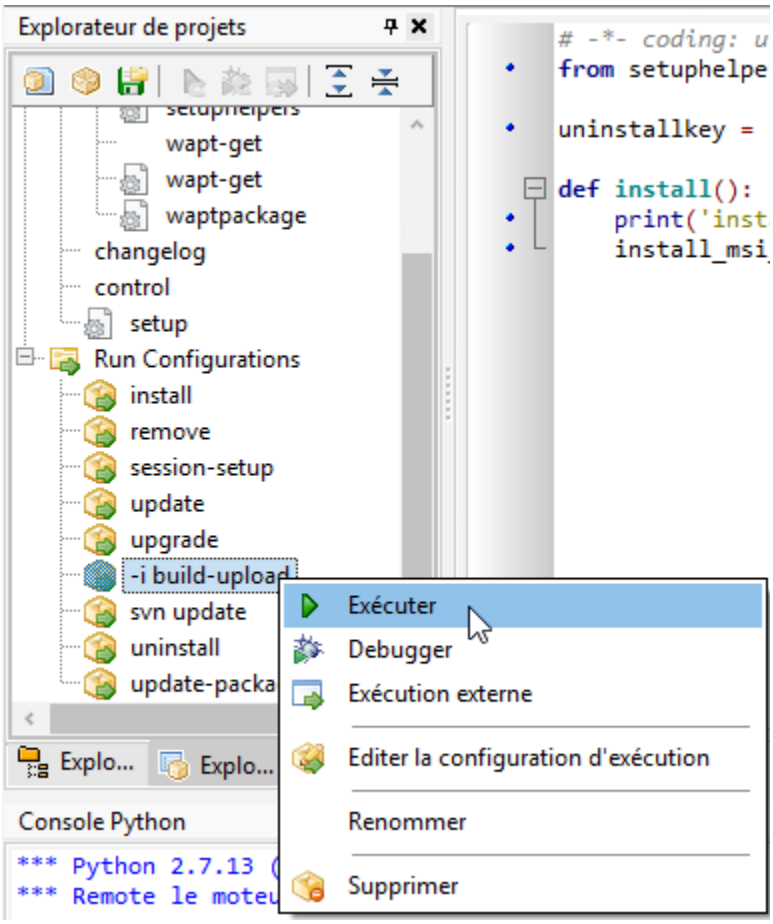
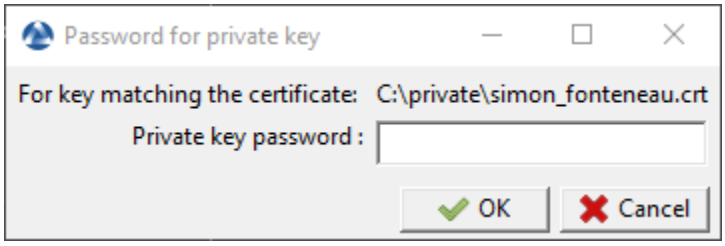
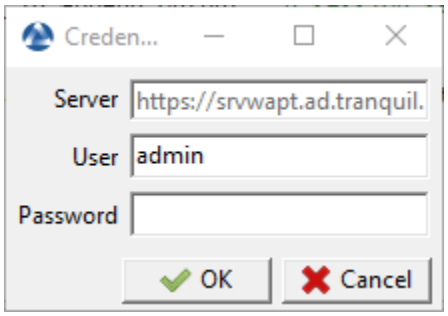


Figure155: Option « -i build-upload » du projet PyScripter

- entrer le mot de passe de votre clé privée (pour signer votre paquet WAPT) ;



- entrer le nom d'utilisateur et le mot de passe pour envoyer le paquet WAPT au serveur ;
- le paquet est maintenant disponible et visible dans la console WAPT dans l'onglet *Dépôt Privé*.
- cliquer sur *Mettre à jour la liste des paquets disponibles* pour actualiser la liste des paquets WAPT disponibles ;



6.14.3 Structure d'un paquet WAPT

Un paquet WAPT est un fichier zip qui contient plusieurs éléments :

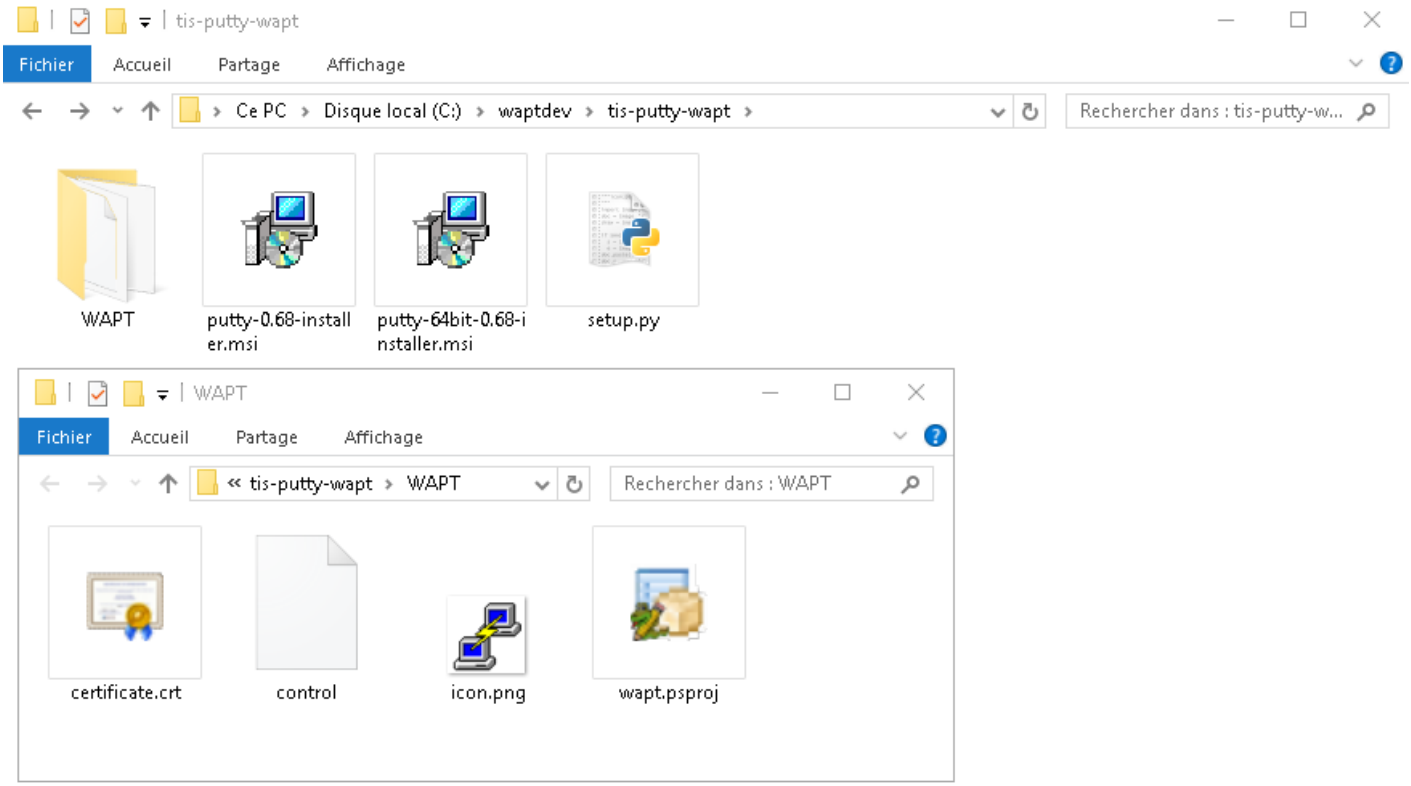


Figure156: Structure d'un paquet WAPT

- un fichier `setup.py` à la racine ;
- un fichier ou plusieurs fichiers **`binaire.exe`** à la racine ;
- un fichier ou plusieurs autres fichiers additionnels à la racine ;
- Un fichier `control` dans le dossier `WAPT` ;
- Un fichier `icon.png` dans le dossier `WAPT` ;
- Un fichier `certificate.crt` dans le dossier `WAPT` ;

- Un fichier `manifest.sha256` dans le dossier `WAPT` ;
- Un fichier `signature.sha256` dans le dossier `WAPT` ;
- un fichier `wapt.psproj` dans le dossier `WAPT`, ce fichier est utilisé pour stocker les données de configuration **`PyScripter`** pour le paquet WAPT ;
- depuis WAPT 1.8, un dossier caché `.vscode` qui contient un fichier `launch.json` et un fichier `settings.json` utilisés pour stocker les données de configuration **`VScode`** pour le paquet WAPT ;

Le fichier *control*

Le fichier `control` est la fiche d'identité du paquet.

```
package           : tis-firefox-esr
version           : 62.0-0
architecture      : all
section           : base
priority          : optional
maintainer        : Administrateur
description       : Firefox Web Browser French
description_fr    : Navigateur Web Firefox Français
description_es    : Firefox Web Browser
depends           :
conflicts         :
maturity          : PROD
locale            : fr
target_os         : windows
min_os_version    :
max_os_version    :
min_wapt_version  : 1.6.2
sources           :
installed_size    :
impacted_process  : firefox.exe
audit_schedule    :
editor            : Mozilla
keywords          : Navigateur
licence           : MPL
homepage          : https://www.mozilla.org/en-US/firefox/organizations/
signer            : Tranquil IT
signer_fingerprint: 459934db53fd804bbb1dee79412a46b7d94b638737b03a0d73fc4907b994da5d
signature         : MLOzLiz0qCHN5fChdylnvXUZ8xNJj4rEu5FAAsDTdEtQ(...)hsduxGRJpN1wLEjGRaMLBlod/
↪p8w==
signature_date    : 20170704-164552
signed_attributes : package,version,architecture,section,priority,maintainer,description,depends,
↪conflicts,maturity,locale,min_os_version,max_os_version,min_wapt_version,sources,installed_
↪size,signer,signer_fingerprint,signature_date,signed_attributes
```

Table27: Description des options du fichier control

Paramètre	Description	Valeur Exemple
package	Nom du paquet	tis-geogebra
version	Version du paquet, ne peut pas comporter plus de 5 délimiteurs	5.0.309.0-0
architecture	Architecture de processeur	x64
section	Type de paquet (host, group, base)	base
priority	Niveau de priorité d'installation du paquet (optionnel pour le moment)	Non pris en charge pour le moment
maintainer	Auteur du paquet	Gérard Bouchard mail@mydomain.lan
description	Description du paquet qui apparaîtra dans la console et sur l'interface web	The Graphing Calculator for Functions,Geometry, Algebra, Calculus, Statistics and 3D
description_fr	Description du paquet dans une langue précise	Calculatrice graphique
depends	Dépendances à **installer** avant d'installer le paquet	tis-java
conflicts	Paquets à **désinstaller** avant d'installer ce paquet	tis-graph
maturity	Niveau de maturité du paquet (*BETA*, *DEV*, *PROD*)	PROD
locale	Environment linguistique prévu pour le paquet	fr,en,es
target_os	OS Accepté par le paquet	windows,mac,linux
min_os_version	Version minimale de l'OS pour que le paquet soit vu par l'agent WAPT le paquet	6.0
max_os_version	Version maximale de l'OS pour que le paquet soit vu par l'agent WAPT le paquet	8.0
min_wapt_version	Version minimale de WAPT pour un fonctionnement correct du paquet	1.3.8
sources	Lien de stockage des versions historisées du paquet (commande **sources**)	https://srv-svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/
installed_size	Espace de disque dur libre minimal requis pour installer le paquet	254251008
impacted_process	Indique une liste de processus impactés lors de l'installation du paquet	firefox.exe
audit_schedule	Indique la périodicité pour l'exécution de la fonction audit du paquet	60
editor	Indique l'editeur du logiciel integrée dans le paquet	Mozilla
licence	Indique la licence du logiciel integrée dans le paquet	GPLV3
keywords	Indique une liste de mots clé correspondant au paquet	Bureautique,Editeur,calcul
homepage	Indique la page d'accueil du site officiel de logiciel integrée dans le paquet	https://www.tranquil.it/
signer	Nom commun (CN) du signataire du paquet	Tranquil IT
signer_fingerprint	Empreinte de la signature du signataire du paquet	2BAFAF007C174A3B00F12E9CA1E749
signature	Hash SHA256 du paquet	MLOzLiz0qCHN5fChdylnvXUZ8xNJj4r
signature_date	Indique la date de signature du paquet	20180307-230413
signed_attributes	Liste des attributs signés	package, version, architecture, section, priority, maintainer, description, depends, conflicts, maturity, locale, min_wapt_version, sources, installed_size, signer, signer_fingerprint, signature_date, signed_attributes

Note: Si le fichier `control` comporte des caractères accentués, le fichier doit être enregistré en format **UTF-8 (No BOM)**.

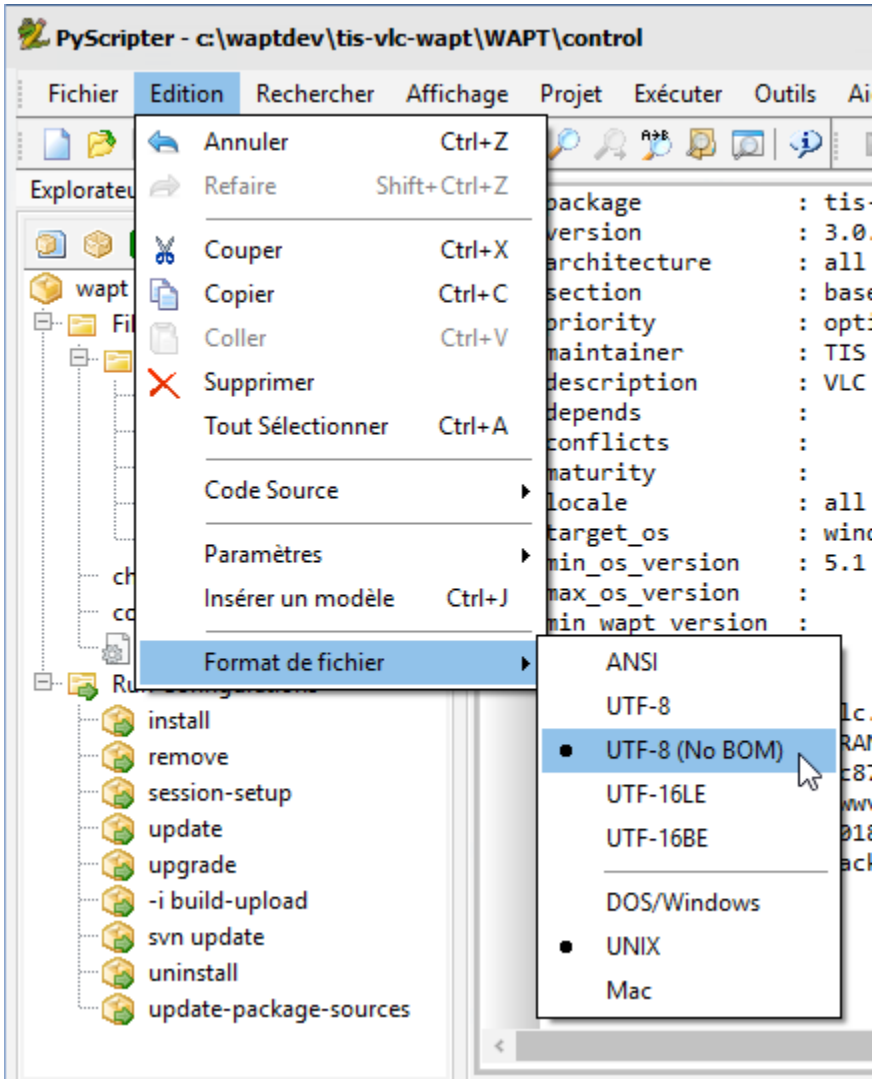


Figure157: PyScripter - UTF-8 (No BOM)

Détails des champs

package

Nom du paquet WAPT, sans accent, sans espace, sans caractère spécial, sans majuscule.

version

De préférence, toujours commencer par la version du logiciel (**chiffres uniquement**) séparée par des points (.), puis suffixer avec un tiret (-) et le numéro de version de packaging WAPT.

architecture

Nouveau dans la version 1.5.

Définit si le paquet pourra s'installer sur une machine équipée d'un processeur 32bits ou 64bits.

Note: Un paquet prévu pour une architecture x64 ne sera pas visible avec un agent WAPT installé sur un poste équipé d'un processeur x86.

Valeurs possibles :

- **x86** : le paquet est destiné uniquement aux machines avec un processeur 32bits ;
- **x64** : le paquet est destiné uniquement aux machines avec un processeur 64bits ;
- **all** : le paquet est a destination des machines avec un processeur 32bits **OU** 64bits ;

section

- **host** : paquet machine ;
- **group** : paquet groupe ;
- **base** : paquet logiciel ;
- **unit** : paquet UO ;

priority

Cette option n'est pas prise en en charge pour le moment, ce champ permettra à terme de définir la priorité d'installation d'un paquet WAPT.

maintainer

Indique le nom du créateur du paquet WAPT.

Note: indiquer l'adresse email peut être utile.

description

Décrit les fonctionnalités du paquet ; la description sera affichée dans la console, sur l'interface web des clients WAPT http://127.0.0.1:8088 et dans les lignes de commande `wapt-get.v`

Indication: Ajouter un champ description_fr ou description_es permet par exemple de préciser une description pour une langue précise. Si la langue n'existe pas, l'agent wapt utilisera la champ description classique.

depends

Définit quelle(s) dépendance(s) doit(doivent) être satisfaite(s) avant d'installer le paquet WAPT, par exemple *tis-java* devra être installé avant le paquet *LibreOffice*.

On peut définir plusieurs dépendances en les séparant par des virgules (,)

exemple :

```
depends: tis-java,tis-firefox-esr,tis-thunderbird
```

conflicts

Fonctionne exactement à l'inverse de *depends*.

conflicts définit quel(s) paquet(s) doit(vent) être désinstallé(s) avant d'installer le paquet WAPT. Par exemple si vous voulez désinstaller *firefox* avant d'installer *firefox-esr*, ou bien si vous voulez *LibreOffice*, alors il faudra désinstaller *OpenOffice* d'abord.

On peut définir plusieurs conflits en les séparant par des virgules (,).

maturity

Nouveau dans la version 1.5.1.19.

Indique la maturité du paquet.

Un agent verra par défault les paquets *PROD* et les paquet sans maturité.

Pour qu'un Poste Client puisse voir un paquet d'un autre niveau de maturité, il faudra ajouter dans `wapt-get.ini` la configuration *maturities*.

locale

Nouveau dans la version 1.5.1.19.

Indique la langue du paquet.

Un agent verra par défault les paquets dans sa langue et les paquets avec un champ local sans valeur.

Pour qu'un Poste client puisse voir un paquet d'une autre langue, il faudra ajouter dans `wapt-get.ini` la configuration *locales*.

```
locales = fr,en,es
```

La langue indiquée dans ce champ doit être en format ISO 639-1.

target_os

Nouveau dans la version 1.5.1.18.

Indique le système d'exploitation prévu pour le paquet.

Un agent verra par défault les paquets prévus pour son système d'exploitation et les paquets avec un champ `target_os` sans valeur.

Since version 1.8 the field *target_os* can either be *windows*, *mac*, *linux* or left empty.

min_os_version

Nouveau dans la version 1.3.9.

For a *windows target_os*, this field defines the minimal Windows Operating System Version. For example, this attribute may be used to avoid installing on WindowsXP packages that only work on Windows7 and above.

Since version 1.8, it can also define the minimal Mac OS version. We advise not to use it with Linux since there are several different distributions.

max_os_version

Nouveau dans la version 1.3.9.

For a *windows target_os*, it defines the maximal Windows Operating System Version. For example, this attribute may be used to install on Windows7 more recent versions of a software that are no more supported on Windows XP.

Since version 1.8, it can also define the maximal Mac OS version. We advise not to use it with Linux since there are several different distributions.

min_wapt_version

Nouveau dans la version 1.3.8.

Il s'agit de la version minimale de WAPT nécessaire pour installer du paquet.

Note: Le code de WAPT évoluant, certaines fonctions que vous aurez utilisées dans vos anciens paquets peuvent devenir obsolètes avec les nouvelles versions des agents WAPT.

sources

Définit un dépôt SVN, par exemple :

- https://svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/
- https://svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/

Cela permet de versionner le paquet et de concevoir collaborativement le paquet.

Indication: Le versionnement des paquets est très utile dans un contexte où plusieurs personnes créent des paquets de manière collaborative. Cette fonction permet également une traçabilité qui peut être utile dans un contexte réglementaire.

installed_size

Définit un espace disque libre minimum que doit avoir la machine pour lancer l'installation du paquet.

Exemple :

```
installed_size: 254251008
```

Le test d'espace disque disponible est fait sur le dossier `C:Program Files`.

La valeur renseignée dans *installed_size* doit être en bytes.

Indication: Pour convertir des valeurs de stockage en bytes, visiter https://www.convertworld.com/fr/mesures-informatiques/.

impacted_process

Nouveau dans la version 1.5.1.18.

Indique les processus impactés par le paquet.

Exemple :

```
impacted_process : firefox.exe,chrome.exe,iexplorer.exe
```

Ce champ est utilisé par les fonctions **`install_msi_if_needed`** et **`install_exe_if_needed`** si *killbefore* n'est pas renseigné.

impacted_process est également utilisé lors de la désinstallation d'un paquet. Cela permet de fermer l'application si l'application est ouverture avant sa désinstallation.

audit_schedule

Nouveau dans la version 1.6.

Indique une periodicité pour l'exécution de la fonction audit du paquet.

Exemple :

```
audit_schedule : 60
```

La valeur peut être indiquée de plusieurs manières :

- Un entier (en minutes) ;
- un entier suivi d'une lettre (*m* = minutes , *h* = heure , *d* = jour , *w* = semaine) ;

editor

Nouveau dans la version 1.6.

Indique le nom de l'éditeur du logiciel embarqué dans le paquet WAPT.

Exemple :

```
editor: Mozilla
```

La liste des valeurs pourra être utilisée dans la console et dans le selfservice pour trier les paquets.

keywords

Nouveau dans la version 1.6.

Liste de mots clés décrivant le paquet dans un objectif de classification.

Exemple :

```
keywords: editeur,bureautique,tableur
```

La liste des valeurs pourra être utilisée dans la console et dans le selfservice pour trier les paquets.

licence

Nouveau dans la version 1.6.

Indique la licence du logiciel integré dans le paquet

Exemple :

```
licence: GPLV3
```

La liste des valeurs pourra être utilisée dans la console et dans le selfservice pour trier les paquets.

homepage

Nouveau dans la version 1.6.

Indique la page d'accueil du site officiel de logiciel integrée dans le paquet.

Exemple :

```
homepage: https://wapt.fr
```

La liste des valeurs pourra être utilisée dans la console et dans le selfservice pour trier les paquets.

signer

Automatiquement renseigné lors de la signature des paquets.

Il s'agit du CN renseigné dans la clé privée du signataire du paquet. Il s'agit le plus souvent du nom complet du signataire.

signer_fingerprint

Automatiquement renseigné lors de la signature des paquets.

Il s'agit de l'empreinte de la clé privée qui a signé le paquet.

signature

Automatiquement renseigné lors de la signature des paquets.

Il s'agit de la signature des attributs du paquet.

signature_date

Automatiquement renseigné lors de la signature des paquets.

Il s'agit de la date de signature des attributs du paquet.

signed_attributes

Automatiquement renseigné lors de la signature des paquets.

Il s'agit de la liste des attributs signés.

Le fichier *setup.py*

import setuphelpers

On trouvera au début de chaque paquet cette ligne :

```
from setuphelpers import *
```

Nous demandons au paquet d'importer toutes les fonctions depuis la librairie `Setuphelpers`.

Setuphelpers est la librairie integrée à WAPT, elle embarque des fonctions simplifiées pour aider à créer des paquets.

uninstallkey list

Nous trouvons ensuite :

```
uninstallkey = ['tisnaps2','Mozilla Firefox 45.6.0 ESR (x86 fr)']
```

Nous déclarerons ici une liste des *uninstall keys* associées au paquet. Quand un paquet est supprimé, l'agent WAPT recherche dans la base de registre la *uninstallkey* correspondant au paquet. Cette *uninstallkey* indiquera à l'agent WAPT les actions à déclencher pour supprimer le logiciel.

Même s'il n'y a pas de *uninstallkey* pour le paquet, il faudra quand même déclarer une *uninstallkey* vide :

```
uninstallkey = []
```

Fonction install()

Vient ensuite la déclaration de la fonction **`setup.py`**.

C'est la recette du paquet WAPT.

```
def install():
    run('"install.exe" /S')
```

Le fichier *wapt.psproj*

On trouve dans le dossier WAPT un fichier `wapt.psproj`.

C'est le projet PyScripter du paquet WAPT.

Pour éditer un paquet avec **`PyScripter`**, il suffira d'ouvrir ce fichier.

Le fichier *icon.png*

On trouve dans le dossier WAPT un fichier `icon.png`.

Il permet d'associer une icône au paquet.

Cette icône apparaîtra dans l'interface web du self-service WAPT (http://127.0.0.1:8088).

Indication: L'icône devra être au format png 48px par 48px.

Le fichier *manifest.sha256*

On trouve dans le dossier WAPT un fichier `manifest.sha256`.

Il contient l'empreinte sha256 de chaque fichier du package.

Le fichier *control*

On trouve dans le dossier WAPT un fichier `signature`.

Le fichier contient la signature du fichier `manifest.sha256`.

A l'installation du paquet, **`wapt-get`** vérifie :

- C:\Program Files (x86)\wapt\
- que l'empreinte sha256 de chaque fichier est identique à celle contenue dans le fichier `manifest.sha256` ;

Autres fichiers

D'autres fichiers peuvent être embarqués dans le paquet WAPT. Par exemple :

- un installeur à côté du `setup.py` à appeler dans votre **`setup.py`** ;
- un fichier de réponses à passer à l'installeur du logiciel ;
- un fichier de licence ;

6.14.4 Packager des paquets .msi simples

Indication: Pré-requis

Pour la création de paquet il faut d'abord avoir *installé l'environment de développement WAPT* ;

Indication: Depuis la version 1.3.12 de WAPT, une nouvelle méthode de création rapide de paquets depuis la console WAPT est devenue disponible. Cette documentation est toujours d'actualité, nous vous conseillons cependant d'"*utiliser l'interface graphique pour instancier vos modèles de paquets*.

- Télécharger l'installeur MSI de TighVNC
  - télécharger TightVNC MSI x64 ;
  - télécharger TightVNC MSI x86 ;
- rechercher la documentation associée pour les flags silencieux ;
  - documentation TightVNC ;

```
msiexec /i tightvnc-2.7.1-setup-64bit.msi /quiet /norestart
```

Cette commande devrait installer **TightVNC** avec les paramètres par défaut. Cependant, MSI vous permet de personnaliser son installation grâce aux propriétés de MSI. La syntaxe générale est :

```
msiexec /i tightvnc-2.7.1-setup-64bit.msi /quiet /norestart PROPERTY1=value1␣
↪PROPERTY2=value2 PROPERTY3=value3
```

- lancer une invite de commande Windows **cmd.exe** en tant qu'"*Administrateur Local* ;
- instancier le paquet à partir du modèle prévu pour un installeur MSI ;

```
wapt-get make-template c:\download\file.msi <yourprefix>-tightvnc
```

Exemple avec **TightVNC** :

```
wapt-get make-template C:\Users\User\Downloads\tightvnc-2.8.5-gpl-setup-64bit.msi tis-tightvnc

Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\tis-tightvnc-wapt
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\tis-tightvnc-wapt
```

Personnaliser le paquet WAPT

PyScripter s'ouvre alors avec le projet TightVNC prêt à être davantage personnalisé.

Note: La procédure de création automatique de ce modèle de paquet :

- il crée le dossier du paquet dans `C:\waptdev` ;
- il copie le fichier MSI dans ce dossier ;

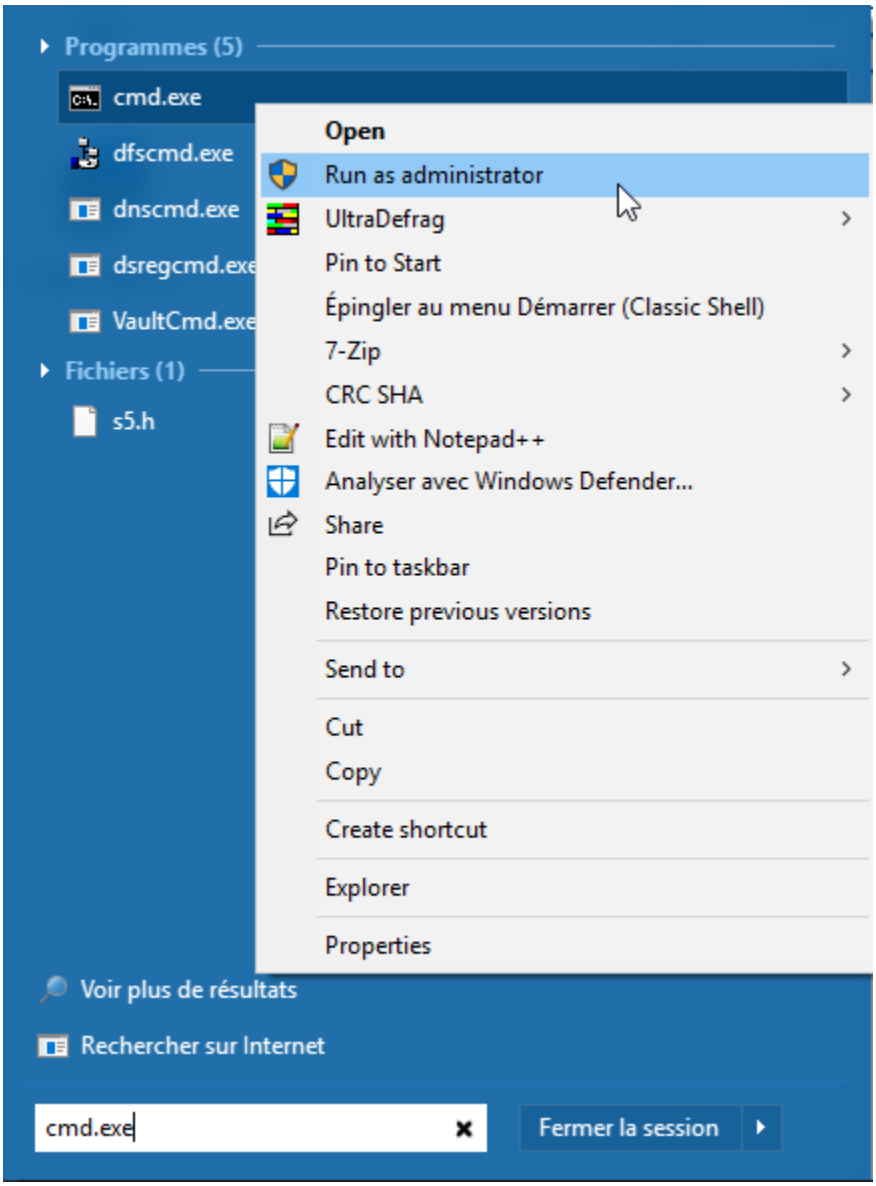


Figure158: Invite de commande Windows lancée en tant qu'Administrateur Local

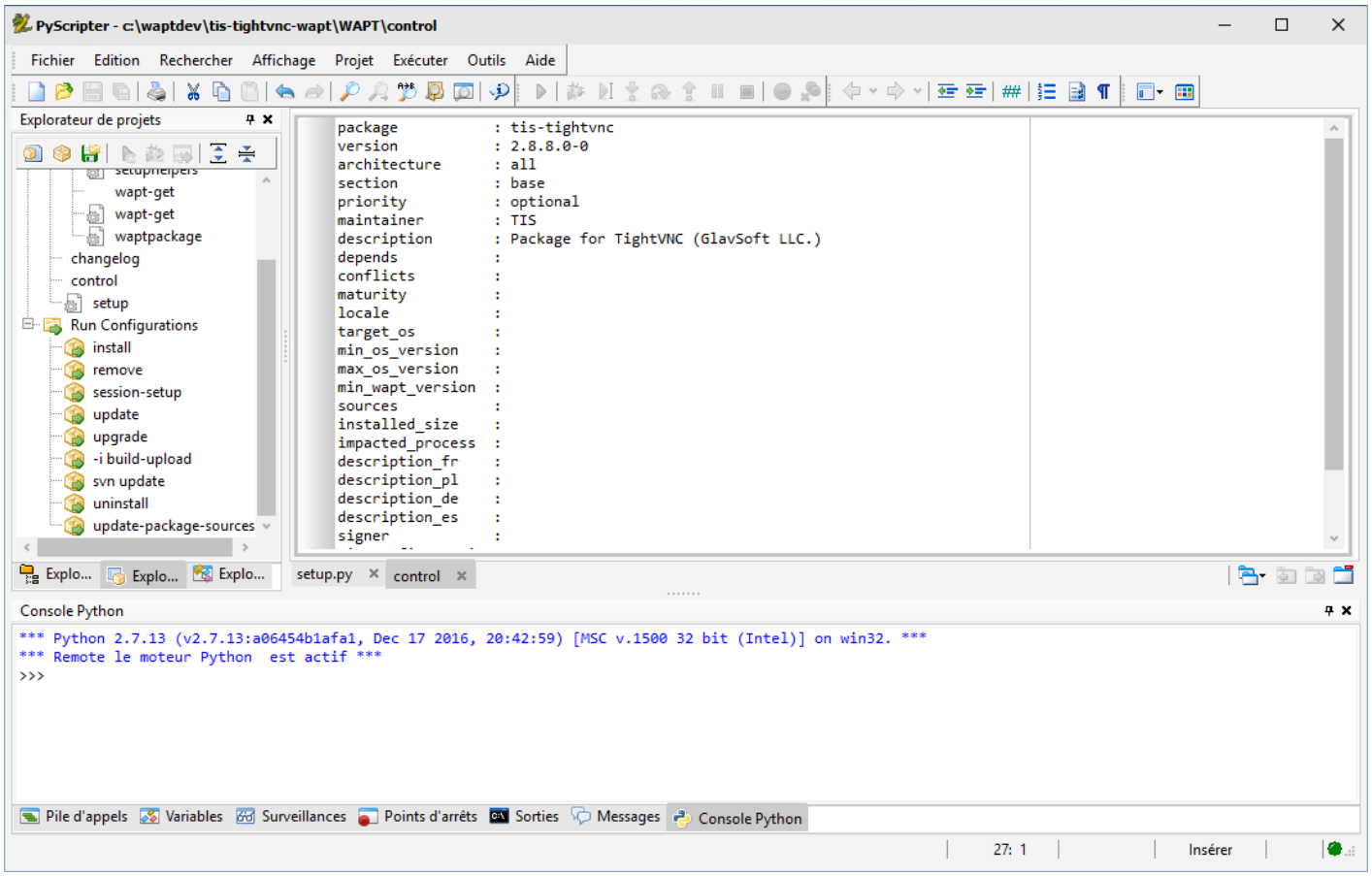


Figure159: PyScripter avec le projet TightVNC ouvert

- il crée le fichier `setup.py` générique ;
- il crée le fichier `control` qui identifie le paquet ;
- il crée le fichier `wapt.psproj` générique contenant des fonctions d'exécution WAPT pré-paramétrées ;

Indication: On pourra ensuite :

- contrôler et compléter les informations du fichier control ;
- ajouter des fichiers supplémentaires (version x86 du MSI, fichiers de configuration, etc) ;
- personnaliser la procédure d'installation dans `setup.py` ;
- modifier le fichier `control` si nécessaire, voir : *Le fichier control* ;

```
package      : tis-tightvnc
version      : 2.7.0-1
architecture : all
section      : base
priority     : optional
maintainer   : Tranquil-IT Systems
description  : package for TightVNC
depends      :
conflicts    :
sources      :
```

Note: À noter qu'une sous version (*-1* ; tiret + chiffre un) a été ajoutée. C'est la version du paquet WAPT.

Cela permet de ne pas modifier la version réelle du logiciel quand on pousse une nouvelle installation sur le parc.

- vérifier le fichier *setup.py* ;

Dans le fichier `setup.py` il faudra vérifier que la clé de désinstallation et les paramètres silencieux sont corrects pour le MSI.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ["{8B9896FC-B4F2-44CD-8B6E-78A0B1851B59}"]

def install():
    print('installing tis-tightvnc')
    run(r'"tightvnc-2.8.5-gpl-setup-64bit.msi" /q /norestart')
```

Note: Depuis la version 1.3.12 **`install_msi_if_needed`** est la fonction utilisée par défaut lors de la création d'un modèle de paquet pour un MSI.

> Dans cet exemple nous utilisons la fonction **`run`** pour montrer l'évolution du paquet. La fonction **`install_msi_if_needed`** est expliquée dans la *partie paquet msi evolué de la documentation*.

- tester l'installation du paquet sur votre machine de développement ;

L'intérêt de l'environment de développement intégré **PyScripter** est de pouvoir tester en local les paquets WAPT rapidement et de manière itérative.

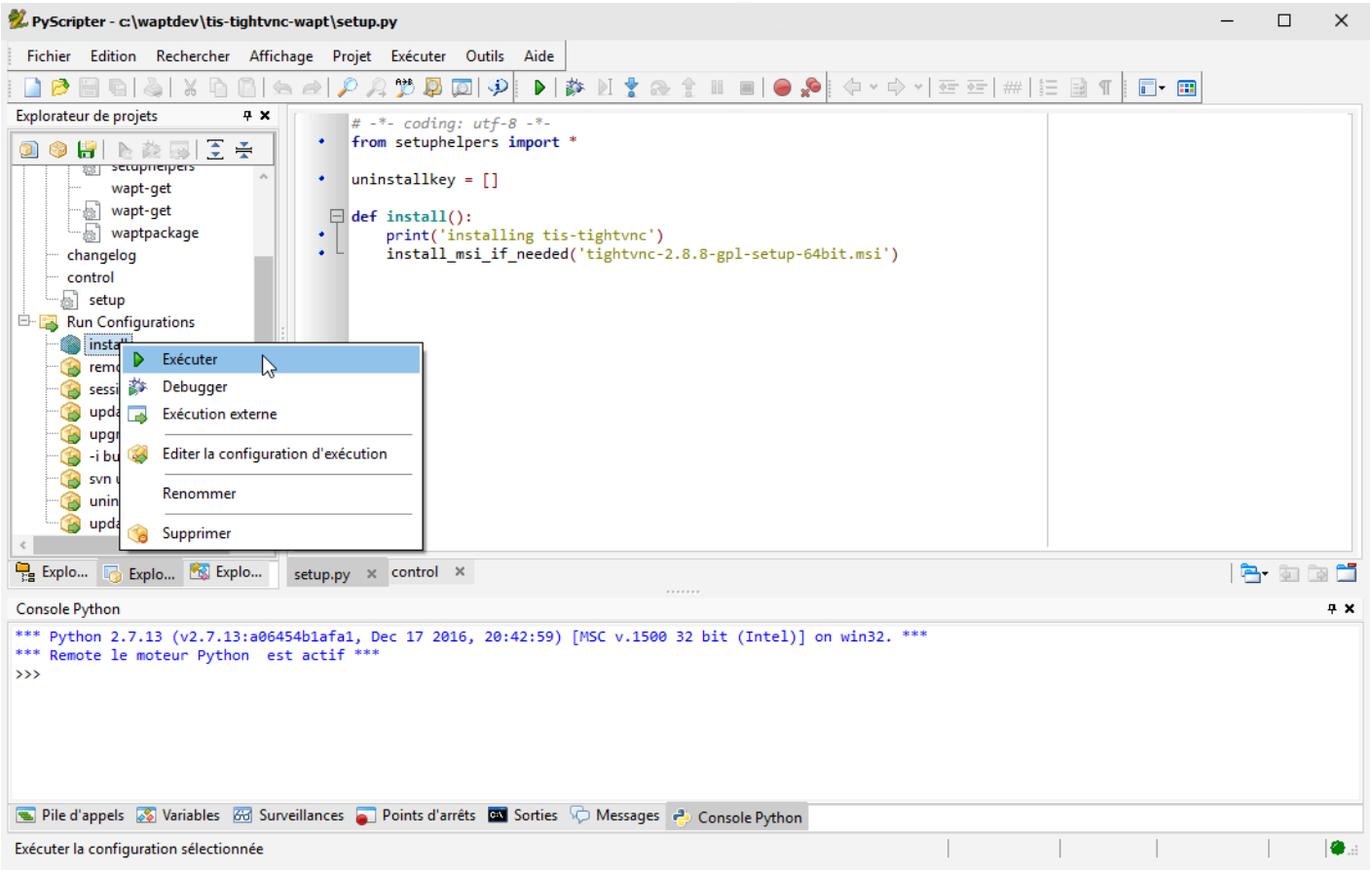


Figure160: Tester l'installation du paquet sur la machine de développement

Table28: Title

Paramètre	Valeur	Description
install	Exécuter	Lancer l'installation du logiciel avec les paramètres renseignés dans `setup.py`.
install	Débogguer	Lancer le debugger ligne par ligne.
remove	Exécuter	Lancer la désinstallation.
-i build-upload	Exécuter	Augmente le numéro de version du paquet WAPT, construit le paquet et le télécharge sur le dépôt WAPT.

Construire et charger le paquet WAPT

Une fois le projet de paquet créé, on peut lancer la construction du paquet sans modification depuis l'invite de commande Windows.

```
wapt-get build-upload -i c:\waptdev\tis-tightvnc-wapt
```

Note: Lorsqu'on exécute cette commande, WAPT :

- génère le fichier `manifest.sha256` avec la liste des fichiers contenus dans le paquet ;
- il zippe le répertoire `C:waptdevtis-tightvnc-wapt` avec un nom canonique pour les paquets WAPT ;
- ajoute la signature (nécessite la présence de la clé privée) ;
- charge le fichier WAPT en https vers le dépôt WAPT ;
- regénère le fichier d'index `https://srvwapt.mydomain.lan/wapt/Packages` pour prendre en compte le nouveau paquet ou la nouvelle version du paquet ;

Le paquet est prêt à être déployé.

Exemple :

```
wapt-get -i build-upload C:\waptdev\tis-tightvnc-wapt

Building  C:\waptdev\tis-tightvnc-wapt
Package tis-tightvnc (=2.8.5.0-1) content:
  setup.py
  tightvnc-2.8.5-gpl-setup-64bit.msi
  WAPT\control
  WAPT\wapt.psproj
...done. Package filename C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
Signing C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt

7-Zip [64] 16.04: Copyright (c) 1999-2016 Igor Pavlov: 2016-10-04

Open archive: C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
--
Path = C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt
Type = zip
Physical Size = 1756458

Updating archive: C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt

Items to compress: 0

Files read from disk: 0
Archive size: 1755509 bytes (1715 KiB)
Everything is Ok
Package C:\waptdev\tis-tightvnc_2.8.5.0-1_all.wapt signed: signature:
FVn2yx77TwUHaDauSPHxJZiPAyMQe4PqLF5n6wY9YPAwY4ijHe6NgDFrexXf8ZYbHAiNa5b8V/Qj
wTVHiqpbXnZotiVIGrJDhgbaLwZ9CK6pfWiflC4126nx6PMF3T1i6w0R0NOE2wJpOSRYESk7lDUz
9CPfzJCLcOXwh0F5eZc96wbkDkSbpn1f+x5tOlvyy/FW2m8RbZQhJcO21j9gGX7It0QNecaOxXgz
qkZZKBDNASOBYAF22M1+zHb59DWQ63Q8yMj5t5szEUTkGtQNG6vZz3gb9Yraq361BIGaBDYUM31j
ZgpaHvP0vdK3c1x1mhyhC7q6eZ/UCW5tETTCiA==
```

(suite sur la page suivante)

(suite de la page précédente)

```
Uploading files...
WAPT Server user :admin
WAPT Server password:
Status: OK, tis-tightvnc_2.8.5.0-1_all.wapt uploaded, 1 packages analysed
```

On peut aussi faire un **`build-upload`** directement depuis le panneau *Run Configurations* du projet dans **`PyScripter`** :

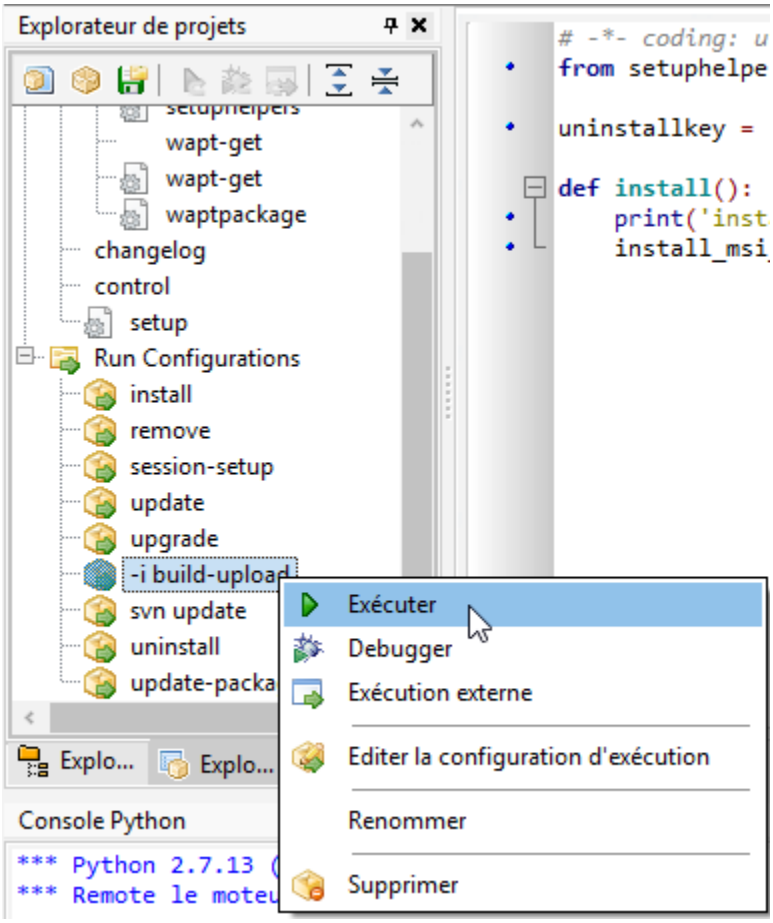


Figure161: L'option « -i build-upload » du projet PyScripter

6.14.5 Packager des paquets .msi évolués

Améliorer le paquet MSI

L'installation / la mise à jour du paquet TightVNC pris pour exemple précédemment nécessitera de le fermer le temps de la mise à jour.

Méthode manuelle : killalltasks

Une première méthode est de *tuer* tous les processus VNC avant de lancer l'installation du paquet *tis-tightvnc*.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ["{8B9896FC-B4F2-44CD-8B6E-78A0B1851B59}"]

def install():
    print('installing tis-tightvnc')
    killalltasks("vncviewer.exe")
    run(r'"tightvnc-2.8.5-gpl-setup-64bit.msi" /q /norestart')
```

Méthode élégante : install_msi_if_needed

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-tightvnc')
    install_msi_if_needed('tightvnc-2.8.5-setup-64bit.msi')
```

- la librairie de fonctions setuphelpers fournit la fonction **`install_msi_if_needed`** qui permet de gérer ce cas directement en une seule fonction ;
- la fonction testera également si une version du logiciel est déjà installée sur la machine avec la clé de désinstallation ;
- si présence il y a, l'installation sera enclenchée uniquement si la version actuellement installée est inférieure ;
- après installation, la fonction testera finalement la présence de la clé de déinstallation et sa version pour vérifier que tout s'est bien passé ;

Table29: Liste des arguments disponibles avec *install_exe_if_need*.

Paramètres	Valeur par défaut	Description
`msi`		nom du fichier MSI à exécuter.
`min_version`	None	version minimale au dessus de laquelle il mettra à jour.
`killbefore`	[]	liste des programmes à tuer avant de lancer l'installation.
`accept_returncodes`	[0,3010]	codes de retour autres que 0 ou 3010 acceptés en retour par la fonction.
`timeout`	300	durée d'attente maximale d'installation (en secondes).
`properties`	{}	propriétés supplémentaires à passer en argument au MSI pour l'installation.
`get_version`	None	valeur passée en paramètre pour le contrôle de version au lieu de celle retournée par la fonction *installed_softwares*
`remove_old_version`	False	supprimme automatiquement une ancienne version d'un logiciel dont la *uninstallkey* est identique
`force`	False	force l'installation du logiciel même si une *uninstall key* avec une version identique est trouvée.

Note: La fonction **install_msi_if_needed** récupère la clé de désinstallation depuis le MSI, il n'est pas nécessaire de l'écrire dans le fichier `setup.py`.

En lançant le paquet, on observe ce qu'il se passe dans la console quand le logiciel est déjà installé.

```
wapt-get -ldebug install C:\waptdev\tis-tightvnc-wapt
Installing WAPT file C:\waptdev\tis-tightvnc-wapt
installing tis-tightvnc
installing x64 version
MSI tightvnc-2.8.5-gpl-setup-64bit.msi already installed. Skipping msiexec

Results:

=== install packages ===
C:\waptdev\tis-tightvnc-wapt    | tis-tightvnc (2.8.5.0-1)
```

Gérer les plateformes x32/x64

Pour la gestion de l'architecture de processeur x32 / x64, on utilise la fonction **iswin64()**.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print(u'Installation en cours de TightVNC')
    if iswin64():
        print('installation version 64 bits')
        install_msi_if_needed('tightvnc-2.8.5-setup-64bit.msi')
    else:
        print('installation version 32 bits')
        install_msi_if_needed('tightvnc-2.8.5-setup-32bit.msi')
    print(u'Installation terminée.')
```

Ajouter des propriétés supplémentaires en argument

Pour ajouter des propriétés supplémentaires on va les stocker dans un élément *dict*.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

properties = {
    'SERVER_REGISTER_AS_SERVICE':0,
    'SERVER_ADD_FIREWALL_EXCEPTION':0,
    }

def install():
```

(suite sur la page suivante)

(suite de la page précédente)

```
print(u'Installation en cours de TightVNC')
if iswin64():
  print('installation version 64 bits')
  install_msi_if_needed('tightvnc-2.8.5-setup-64bit.msi', properties =
                                                          properties)
else:
  print('installation version 32 bits')
  install_msi_if_needed('tightvnc-2.8.5-setup-32bit.msi', properties =
                                                          properties)
print(u'Installation terminée.')
```

6.14.6 Packager des paquets .exe simples

Note: Variation par rapport aux MSI

WAPT préfère les MSI car les installeurs `.exe` ne sont pas standardisés et leurs arguments silencieux peuvent être différents d'un logiciel à un autre.

Indication: Depuis la version 1.3.12 de WAPT, une nouvelle méthode de création rapide de paquets depuis la console WAPT est devenue disponible.

Cette documentation est toujours d'actualité, nous vous conseillons cependant d'utiliser l'interface graphique pour instancier vos modèles de paquets. voir : *Créer un paquet WAPT depuis la console*.

- obtenir l'installeur exe depuis une source sûre ;

  Télécharger l'installeur au format *.exe (par exemple Firefox ESR x64) :

- rechercher la documentation associée pour les flags silencieux ;
  - Sur le site de la Fondation Mozilla ;
  - autres méthodes pour récupérer le flag silencieux :
    - dépôt de paquets WPKG ;
    - dépôt de paquets Chocolatey ;
    - recherche Internet avec le terme « Firefox silent install » ;

Créer un template de base de paquet à partir du fichier .exe

- lancer une invite de commande (cmd.exe) en mode *Administrateur Local* ;
- créer un projet de paquet logiciel ;

```
wapt-get make-template <chemin_exe> <nom_du_paquet>
```

Exemple avec Mozilla Firefox ESR :

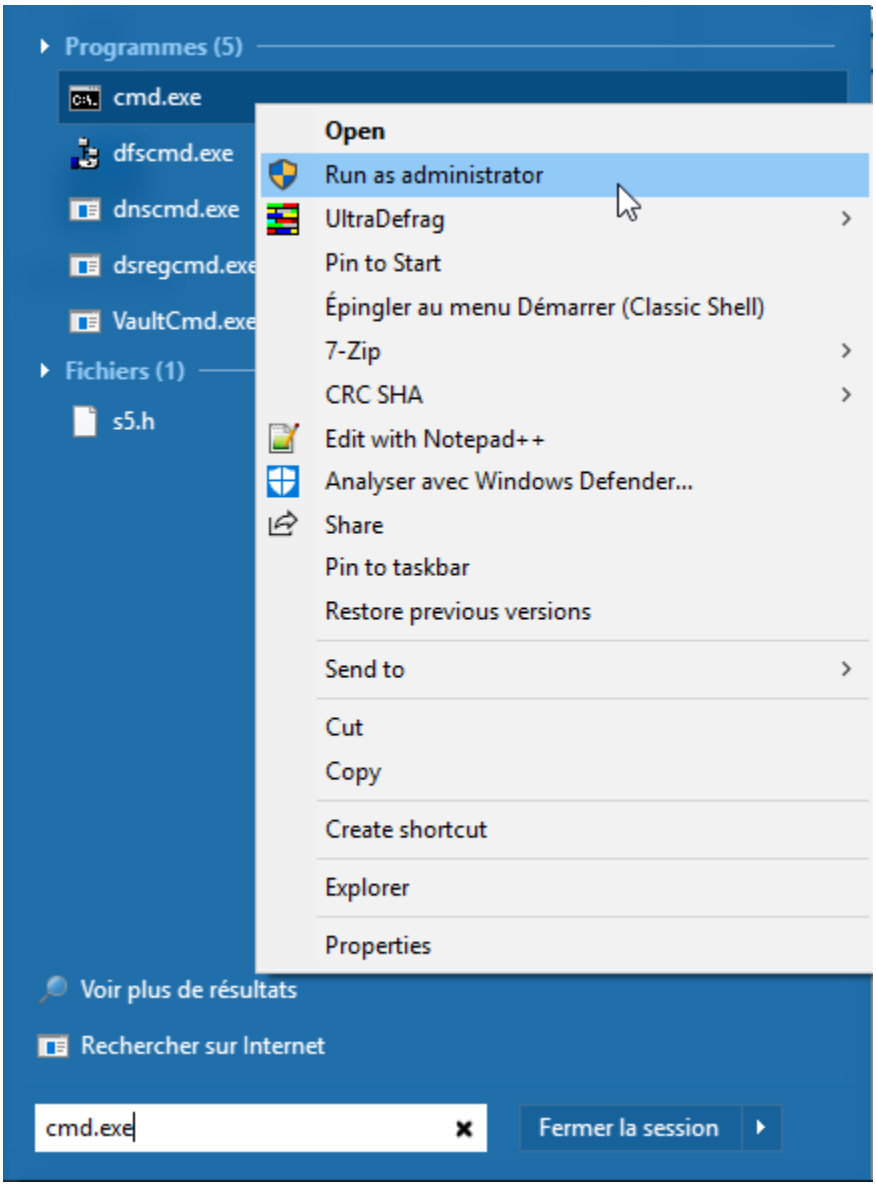


Figure162: Lancement de l'invite de commande Windows en tant qu'Administrateur Local

```
wapt-get make-template "Firefox Setup 52.6.0esr.exe" "tis-firefox-esr"

Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package C:\waptdev\tis-firefox-esr-wapt
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload C:\waptdev\tis-firefox-esr-wapt
```

PyScripter s'ouvre ensuite avec le projet du paquet *.exe*.

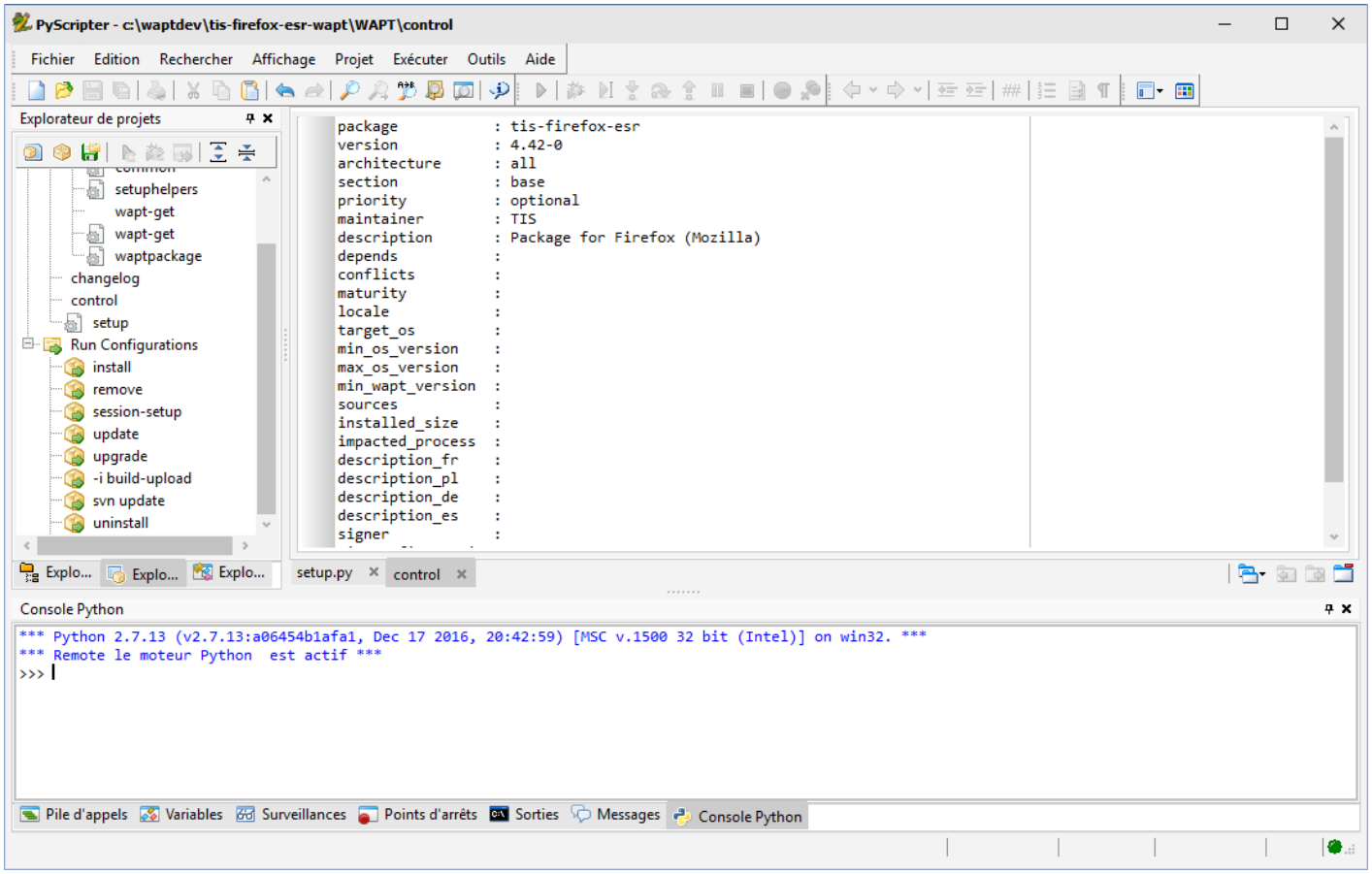


Figure163: PyScripter affichant le fichier *control*

- vérifier le fichier `control` ;

  Mozilla Firefox-ESR ne répond pas aux standards et retourne un numéro de version erroné (il s'agit du numéro de version du logiciel qui créé l'installeur).

  Fichier *control* d'origine

```
package             : tis-firefox-esr
version             : 4.42.0.0-0
architecture        : all
section             : base
priority            : optional
```

(suite sur la page suivante)

(suite de la page précédente)

```
maintainer          : user
description         : automatic package for firefox setup 52.6.0esr
```

Fichier *control* modifié

```
package             : tis-firefox-esr
version             : 52.6.0-1
architecture        : all
section             : base
priority            : optional
maintainer          : Tranquil-IT Systems
description         : Mozilla Firefox 52.6.0 ESR
```

Note: Une sous-version *-1* a été ajoutée. C'est la version de packaging du paquet WAPT.

Cela permet de ne pas modifier la version réelle du logiciel quand on pousse par exemple une nouvelle configuration du même logiciel sur le parc.

- vérifier le fichier `setup.py` ;

  WAPT a mis un flag silencieux générique */VERYSILENT* qui ne fonctionne peut-être pas avec Mozilla Firefox ESR.

  Dans ce cas il convient d'indiquer le bon flag silencieux que nous avons recherché précédemment.

- changer le code du fichier `setup.py` en conséquence ;

```
:emphasize-lines: 8
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-firefox-esr')
    run(r'"Firefox Setup 52.6.0esr.exe" -ms')
```

- enregistrer le paquet ;

Gérer la désinstallation

Avec un exe, la clé de désinstallation n'est pas disponible sans avoir installé le logiciel une première fois.

Cette clé est disponible dans la base de registre :

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
```

or on 64bits systems

```
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall
```

- ouvrir une invite de commande Windows ;
- récupérer la clé de désinstallation avec `wapt-get list-registry firefox`

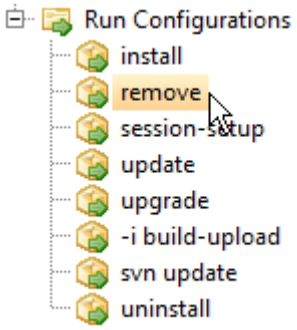
```
UninstallKey                          Software                                Version                 ␣
↪Uninstallstring
------------------------------------ --------------------------------------- ------------------ -
↪--------------------------------------------------
Mozilla Firefox 52.6.0 ESR (x64 fr)   Mozilla Firefox 52.6.0 ESR (x64 fr)    52.6.0
↪"C:\Program Files\Mozilla Firefox\uninstall\helper.exe"
```

- copier la clé de désinstallation **UninstallKey** : *Mozilla Firefox 52.6.0 ESR (x64 fr)* ;
- modifier en conséquence le script **`setup.py`** avec la clé de désintallation correcte ;

```
:emphasize-lines: 4

# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ['Mozilla Firefox 52.6.0 ESR (x64 fr)']

def install():
    print('installing tis-firefox-esr')
    run(r'"Firefox Setup 52.6.0esr.exe" -ms')
```

Note: La clé de désinstallation doit être exactement la même que celle listée avec la commande *list-registry*. La clé de désinstallation peut être un GUID tel que *9516000000-0052-040C-0000-0000000FF1CE*, un GUID avec des caractères entre crochets, *{95160000-0052-040C-0000-0000000FF1CE}*, ou simplement une chaîne de caractères comme *Git_is1* ou *Mozilla Firefox 52.6.0 ESR (x64fr)*.

- relancer l'installation du paquet afin que la clé de désinstallation soit prise en compte dans la base de données locale WAPT ;
- tester la désinstallation du paquet ;
- lancer un **`remove`** du paquet depuis le panneau *Run Configurations* de **`PyScripter`** ;



À l'issue de la désinstallation, le programme est désinstallé

On pourra vérifier l'absence de clé de registre de désinstallation en relançant la commande **`wapt-get list-registry`**.

```
UninstallKey          Software          Version           Uninstallstring
--------------------- ----------------- ----------------- ---------------
--------------------- ----------------- ----------------- ---------------
```

Cas particulier d'un désinstalleur non-silencieux

Il arrive dans certains cas que le programme s'installe silencieusement mais que le désinstalleur ne s'exécute pas silencieusement.

Dans ce cas précis on va surcharger la fonction **uninstall()**.

Exemple avec Mozilla Firefox si cela était nécessaire :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ['Mozilla Firefox 52.6.0 ESR (x64 fr)']

def install():
    print('installing tis-firefox-esr')
    run(r'"Firefox Setup 52.6.0esr.exe" -ms')

def uninstall():
    print('uninstalling tis-firefox-esr')
    run(r'"C:\Program Files\Mozilla Firefox\uninstall\helper.exe" -ms')
```

Indication: Dans la fonction **uninstall()**, on ne peut pas appeler des fichiers contenus dans le paquet WAPT. Pour les appeler, il faudra avoir copié les fichiers dans un répertoire local de la machine lors de l'installation du paquet.

Construire et charger le paquet WAPT

Une fois l'installation et la désinstallation configurées et testées vous pouvez effectuer un **build-upload** pour charger votre nouveau paquet WAPT sur votre dépôt.

6.14.7 Créer un paquet .exe evolué

Améliorer l'installation

Notre paquet Mozilla Firefox ESR est fonctionnel mais il n'est pas encore optimal.

- le paquet exécute l'installation du logiciel même si le logiciel est déjà installé, il l'écrase dans tous les cas ;
- le paquet ne vérifie pas la version du logiciel s'il est déjà installé ;
- le paquet ne vérifie pas la version du logiciel s'il est déjà installé ;

Comme avec les MSI, nous résolvons ces problèmes en utilisant la fonction `install_exe_if_needed`.

Utiliser *install_exe_if_needed*

La fonction est sensiblement la même que celle utilisée pour les installeurs MSI, avec quelques différences :

- la fonction nécessite l'ajout des flags silencieux en paramètre ;
- la fonction nécessite l'ajout de la clé de désinstallation en paramètre ;

On modifie notre exemple Mozilla Firefox ESR en conséquence :

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-firefox-esr')
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",
                          silentflags="-ms",
                          key='Mozilla Firefox 45.4.0 ESR (x64 fr)'
                          min_version="45.5.0"
                          killbefore="firefox.exe")
```

Table30: Liste des arguments disponibles avec *install_exe_if_need*.

Paramètres	Valeur par défaut	Description
`exe`		nom du fichier *.exe* à exécuter.
`silentflags`		paramètres silencieux à passer en argument à l'installeur.
`key`	None	clé de désinstallation du programme.
`min_version`	None	version minimale au dessus de laquelle il mettra à jour.
`killbefore`	[ ]	liste des programmes à tuer avant de lancer l'installation.
`accept_returncodes`	[0,3010]	codes de retour autres que 0 ou 3010 acceptés en retour par la fonction.
`timeout`	300	durée d'attente maximale d'installation (en secondes).
`get_version`	None	valeur passée en paramètre pour le contrôle de version au lieu de celle retournée par la fonction **`installed_softwares`**.
`remove_old_version`	False	Permet d'enclencher automatiquement la suppression d'une uninstallkey identique avec une version précédente à celle mentionnée
`force`	False	Permet de forcer l'installation même si une uninstallkey avec une version identique est trouvée.

En conséquence, le paquet aura un comportement différent :

- le logiciel Firefox s'installera uniquement si le logiciel n'est pas installé et si la version est strictement inférieure à 45.5.0, sauf si l'option `--force` est indiquée lors de l'installation du paquet ;
- à l'installation, les processus **`firefox.exe`** en cours d'exécution seront terminés ;
- la fonction ajoutera elle-même la clé de désinstallation, donc laisser *uninstallkey* vide ;
- à la fin de l'installation, la fonction ira vérifier si l'*uninstallkey* est bien présente sur le poste et si la version est bien égale ou supérieure à 45.5.0, si ce n'est pas le cas, elle basculera le paquet en **ERROR** ;

Gérer la désinstallation

Cas particulier d'un dé-installeur non-silencieux

Dans ce cas particulier, un paquet utilisant **`install_exe_if_needed`** renseigne la clé de désinstallation, mais cette clé de désinstallation pointe vers un dé-installeur non-silencieux.

Il nous faut contourner le problème en utilisant une fonction qui va supprimer la clé de désinstallation à la fin de l'installation.

```
:emphasize-lines: 13

# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-firefox-esr')
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",
                          silentflags="-ms",
                          key='Mozilla Firefox 45.4.0 ESR (x64 fr)',
                          min_version="45.5.0",
                          killbefore="firefox.exe")
    uninstallkey.remove('Mozilla Firefox 45.4.0 ESR (x64 fr)')

def uninstall():
    print('uninstalling tis-firefox-esr')
    run(r'"C:\Program Files\Mozilla Firefox\uninstall\helper.exe" -ms')
```

6.14.8 Fonctionnalités supplémentaires

Personnaliser le contexte utilisateur

Un des principaux interêts de WAPT est l'installation et la désinstallation silencieuse de logiciels, et ceci **même quand un utilisateur restreint est connecté**.

L'agent WAPT fonctionne en compte système et il pourra personnaliser des paramètres en contexte utilisateur de manière dynamique.

Les personnalisations de l'environment utilisateur s'appuient sur la fonction `session_setup` définie dans la librairie WAPT *Setuphelpers*.

Principe de fonctionnement

La personnalisation en contexte utilisateur fonctionne comme suit :

- on décrit les instructions dans la fonction **`session_setup()`** du fichier `setup.py` du paquet WAPT ;
- quand le paquet est déployé, les instructions sont stockées dans la base de données locale de l'agent WAPT ;
- quand l'utilisateur se connecte à la machine, les instructions sont exécutées dans la limite d'une exécution par utilisateur et par paquet WAPT ;

Indication: La personnalisation du `session_setup` n'aura lieu qu'une fois par utilisateur et par version de paquet ; un raccourci créé dans ce contexte ne l'est qu'une fois et non à chaque démarrage. Pour exécuter une tâche à chaque démarrage, il est préférable de définir une tâche Windows planifiée qui sera lancée par une *GPO* locale ou par un script de démarrage.

Exemple : créer un raccourci personnalisé

One of the possibilities offered by *Setuphelpers* is adding personalized shortcuts on user desktops, instead of a desktop shortcut common to all users.

Nous utiliserons pour ça la fonction `create_user_desktop_shortcut()` pour créer un raccourci contenant le nom de l'utilisateur et qui passera en argument à Firefox le site https://tranquil.it par exemple.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-firefox-esr')
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",
                          silentflags="-ms",
                          key='Mozilla Firefox 45.4.0 ESR (x64 fr)'
                          min_version="45.5.0"
                          killbefore="firefox.exe")

def session_setup():
  create_user_desktop_shortcut("Mozilla Firefox de %s" % get_current_user(),
                               r'C:\Program Files\Mozilla Firefox\firefox.exe',
                               arguments="-url https://tranquil.it")
```

Exemple: déployer un logiciel portable avec WAPT

Un bon exemple de paquet applicatif WAPT est celui d'un logiciel dit *portable*. Pour cela, il faudra :

- créer le répertoire d'installation dans `C:\Program Files (x86)` ;
- copier l'application dans le dossier ;
- créer un raccourci sur le bureau de l'utilisateur ;
- gérer la désinstallation de l'application portable ;
- fermer l'application si elle est en cours d'exécution ;

Exemple avec ADWCleaner

- créer un paquet groupe et modifier le fichier `control` pour en faire un paquet applicatif ;

```
wapt-get make-group-template tis-adwcleaner
```

```
package          : tis-adwcleaner
version          : 6.041-1
architecture     : all
section          : base
priority         : standard
maintainer       : Tranquil-IT Systems
description      : ADW Cleaner
```

Le fichier `C:\waptdev\tis-adwcleaner-wapt` est créé.

- télécharger puis copier / coller l'exécutable **`adwcleaner.exe`** dans le répertoire `C:waptdevtis-adwcleaner-wapt` ;
- ouvrir et modifier ensuite le fichier d'installation `C:waptdevtis-adwcleaner-waptsetup.py` ;

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

targetdir = makepath(programfiles32,'adwcleaner')
exename = 'adwcleaner_6.041.exe'

def install():
  mkdirs(targetdir)
  filecopyto(exename,targetdir)
  create_programs_menu_shortcut('ADWCleaner',target=makepath(targetdir,exename))
  # control est un objet PackageEntry correspondant au paquet en cours d'installation
  register_windows_uninstall(control)

def uninstall():
  killalltasks(exename)
  remove_programs_menu_shortcut('ADWCleaner')
  if isdir(targetdir):
      remove_tree(targetdir)
  unregister_uninstall('tis-adwcleaner')
```

6.14.9 Personnaliser l'environment utilisateur

Principe du *session_setup*

Il est parfois nécessaire de personnaliser un programme ou un logiciel en contexte utilisateur pour rendre le logiciel immédiatement exploitable par l'utilisateur dans le contexte spécifique de son entreprise ou du service au sein de son entreprise :

- créer des raccourcis sur le bureau utilisateur avec des arguments spécifiques ;
- modifier de clés registres utilisateurs ;
- modifier des fichiers, une configuration de navigateur ;

- configurer des raccourcis réseaux aux modèles de documents de l'entreprise pour assurer la conformité des documents aux chartes éditoriales en vigueur ;
- paramétrer la messagerie de l'utilisateur à partir d'un référentiel externe (annuaire, base de données, etc) ;
- paramétrer un logiciel bureautique ou métier à partir d'un référentiel externe (annuaire, base de données, etc) ;

La fonction **session_setup** bénéficie de toute la puissance et de l'étendue des librairies python pour atteindre un niveau d'automatisation élevé.

Fonctionnement du *session_setup*

La fonction WAPT **session_setup** s'exécute à chaque ouverture de session utilisateur, en appelant la commande :

```
C:\Program Files (x86)\wapt\wapt-get.exe session-setup ALL
```

L'appel à cette fonction permet d'exécuter la partie **session_setup** de chaque paquet WAPT logiciel installé sur la machine.

C:\Program Files (x86)\wapt\

Attention: Le **session_setup** de chaque paquet n'est exécuté qu'**une seule fois par paquet ou version de paquet et par profil utilisateur**.

Le **session_setup** va enregistrer dans le fichier `%appdata%\waptwaptsession.sqlite` les **session_setup** déjà exécutées.

Pour lancer un programme à chaque démarrage, il s'agira de privilégier l'utilisation d'un script au démarrage utilisateur contenu dans le **démarrage utilisateur** (*startup(0)*) ou **Démarrage All Users** (*startup(1)*), ou via une stratégie de groupe GPO.

Exemple de sortie de la commande `wapt-get session-setup ALL` :

Note: le `session_setup` de l'utilisateur connecté, avait déjà été exécuté.

```
wapt-get session-setup ALL

Configuring tis-7zip ... No session-setup. Done
Configuring tis-ccleaner ... Already installed. Done
Configuring tis-vlc ... No session-setup. Done
Configuring tis-tightvnc ... No session-setup. Done
Configuring tis-paint.net ... No session-setup. Done
Configuring wsuser01.mydomain.lan ... No session-setup. Done
```

Utiliser le *session_setup*

On définit la partie **session_setup** dans le fichier `setup.py` du paquet dans une fonction *def session_setup()* :

Exemple :

```
def session_setup():
   registry_setstring(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows Live\\Common",'TOUVersion
↪','16.0.0.0', type=REG_SZ)
```

Attention: Avec **session_setup**, il n'est pas possible de faire appel à des fichiers contenus dans le paquet.

Pour utiliser des fichiers lors de la désinstallation, copier / coller les fichiers dans un répertoire extérieur au paquet lors de l'installation du paquet (exemple : un sous-répertoire créé à la racine du répertoire utilisateur).

6.14.10 Utiliser les fonctions d'Audit

Note: Cette fonctionnalité est disponible dans la version **Entreprise**.

Pour quoi faire ?

L'audit permet d'effectuer des vérifications régulières sur les configurations des postes et de centraliser le résultat des vérifications dans la console WAPT. Ceci permet de vérifier que votre parc est conforme à votre référentiel sur la durée.

Vous pouvez par exemple :

- vérifier régulièrement la liste des administrateurs locaux des postes ;
- vérifier régulièrement la bonne configuration d'un logiciel ;
- vérifier régulièrement la présence de la bonne version d'un logiciel ;
- vérifier régulièrement les configurations de sécurité d'un poste ;

La fonction **audit** bénéficie de toute la puissance et de l'étendue des librairies python pour atteindre une précision d'audit élevée.

Fonctionnement de l'Audit

Les tâches d'**audit** s'exécutent après un **upgrade** puis à intervalle régulier défini par la valeur de `audit_schedule`.

Pour executer manuellement un audit vous pouvez également exécuter la commande :

```
C:\Program Files (x86)\wapt\wapt-get.exe audit
```

Note: Par défaut l'audit ne sera pas exécuté si l'audit n'est pas nécessaire.

Pour forcer l'exécution vous pouvez exécuter la commande :

```
C:\Program Files (x86)\wapt\wapt-get.exe audit -f
```

L'appel à cette fonction permet d'exécuter la partie **audit** de chaque paquet WAPT logiciel installé sur la machine.

C:\Program Files (x86)\wapt\

Exemple de sortie de la commande `wapt-get audit` :

```
Auditing tis-disable-ipv6 ...
Skipping audit of tis-disable-ipv6(=1.0-6), returning last audit from 2018-09-25T11:20:58.426000
tis-disable-ipv6 -> OK
Auditing tis-disable-js-adobe ...
Skipping audit of tis-disable-js-adobe(=13), returning last audit from 2018-09-25T11:20:58.502000
tis-disable-js-adobe -> OK
Auditing tis-disable-js-chrome ...
Skipping audit of tis-disable-js-chrome(=3), returning last audit from 2018-09-25T11:20:58.566000
tis-disable-js-chrome -> OK
Auditing tis-disable-office-dde ...
Skipping audit of tis-disable-office-dde(=1.0-2), returning last audit from 2018-09-25T11:20:58.
↪615000
tis-disable-office-dde -> OK
Auditing tis-sysmon ...
Skipping audit of tis-sysmon(=8.0-12), returning last audit from 2018-09-25T11:20:58.722000
tis-sysmon -> OK
Auditing tis-java ...
OK: Uninstall Key {26A24AE4-039D-4CA4-87B4-2F32180181F0} in Windows Registry.
OK: Uninstall Key {26A24AE4-039D-4CA4-87B4-2F64180181F0} in Windows Registry.
tis-java -> OK
```

Note: Dans l'exemple ci-dessus, l'audit avait déjà été exécuté pour *tis-disable-js-chrome*, *tis-disable-ipv6* . . . mais pas pour *tis-java*.

Comment utiliser la fonction d'audit ?

On définit la partie **audit** dans le fichier `setup.py` du paquet dans une fonction **def audit()** :

Exemple :

```
def audit():
    if not registry_readstring(HKEY_LOCAL_MACHINE,makepath('SYSTEM','CurrentControlSet','Services
↪','USBSTOR'),'Start'):
        print(r"La key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USBSTOR\Start n
↪'existe pas")
        return "ERROR"
    valuestart = registry_readstring(HKEY_LOCAL_MACHINE,makepath('SYSTEM','CurrentControlSet',
↪'Services','USBSTOR'),'Start')
    if int(valuestart) != 4:
        print("La valeur de Start n'est pas 4 , Start=%s " % valuestart )
        return "WARNING"
    print(ur"La valeur de Start est bien est bien égal a 4")
    return "OK"
```

Indication: Cet exemple permet de vérifier que les clés usb sont bien interdites sur le poste.

L'audit restitue une des 3 valeurs suivantes :

- **OK** ;
- **WARNING** ;
- **ERROR** ;

Attention: Avec **audit**, il n'est pas possible de faire appel à des fichiers contenus dans le paquet.

Pour utiliser des fichiers lors de l'audit il faut d'abord les copier dans un répertoire temporaire de la machine lors de l'installation du paquet.

Planifier un audit

Les tâches d'"**audit** s'exécutent après un **upgrade** puis à intervalle régulier défini par la valeur de `audit_schedule`.

La valeur est contenue dans le fichier `control` du paquet WAPT.

Par défault si **audit_schedule** est vide alors il faudra effectuer l'audit manuellement ou à partir de la console.

Sinon la valeur peut être indiquée de plusieurs manières :

- un entier (en minutes) ;
- un entier suivi d'une lettre (m = minutes , h = heure , d = jour , w = semaine) ;

Audit par défaut

Par défault, si aucun audit n'est déclaré, l'agent WAPT vérifiera la présence des uninstallkey du paquet wapt.

De cette manière WAPT vérifie que le logiciel est toujours présent.

6.14.11 Exemples simples de fonctions couramment utilisées

Nous présentons ici quelques fonctions implémentées dans *Setuphelpers* et fréquemment utilisées pour développer des paquets WAPT.

Tests et manipulation de dossiers et fichiers

Créer un chemin avec récursion

La commande **makepath** ...

```
makepath(programfiles,'Mozilla','Firefox')
```

... fabrique la variable pour le chemin `C:Program Files (x86)MozillaFirefox`.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=makepath#setuphelpers.makepath

Créer et détruire des répertoires

La commande **mkdirs** ...

```
mkdirs('C:\\test')
```

... crée le répertoire `C:\test`.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=mkdirs#setuphelpers.mkdirs

---

La commande **remove_tree** ...

```
remove_tree(r'C:\tmp\target')
```

... détruit le répertoire `C:\tmp\target`.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_tree#setuphelpers.remove_tree

---

Tester si des éléments de l'arborescence sont des fichiers ou des répertoires

La commande **isdir** ...

```
isdir(makepath(programfiles32,'software')):
    print('The directory exists')
```

... vérifie que `C:\Program Files (x86)\software` est un répertoire.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=isdir#setuphelpers.isdir

---

La commande **isfile** ...

```
isfile(makepath(programfiles32,'software','file')):
    print('file exist')
```

... vérifie que `C:\Program Files (x86)\software\file` est un fichier.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=isfile#setuphelpers.isfile

---

Tester si un répertoire est vide

La commande **dir_is_empty** ...

```
dir_is_empty(makepath(programfiles32,'software')):
    print('dir is empty')
```

... vérifie que le répertoire `C:\Program Files (x86)\software` est vide.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=dir_is_empty#setuphelpers.dir_is_empty

---

Copier un fichier

La commande **filecopyto** ...

```
filecopyto('file.txt',makepath(programfiles32,'software'))
```

... copie le fichier `file.txt` dans le répertoire `C:\Program Files (x86)\software`.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=filecopyto#setuphelpers.filecopyto

---

Copier un répertoire

La commande **copytree2** ...

```
copytree2('sources','C:\\projet')
```

... copie le dossier `sources` dans le répertoire `C:\projet`.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=copytree2#setuphelpers.copytree2

Récupérer la version d'un fichier

La commande **get_file_properties** ...

```
get_file_properties(makepath(programfiles32,'InfraRecorder','infrarecorder.exe'))['ProductVersion
↪']
```

... affiche les propriétés du paquet.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_file_properties#setuphelpers.get_file_properties

---

Manipulation de clés de registre

Tester la présence d'une clé de registre

La commande **registry_readstring** ...

```
if registry_readstring(HKEY_LOCAL_MACHINE, "SOFTWARE\\Google\\Update\\Clients\\{8A69D345-D564-
↪463c-AFF1-A69D9E530F96}", 'pv'):
    print('key exist')
```

... vérifie que la clé *{8A69D345-D564-463c-AFF1-A69D9E530F96}* existe dans le répertoire `SOFTWARE\Google\Update\Clients` de la ruche *HKEY_LOCAL_MACHINE*.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_readstring#setuphelpers.registry_readstring

---

Afficher la valeur d'une clé de registre

La commande **registry_readstring** ...

```
print(registry_readstring(HKEY_LOCAL_MACHINE, r'SOFTWARE\Google\Update\Clients\{8A69D345-D564-
↪463c-AFF1-A69D9E530F96}', 'pv'))
```

... lit la valeur de la clé *{8A69D345-D564-463c-AFF1-A69D9E530F96}* inscrite dans le répertoire `SOFTWARE\Google\Update\Clients` de la ruche *HKEY_LOCAL_MACHINE*.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_readstring#setuphelpers.registry_readstring

---

Modifier la valeur d'une clé de registre

La commande **registry_setstring** ...

```
registry_setstring(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows Live\\Common",'TOUVersion',
↪'16.0.0.0', type=REG_SZ)
```

... modifie la valeur de la clé *TOUVersion* dans le répertoire `SOFTWARE\Microsoft\Windows Live` de la ruche *HKEY_CURRENT_USER*.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_setstring#setuphelpers.registry_setstring

---

Créer et supprimer des raccourcis

create_desktop_shortcut / remove_desktop_shortcut

La commande **create_desktop_shortcut** ...

```
create_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

C:\Program Files (x86)\wapt\

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=create_desktop_shortcut#setuphelpers.create_desktop_shortcut

---

La commande **remove_desktop_shortcut** ...

```
remove_desktop_shortcut('WAPT Console Management')
```

... supprime le raccourci *WAPT Console Management* du répertoire `C:\Users\Public` ; le raccourci est supprimé pour tous les utilisateurs.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_desktop_shortcut#setuphelpers.remove_desktop_shortcut

---

create_desktop_shortcut / remove_desktop_shortcut

Indication: Ces fonctions sont utilisées avec le session_setup.

La commande **create_user_desktop_shortcut** ...

```
create_user_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

C:\Program Files (x86)\wapt\

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=create_user_desktop_shortcut#setuphelpers.create_user_desktop_shortcut

Remove a shortcut from the current users

La commande **remove_user_desktop_shortcut** ...

```
remove_user_desktop_shortcut('WAPT Console Management')
```

... supprime le raccourci *WAPT Console Management* du bureau de l'utilisateur.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_user_desktop_shortcut#setuphelpers.remove_user_desktop_shortcut

Environment Windows / Logiciels / Services

windows_version

La commande **windows_version** ...

```
windows_version()<Version('6.2.0'):
```

... vérifie que la version de Windows est strictement inférieure à *6.2.0*.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=windows_version#setuphelpers.windows_version

Liste des numéros de version de Windows.

iswin64

La commande **iswin64** ...

```
if iswin64():
    print('Pc x64')
else:
    print('Pc not x64')
```

... vérifie que le processeur de la machine est 64bits.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=iswin64#setuphelpers.iswin64

---

programfiles / programfiles32 / programfiles64

Renvoie les différentes localisations de *Program Files*

La commande **programfiles64** ...

```
print(programfiles64())
```

Renvoie le répertoire natif du programme, c.à.d. `C:Program Files (x86)` pour les architecture win64 et win32.

```
print(programfiles())
```

Renvoie le chemin vers le répertoire `Programs Files (x86)` (sur architecture win64) ou `Programs Files` (sur architecture win32).

```
print(programfiles32())
```

user_appdata / user_local_appdata

Indication: Ces fonctions sont utilisées avec le **session_setup**.

La commande **user_appdata** ...

```
print(user_appdata())
```

... renvoie le profil *appdata* itinérant de l'utilisateur courant (`C:\Users\%username%\AppData\Roaming`).

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=user_appdata#setuphelpers.user_appdata

---

La commande **user_local_appdata** ...

```
print(user_local_appdata())
```

... renvoie le profil *appdata* local de l'utilisateur courant (`C:\Users\%username%\AppData\Local`).

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=user_local_appdata#setuphelpers.user_local_appdata

---

disable_file_system_redirection

La commande **disable_file_system_redirection** ...

```
with disable_file_system_redirection():
    filecopyto('file.txt',system32())
```

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=disable_file_system_redirection#setuphelpers.disable_file_system_redirection

---

Gère le contexte pour désactiver temporairement le redirecteur wow3264.

get_computername / get_current_user

La commande **get_current_user** ...

```
print(get_current_user())
```

... affiche l'identifiant de l'utilisateur connecté.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_current_user#setuphelpers.get_current_user

---

La commande **get_computername** ...

```
print(get_computername())
```

... affiche le nom de la machine sans le domaine.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_computername#setuphelpers.get_computername

La commande **get_domain_fromregistry** ...

```
get_domain_fromregistry()
```

... renvoie le nom de la machine avec le domaine.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_domain_fromregistry#setuphelpers.get_domain_fromregistry

installed_softwares / uninstall_cmd

installed_softwares

La commande **installed_softwares** ...

```
installed_softwares('winscp')
```

... renvoie la liste des logiciels inscrits dans la base de registre machine sous forme de tableau.

```
[{'install_location': u'C:\\Program Files\\WinSCP\\', 'version': u'5.9.2', 'name': u'WinSCP 5.9.2
↪', 'key': u'winscp3_is1', 'uninstall_string': u'"C:\\Program Files\\WinSCP\\unins000.exe"',
↪'publisher': u'Martin Prikryl', 'install_date': u'20161102', 'system_component': 0}]
```

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=installed_softwares#setuphelpers.installed_softwares

uninstall_cmd

La commande **uninstall_cmd** ...

```
uninstall_cmd('winscp3_is1')
```

... renvoie la commande de désinstallation silencieuse :

```
"C:\Program Files\WinSCP\unins000.exe" /SILENT
```

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=uninstall_cmd#setuphelpers.uninstall_cmd

désinstaller des logiciels

```
for soft in installed_softwares('winscp3'):
    if Version(soft['version']) < Version('5.0.2'):
        run(WAPT.uninstall_cmd(soft['key']))
```

- pour chaque élément de la liste retournée par *installed_softwares* avec le mot clé *winscp* ;
- si la version dans la liste est plus petite que 5.0.2 ;
- alors lancer la désinstallation avec *uninstall_cmd* et en indiquant la *uninstallkey* ;

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://dev.tranquil.it/sphinxdocs/source/setuphelpers.html?highlight=uninstall_cmd#setuphelpers.uninstall_cmd

---

killalltasks

La commande **`killalltasks`** ...

```
killalltasks('firefox')
```

... termine l'exécution du logiciel *Firefox*.

Indication: Pour plus d'informations ou pour connaître les paramètres complémentaires de la commande, consultez la documentation de référence (en anglais) en visitant :

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=killalltasks#setuphelpers.killalltasks

---

Utiliser les champs du fichier control

```
def setup():
    print(control['version'])
```

... affiche le champ `version` du fichier `control` du paquet WAPT.

```
def setup():
    print(control['version'].split('-',1)[0])
```

... affiche le numéro de version du fichier `control` sans le numéro de version de packaging WAPT.

Appeler des actions WAPT dans des paquets WAPT

Installer un paquet

La commande **install** ...

```
WAPT.install('tis-scratch')
```

... installe *tis-scratch* sur la machine.

Supprimer un paquet

La commande **remove** ...

```
WAPT.remove('tis-scratch')
```

... désinstalle *tis-scratch* de la machine.

Oublier un paquet

La commande **forget_packages** ...

```
WAPT.forget_packages('tis-scratch')
```

... informe WAPT de ne plus suivre le paquet *tis-scratch* ; WAPT ne connaîtra plus l'existence de ce paquet.

Indication: Si vous voulez supprimer *tis-scratch*, il faudra soit réinstaller le paquet (**wapt-get install "tis-scratch"**), puis le supprimer (**wapt-get remove "tis-scratch"**), ou bien le supprimer manuellement à partir du panneau de configuration Windows *Ajout / Suppression de Programmes*.

6.14.12 Packager des paquets Linux simplifiés

Avant de commencer, nous assumons quelques hypothèses :

- vous avez une interface graphique sur votre linux ;
- vous avez installé le paquet **vscode** depuis les dépôts officiels de Tranquil IT ;
- votre utilisateur se nomme « linuxuser » et est membre du groupe des sudoers ;

Crer un modèle de paquet depuis votre poste linux

- ouvrir une invite de commande;
- en tant que *linuxuser*, créer un modèle de paquet WAPT ;

```
wapt-get make-template <template_name>
```

> **Avertissement:** Ne pas lancer cette commande en tant que root ou avec sudo.

Lorsque vous créez un modèle, plusieurs fichiers son créés dans le dossier `.vscode` à l'intérieur de votre dossier du paquet :

- settings.json ;
- launch.json ;

Exemple avec VLC:

```
wapt-get make-template "tis-vlc"

Using config file: /opt/wapt/wapt-get.ini
Template created. You can build the WAPT package by launching
/opt/wapt//wapt-get.py build-package /home/linuxuser/waptdev/tis-vlc-wapt
You can build and upload the WAPT package by launching
/opt/wapt//wapt-get.py build-upload /home/linuxuser/waptdev/tis-vlc-wapt
```

> **Indication:** Tous les paquets sont situés dans le dossier personnel de l'utilisateur *linuxuser*.

VSCode charge et s'ouvre sur le projet du paquet.

- vérifier le contenu du fichier `control` ;

Vous devez donner une **description** à votre paquet, renseignez l" **os_target** et la **version** de votre paquet.

> **Indication:** **os_target** pour unix est *linux*

> **Avertissement:** *version* dans votre fichier `control` doit commencer à 0 et non pas la version du logiciel, nous ne savons pas précisément quelle version sera récupérée depuis les dépôts apt/yum.

Fichier *control* original

```
package            : tis-vlc
version            : 0-0
architecture       : all
section            : base
priority           : optional
maintainer         : user
description        : automatic package for vlc
```

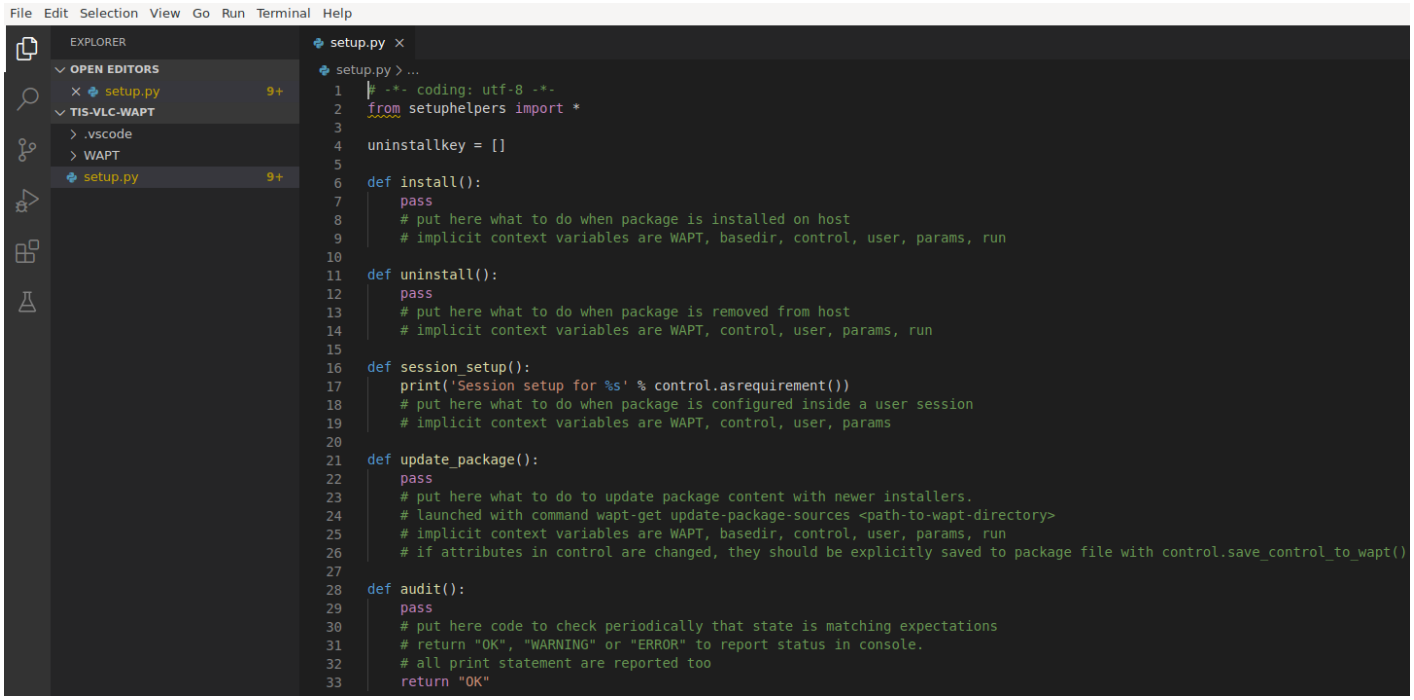


Figure164: VSCode s'ouvre en se focalisant sur le fichier *setup*

Fichier *control* modifié

```
package            : tis-vlc
version            : 0
architecture       : all
section            : base
priority           : optional
maintainer         : Tranquil-IT Systems
description        : VLC for linux
target_os          : linux
min_wapt_version   : 1.8
```

Note: Une sous-version *-1* a été ajoutée. C'est la version de packaging du paquet WAPT.

Cela permet de ne pas modifier la version réelle du logiciel quand on pousse par exemple une nouvelle configuration du même logiciel sur le parc.

- changer le code du fichier `setup.py` en conséquence ;

```
:emphasize-lines: 8
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    apt_install('vlc')
```

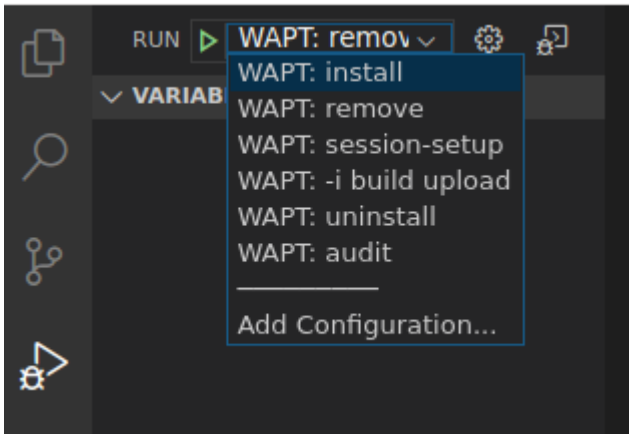
- enregistrer le paquet ;

Gérer la désinstallation

- changer le code du fichier `setup.py` avec une desinstallation ;

```
def uninstall():
apt_remove('vlc')
```

- lancer un *remove* du paquet depuis le panneau *Run Configurations* de VSCode ;



- vérifier que le programme est désinstallé

```
dpkg -l | grep vlc
```

Indication: Dans la fonction **`uninstall()`**, on ne peut pas appeler des fichiers contenus dans le paquet WAPT. Pour les appeler, il faudra avoir copié les fichiers dans un répertoire local de la machine lors de l'installation du paquet.

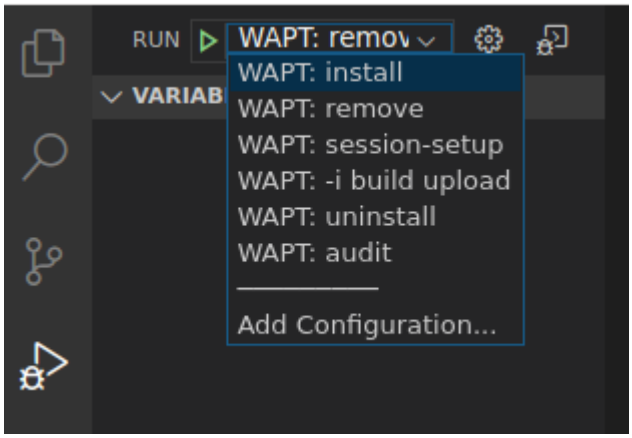
Gérer le session-setup

- changer le code du fichier `setup.py` avec un session-setup ;

  Dans cet exemple, vous aurez besoin d'un fichier `vlcrc` dans votre paquet pour le copier dans le dossier personnel de l'utilisateur. La fonction `ensure_dir` et la fonction `filecopyto` proviennent de la bibliothèque des **setuphelpers**, la première va tester si le chemin existe, la seconde va copier votre fichier depuis le paquet WAPT vers sa destination.

```
def session-setup():
  vlcdir = os.path.join(os.environ['HOME'], '.config', 'vlc')
  ensure_dir(vlcdir)
  filecopyto('vlcrc',vlcdir)
```

- lancer un *session-setup* du paquet depuis le panneau *Run Configurations* de VSCode ;



Construire et charger le paquet WAPT

Une fois l'installation et la désinstallation configurées et testées vous pouvez effectuer un **`build-upload`** pour charger votre nouveau paquet WAPT sur votre dépôt.

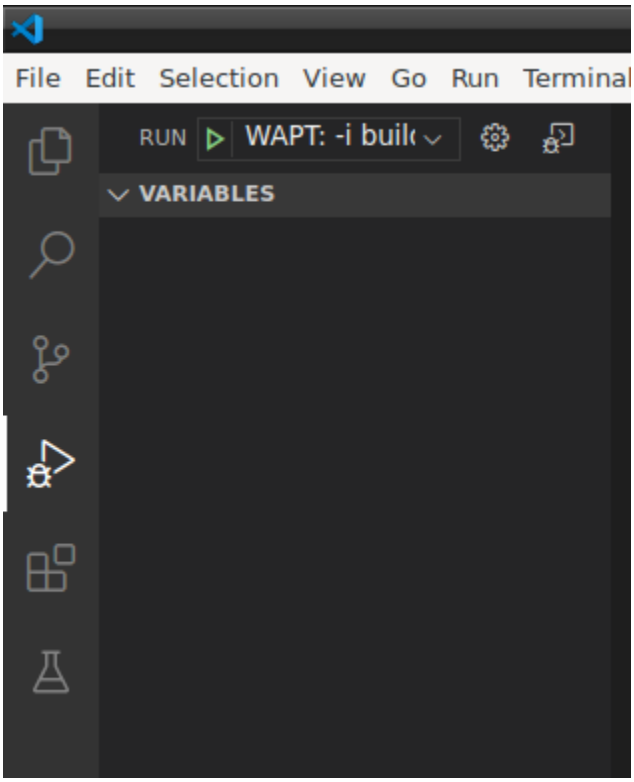
Si vous avez construit des paquets sur une autre machine (ex : Windows pour construire vos paquets WAPT Windows), vous devez copier vos clés *`.pem`* et *`.crt`* sur votre machine Linux avec **`WinSCP`** ou équivalent. Habituellement, ce paquet de certificats se trouve dans le dossier `C:\private` de votre ordinateur Windows. Ensuite, indiquez le chemin d'accès aux certificats dans le fichier `/opt/wapt/wapt-get.ini`.

```
sudo vim /opt/wapt/wapt-get.ini
```

- renseigner le chemin de votre certificat ;

```
personnal_certificate_path=/opt/wapt/private/mykey.crt
```

- Puis lancer un **`build-upload`** depuis le panneau VSCode *Run Configurations* ;



- renseigner le mot de passe de votre clé privée puis l'identifiant et le mot de passe de passe de votre console WAPT ;

Votre paquet est maintenant téléchargé et disponible dans votre dépôt privé sur votre serveur WAPT.

6.14.13 Automatiser la mise à jour d'un paquet logiciel

Note: Cette partie de la documentation est déconseillée aux utilisateurs qui débutent avec WAPT.

Pour quoi faire ?

Les fonctions *update_package* sont très pratiques, elles permettent de gagner du temps lorsque qu'il faut mettre à jour un paquet avec la version la plus récente d'un logiciel.

Fonctionnement

la fonction *update_package* paquet ira :

- récupérer la dernière version du logiciel en ligne ;
- télécharger la dernière version du binaire ;
- supprimer les anciennes version des binaires ;
- mettre à jour la version dans le fichier `control` ;

Si votre fonction *install* se base sur la version du fichier `control` pour l'installation, alors vous n'avez pas besoin de modifier votre `setup.py`.

Il vous reste maintenant à tester l'installation avant de lancer un **`build-upload`**.

Exemple

Voici l'"*update_package* de **`firefox-esr`** comme exemple :

```
def update_package():
      """ You can do a CTRL F9 in pyscripter to update the package """
      import re,requests,urlparse,glob

      url = requests.head('https://download.mozilla.org/?product=firefox-esr-latest&os=win&
↪lang=fr',proxies={}).headers['Location']
      filename = urlparse.unquote(url.rsplit('/',1)[1])

      if not isfile(filename):
          print('Downloading %s from %s'%(filename,url))
          wget(url,filename)

      exes = glob.glob('*.exe')
      for fn in exes:
          if fn != filename:
              remove_file(fn)

      # updates control version from filename, increment package version.
      control = PackageEntry().load_control_from_wapt ('.')
      control.version = '%s-0'%(re.findall('Firefox Setup (.*)esr\.exe',filename)[0])
      control.save_control_to_wapt('.')

  if __name__ == '__main__':
      update_package()
```

Vous pouvez lancer l'update_package* en appuyant sur la touche `F9` dans **`PyScripter`**.

Vous trouverez de nombreux exemples d'"*update_package* qui vous inspireront dans les paquets du store de Tranquil IT.

6.14.14 Créer des paquets WAPT de mises à jour Windows avec des .msu

Indication: Pré-requis : pour construire des paquets WAPT, *l'environnement de développement WAPT doit être installé* ;

Entre les sorties de *Patch Tuesday*, Microsoft peut publier des KB supplémentaires ou des mises à jour critiques qui devront être rapidement poussées sur les machines.

À cette fin, le WAPT fournit un modèle de paquet pour les fichiers *.msu.

Dans cet exemple, nous utilisons la KB4522355 téléchargée du site officiel Microsoft.

- télécharger le paquet KB depuis le catalogue Microsoft :
  - télécharger KB4522355 MSU ;

Créer un modèle de paquet MSU à partir de la console WAPT

- créer un modèle de paquet WAPT à partir du fichier MSU téléchargé ;

  Dans la console WAPT, cliquez sur *Outils* → *Assistant de paquets* ;

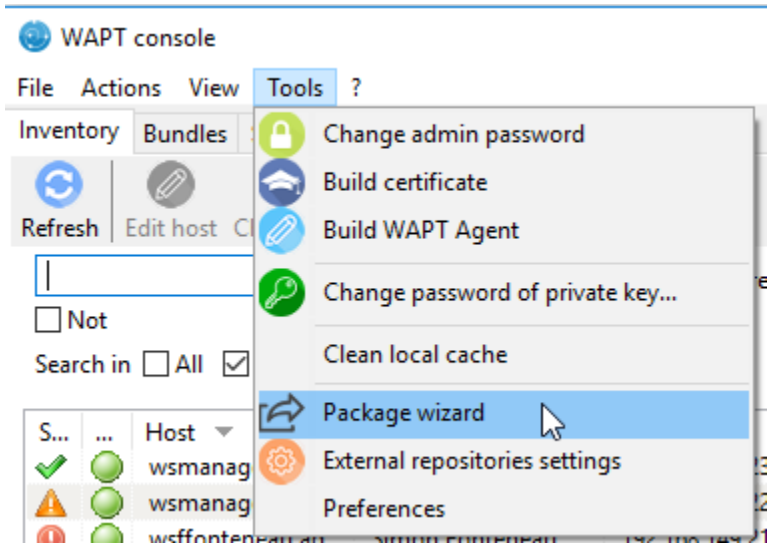


Figure165: Pyscripter - Fenêtre de la console WAPT pour créer un modèle de paquet

- sélectionnez le paquet MSU téléchargé et remplissez les champs requis ;
- cliquer sur *Créer et éditer . . . .* (recommandé) pour lancer la personnalisation du paquet ;
- L'éditeur de code intégré se lance en utilisant le code source du modèle MSU prédéfini.
- comme d'habitude avec les paquets WAPT, tester - assembler - signer - charger sur le dépôt - affecter aux hôtes et c'est fait !!
- si le KB est groupé avec le *Patch Tuesday* suivant, vous pourrez sélectionner les hôtes sur lesquels le paquet a été appliqué et oublier le paquet KB sur les hôtes ;

Créer un modèle de paquet MSU en ligne de commande

- lancer un utilitaire en ligne de commande Windows **`cmd.exe`** en tant qu"*Administrateur Local* ;
- instancier un paquet à partir du modèle MSU prédéfini ;

```
wapt-get make-template c:\download\file.msu <yourprefix>-kb4522355
```

- exemple avec **`KB4522355`** :

```
C:\WINDOWS\system32>wapt-get make-template C:\windows10.0-kb4522355-x64_
↪af588d16a8fbb572b70c3b3bb34edee42d6a460b.msu tis-kb4522355
Using config file: C:\Users\user-adm\AppData\Local\waptconsole\waptconsole.ini

Template created. You can build the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-package c:\waptdev\tis-kb4522355-wapt
```

(suite sur la page suivante)

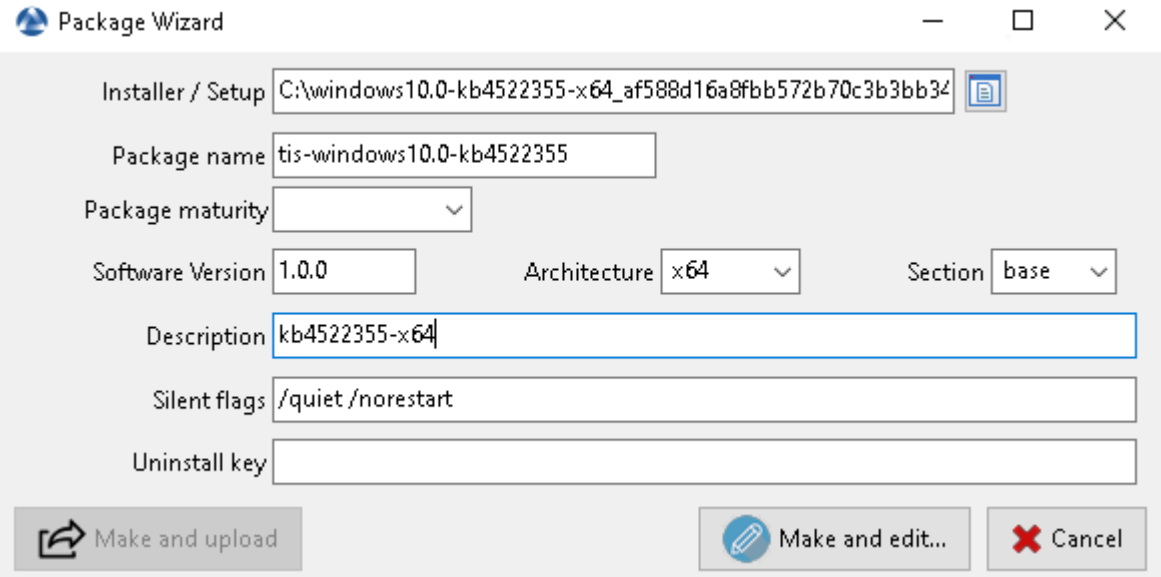


Figure166: Informations requises pour la création du paquet MSU

(suite de la page précédente)

```
You can build and upload the WAPT package by launching
  C:\Program Files (x86)\wapt\wapt-get.exe build-upload c:\waptdev\tis-kb4522355-wapt
```

- L'IDE configuré dans votre console se lance, voici un exemple de code source du modèle MSU prédéfini:

```
# -*- coding: utf-8 -*-
from setuphelpers import *
import re

uninstallkey = []

def is_kb_installed(hotfixid):
    installed_update = installed_windows_updates()
    if [kb for kb in installed_update if kb['HotFixID' ].upper() == hotfixid.upper()]:
        return True
    return False

def waiting_for_reboot():
    # Query WUAU from the registry
    if reg_key_exists(HKEY_LOCAL_MACHINE,r"SOFTWARE\Microsoft\Windows\CurrentVersion\
↪WindowsUpdate\Auto Update\RebootRequired") or \
        reg_key_exists(HKEY_LOCAL_MACHINE,r"SOFTWARE\Microsoft\Windows\CurrentVersion\
↪Component Based Servicing\RebootPending") or \
        reg_key_exists(HKEY_LOCAL_MACHINE,r'SOFTWARE\Microsoft\Updates\UpdateExeVolatile'):
        return True
    return False

def install():
    kb_files = [
        'windows10.0-kb4522355-x64_af588d16a8fbb572b70c3b3bb34edee42d6a460b.msu',
```

(suite sur la page suivante)

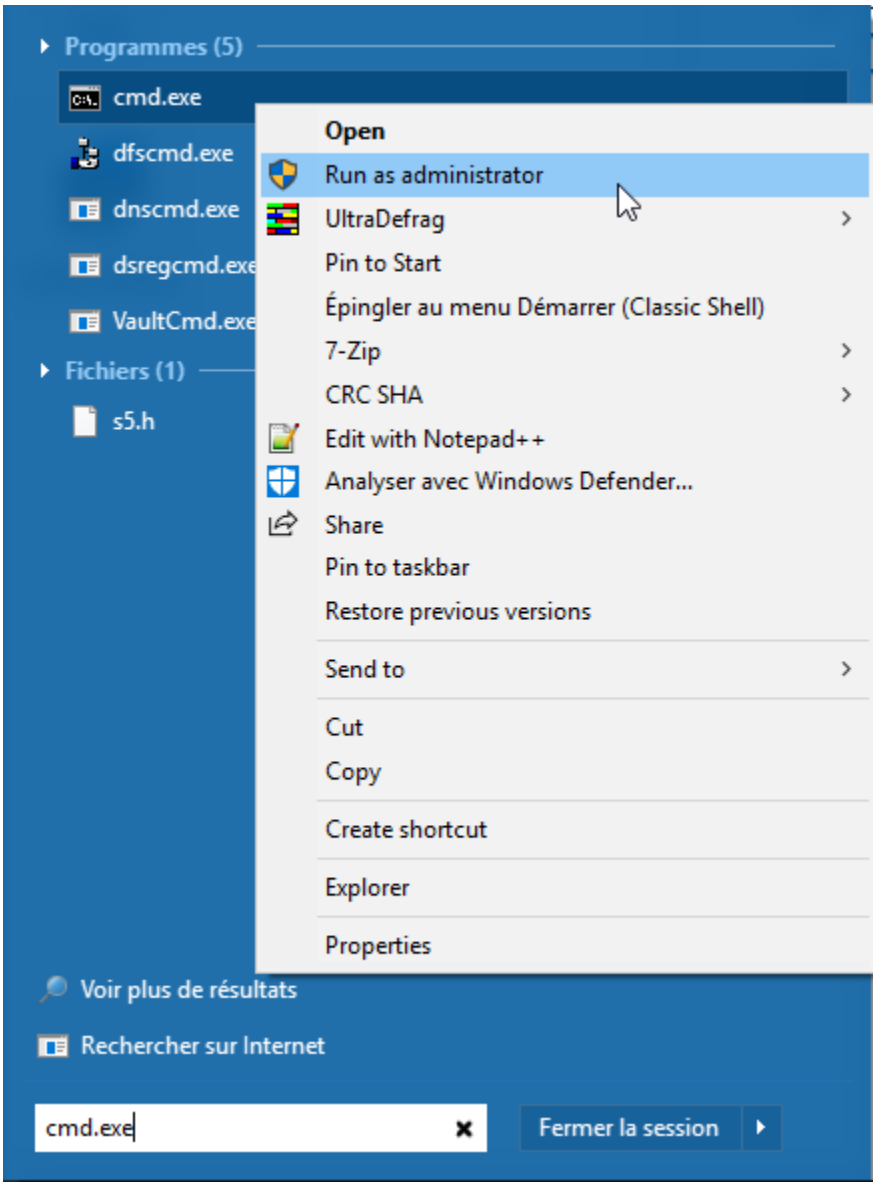


Figure167: Lancement de l'utilitaire de ligne de commande Windows en tant qu'administrateur local

(suite de la page précédente)

```
        ]
    with EnsureWUAUServRunning():
      for kb_file in kb_files:
          kb_guess = re.findall(r'^.*-(KB.*)-',kb_file)
          if not kb_guess or not is_kb_installed(kb_guess[0]):
              print('Installing {}'.format(kb_file))
              run('wusa.exe "{}" /quiet /norestart'.format(kb_file),accept_returncodes=[0,
↪3010,2359302,-2145124329],timeout=3600)
          else:
              print('{} already installed'.format(kb_file))

      if waiting_for_reboot():
          print('A reboot is needed!')
```

6.14.15 Chiffrer certaines données sensibles contenues dans un paquet WAPT

Note: Cette partie de la documentation est déconseillée aux utilisateurs qui débutent avec WAPT.

Pour quoi faire ?

Dans le fonctionnement de WAPT, l'intégrité du paquet est assurée. Un paquet dont le contenu a été modifié sans avoir été re-signé sera systématiquement refusé par le client WAPT.

En revanche le contenu d'un paquet WAPT n'est pas chiffré et sera lisible de tous. Ce modèle technique de transparence apporte cependant de nombreux bénéfices.

Cela peut être génant dans le cas d'un paquet qui contiendrait un mot de passe, une clé de licence, ou une donnée sensible.

Heureusement **nous avons une solution.**!!

Principe de fonctionnement du chiffrement d'une partie du contenu d'un paquet WAPT

Lorsque qu'un agent WAPT s'enregistre auprès du serveur WAPT, il génère un couple clé privée / certificat public dans `C:Program Files (x86)\waptprivate`.

- Le certificat est envoyé au serveur avec l'inventaire lors de l'enregistrement initial du client WAPT ;
- La clé privée est conservée par l'agent et n'est accessible en lecture que par les *Administrateurs Locaux* ;

Nous allons donc chiffrer la donnée sensible contenue dans le paquet avec le certificat appartenant à la machine.

Lors de l'installation l'agent WAPT pourra ainsi déchiffrer la donnée sensible grâce à sa clé privé.

Avec ce mode de fonctionnement le serveur WAPT et les dépôts secondaires n'ont pas connaissance de la donnée sensible.

Cas pratique

Vous trouverez ici un exemple de paquet WAPT ou nous chiffrons un texte dans dans une fonction **update_package** puis nous déchiffrons ce texte dans la partie **install**.

Dans cet exemple, la fonction **update_package** nous permet de parcourir la base de donnée du serveur WAPT pour récupérer le certificat de chaque machine pour ensuite chiffrer le texte sensible avec celui-ci.

Le texte chiffré pour chaque machine est ensuite stocké dans un fichier `encrypt-txt.json` à la racine du paquet.

Lors de l'installation du paquet, l'agent WAPT prendra le texte chiffré et le déchiffrera avec sa clé privé.

```
# -*- coding: utf-8 -*-
from setuphelpers import *
import json
from waptcrypto import SSLCertificate
import waptguihelper

uninstallkey = []

def install():
    encryptlist = json.loads(open('encrypt-txt.json','r').read())
    if WAPT.host_uuid in encryptlist:
        host_key = WAPT.get_host_key()
        encrypttxt = host_key.decrypt(encryptlist[WAPT.host_uuid].decode('base64')).decode('utf-8
↪')
        print( ur'Here is the deciphered text:  %s' % encrypttxt)
    else:
        error('%s not found in encrypt-txt.json' % WAPT.host_uuid)

def update_package():
    urlserver = inifile_readstring(makepath(install_location('WAPT_is1'),'wapt-get.ini'),'global
↪','wapt_server').replace('https://','')
    encrypttxt = str(raw_input('Enter the text to be encrypted :').encode('utf-8'))
    encryptlist = {}
    credentials_url = waptguihelper.login_password_dialog('Credentials for wapt server',
↪urlserver,'admin','')
    data = json.loads(wgets('https://%s:%s@%s/api/v1/hosts?columns=host_certificate&limit=10000'
↪% (credentials_url['user'],credentials_url['password'],urlserver)))
    for value in data['result']:
        if value['host_certificate']:
            host_cert=SSLCertificate(crt_string=value['host_certificate'])
            encryptlist[value['uuid']]=host_cert.encrypt(encrypttxt).encode('base64')
            print value['computer_fqdn'] + ':' + value['uuid'] + ':' + encryptlist[value['uuid']]
    open('encrypt-txt.json','w').write(json.dumps(encryptlist))

if __name__ == '__main__':
    update_package()
```

Attention: La sortie python (log install du paquet) est accessible en lecture aux utilisateurs de la machine, **vous ne devez donc pas afficher le text déchiffré avec un :command:`print` lors de l'installation**.

6.14.16 Travailler avec des codes de retour non standard

Les codes de retour sont utilisés pour indiquer si un logiciel a été correctement installé.

Dans Windows, le code standard de retour réussi est [0].

Si vous savez que vos paquets WAPT s'installent correctement, mais que vous obtenez quand même un code de retour autre que [0], alors vous pouvez explicitement dire à WAPT d'ignorer le code d'erreur en utilisant le paramètre `accept_returncodes`.

Vous pouvez découvrir comment utiliser le paramètre `accept_returncodes` en explorant cet exemple de code.

```
# -*- coding: utf-8 -*-
from setuphelpers import *
import re

uninstallkey = []

def is_kb_installed(hotfixid):
    installed_update = installed_windows_updates()
    if [kb for kb in installed_update if kb['HotFixID' ].upper() == hotfixid.upper()]:
        return True
    return False

def waiting_for_reboot():
    # Query WUAU from the registry
    if reg_key_exists(HKEY_LOCAL_MACHINE,r"SOFTWARE\Microsoft\Windows\CurrentVersion\
↪WindowsUpdate\Auto Update\RebootRequired") or \
        reg_key_exists(HKEY_LOCAL_MACHINE,r"SOFTWARE\Microsoft\Windows\CurrentVersion\Component␣
↪Based Servicing\RebootPending") or \
        reg_key_exists(HKEY_LOCAL_MACHINE,r'SOFTWARE\Microsoft\Updates\UpdateExeVolatile'):
        return True
    return False

def install():
    kb_files = [
        'windows10.0-kb4522355-x64_af588d16a8fbb572b70c3b3bb34edee42d6a460b.msu',
        ]
    with EnsureWUAUServRunning():
      for kb_file in kb_files:
          kb_guess = re.findall(r'^.*-(KB.*)-',kb_file)
          if not kb_guess or not is_kb_installed(kb_guess[0]):
              print('Installing {}'.format(kb_file))
              run('wusa.exe "{}" /quiet /norestart'.format(kb_file),accept_returncodes=[0,3010,
↪2359302,-2145124329],timeout=3600)
          else:
              print('{} already installed'.format(kb_file))

      if waiting_for_reboot():
          print('A reboot is needed!')
```

Indication: La liste complète des messages d'erreur de l'installateur Windows peut être consultée sur cette page.

6.14.17 Utiliser un autre environnement de développement pour WAPT

Introduction

Si vous êtes habitué(e) à travailler avec un autre *IDE*, vous pouvez être soulagé maintenant car WAPT supporte d'autres éditeurs de développement intégrés.

Certains éditeurs de code sont pris en charge en natif :

- PyScripter;
- VSCode ;
- VSCodium ;

D'autres éditeurs peuvent être sélectionnés et seront lancés lorsque vous créerez un nouveau modèle pour un paquet WAPT à partir de la console WAPT.

Configurer WAPT pour utiliser un autre IDE

Note: Utiliser un IDE supporté lancera le projet de paquet WAPT avec une configuration de débogage valide.

Utiliser Microsoft Windows

Pour configurer un autre éditeur pour WAPT, vous devez modifier l'attribut `editor_for_packages` dans la section `[global]` du fichier de configuration `%LOCALAPPDATA%waptconsolewaptconsole.ini` de votre console WAPT.

Les valeurs possibles sont :

Nom de l'éditeur de code	editor_for_packages value
PyScripter	None
Microsoft Visual Studio Code	**`vscode`** ou **`code`**
Microsoft Visual Studio Codium	**`vscodium`** ou **`codium`**

Exemple de configuration de `waptconsole.ini` :

```
[global]
...
editor_for_packages=vscode
```

Utiliser Linux / Apple macOS

Pour configurer un autre éditeur pour WAPT, vous devez modifier l'attribut `editor_for_packages` dans la section `[global]` du fichier de configuration `/opt/wapt/wapt-get.ini` de votre agent WAPT.

Par défaut, si l'attribut `editor_for_packages` est vide, le WAPT essaiera de lancer (dans cet ordre) :

- **`vscodium`** ;
- **`vscode`**;

- **nano** ;
- **vim** ;
- **vi** ;

Les valeurs possibles sont :

Nom de l'éditeur de code	editor_for_packages value
Microsoft Visual Studio Code	**vscode** ou **code**
Microsoft Visual Studio Codium	**vscodium** ou **codium**
Nano	**nano**
Vim	**vim**
Vi	**vi**

```
[global]
...
editor_for_packages=vim
```

Configurer WAPT pour utiliser un éditeur de code personnalisé

Utiliser Microsoft Windows

Des éditeurs de code personnalisés peuvent être utilisés, par exemple **Notepad++** ou **PyCharm**.

Exemple d'éditeurs de code personnalisés :

Nom de l'éditeur de code	editor_for_packages value
Notepad++	`C:\Program Files\Notepad++\notepad++.exe` *`setup_filename`*
PyCharm	`C:\Program Files\JetBrains\PyCharm Community Edition 2019.3.2\bin\pycharm64.exe` *`wapt_sources_dir`*

```
[global]
...
editor_for_packages=C:\Program Files\Notepad++\notepad++.exe {setup_filename}
```

Utiliser Linux/Apple macOS

Des éditeurs de code personnalisés peuvent être utilisés, par exemple **PyCharm**.

Exemple d'éditeurs de code personnalisés :

Nom de l'éditeur de code	editor_for_packages value
PyCharm	`/opt/pycharm/bin/pycharm_x64` *`wapt_sources_dir`*

```
[global]
...
editor_for_packages=/opt/pycharm/bin/pycharm_x64 {wapt_sources_dir}
```

Arguments personnalisés

Les arguments peuvent être passés dans la commande `editor_for_packages` :

Argument	Description
`{setup_filename}`	Lance l'éditeur de code personnalisé et édite le fichier WAPT setup.py
`{control_filename}`	Lance l'éditeur de code personnalisé et modifie le fichier control des paquets WAPT
`{wapt_sources_dir}`	Lance l'éditeur de code personnalisé et ouvre le dossier du paquet WAPT
`{wapt_base_dir}`	Lance l'éditeur de code personnalisé et ouvre le dossier d'installation WAPT

6.14.18 Vidéos montrant WAPT en action

Créer et déployer un paquet msi avec WAPT

Créer, configurer et déployer un paquet exe avec WAPT

6.15 Utiliser l'API du serveur WAPT

Note: Cette documentation ne décrit pas toutes les APIs (Application Protocol Interfaces) disponibles, mais va cependant se concentrer sur les plus utiles.

Toutes les URLs disponibles peuvent être trouvées dans `/opt/wapt/waptserver/server.py`.

Les URLs sont formées en utilisant la bonne commande depuis le serveur WAPT ex: `https://srvwapt/`*`command_path`*.

Indication: Cette documentation contient des exemples en code Python ou bien en curl.

6.15.1 API V1

/api/v1/hosts

- récupérer les données enregistrées d'un ou de plusieurs postes :

```
# Args:
#     has_errors (0/1): filter out hosts with packages errors
#     need_upgrade (0/1): filter out hosts with outdated packages
#     groups (csvlist of packages): hosts with packages
#     columns (csvlist of columns):
#     uuid (csvlist of uuid): <uuid1[,uuid2,...]>): filter based on uuid
#     filter (csvlist of field):regular expression: filter based on attributes
#     not_filter (0,1):
#     limit (int): 1000
#     trusted_certs_sha256 (csvlist): filter out machines based on their trusted package␣
↪certs

# Returns:
```

(suite sur la page suivante)

(suite de la page précédente)

```
#     result (dict): {'records':[],'files':[]}
#     query:
#       uuid=<uuid>
#     or
#       filter=<csvlist of fields>:regular expression
# """
```

- liste tous les postes. Les paramètres disponibles sont :
  - *reachable*
  - *computer_fqdn* ==> computer_name
  - *connected_ips*
  - *mac_addresses*

  Cette exemple montre une requête avec des paramètres:

```
advanced_hosts_wapt = wgets('https://%s:%s@%s/api/v1/hosts?columns=reachable,computer_fqdn,
↪connected_ips,mac_addresses&limit=10000' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(advanced_hosts_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

  Cette exemple est une requête globale:

```
hosts_wapt = wgets('https://%s:%s@%s/api/v1/hosts' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(hosts_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/hosts
```

Ceui-ci donne une requête avec un statut joignable, le nom de la machine, ses IP connectées et ses adresses MAC. La limite d'affichage est de 10000 postes

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/hosts?columns=reachable,
↪computer_fqdn,connected_ips,mac_addresses&limit=10000
```

/api/v1/groups

- récupère tous les paquets groupes. Les groupes peuvent être trouvés avec la section *groupe* dans le paquet.

```
group_wapt = wgets('https://%s:%s@%s/api/v1/groups' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(group_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/groups
```

/api/v1/host_data

dmi

- récupère toutes les informations DMI (Desktop Management Interface) d'un poste :

Note: # # Récupère des données supplémentaires d'un poste # query: # uuid=<uuid> # field=packages, dmi ou softwares

Exemple: récupérer les informations *DMI* du poste ayant l'UUID 14F620FF-DE70-9E5B-996A-B597E8F9B4AD: https://srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-B597E8F9B4AD&field=dmi

Note: le *dmi* n'est pas la seule option disponible. Vous pouvez aussi chercher des informations en utilisant *installed_packages*, *wsusupdates* ou *installed_softwares*.

```
dmi_host_data_wapt = wgets('https://%s:%s@%s/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=dmi' % (wapt_user,wapt_password,wapt_url))
#print(dmi_host_data_wapt)
parsed = json.loads(dmi_host_data_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=dmi
```

installed_packages

L'option *installed_packages* va lister tous les paquets installés sur un poste en particulier.

```
install_packages_data_wapt = wgets('https://%s:%s@%s/api/v1/host_data?uuid=14F620FF-DE70-9E5B-
↪996A-B597E8F9B4AD&field=installed_packages' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(install_packages_data_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=installed_packages
```

installed_softwares

L'option *installed_softwares* va lister tous les logiciels installés sur un poste en particulier.

```
install_softwares_data_wapt = wgets('https://%s:%s@%s/api/v1/host_data?uuid=14F620FF-DE70-9E5B-
↪996A-B597E8F9B4AD&field=installed_softwares' % (wapt_user,wapt_password,wapt_url))
#print(install_softwares_data_wapt)
parsed = json.loads(install_softwares_data_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=installed_softwares
```

wsusupdates

L'option *wsusupdates* va lister toutes les mises à jour installés sur un poste en particulier.

```
wsusupdates_data_wapt = wgets('https://%s:%s@%s/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=wsusupdates' % (wapt_user,wapt_password,wapt_url))
#print(wsusupdates_data_wapt)
parsed = json.loads(wsusupdates_data_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=wsusupdates
```

/api/v1/usage_statistics

Récupère les statistiques d'usage du serveur.

Indication: Cette API est utile si vous avez plusieurs serveurs WAPT et si vous voulez savoir combien de postes il y a.

```
usage_statistics_wapt =  wgets('https://%s:%s@%s/api/v1/usage_statistics' % (wapt_user,wapt_
↪password,wapt_url))
#print(usage_statistics_wapt)
parsed = json.loads(usage_statistics_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/usage_statistics
```

6.15.2 API V2

/api/v2/waptagent_version

Affiche la version du **waptagent.exe** sur le serveur.

```
waptagent_version =  wgets('https://%s:%s@%s/api/v2/waptagent_version' % (wapt_user,wapt_
↪password,wapt_url))
parsed = json.loads(waptagent_version)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication:

Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v2/waptagent_version
```

6.15.3 API V3

/api/v3/packages

Liste les paquets sur le dépôt privé, il récupère le fichier control sur les paquets.

```
packages_wapt =  wgets('https://%s:%s@%s/api/v3/packages' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(packages_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/packages
```

/api/v3/known_packages

Liste tous les paquets avec l'information *signed_on*.

```
known_packages_wapt =  wgets('https://%s:%s@%s/api/v3/known_packages' % (wapt_user,wapt_password,
↪wapt_url))
parsed = json.loads(known_packages_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/known_packages
```

/api/v3/trigger_cancel_task

Annule une tâche en cours.

```
trigger_cancel_task =  wgets('https://%s:%s@%s/api/v3/trigger_cancel_task' % (wapt_user,wapt_
↪password,wapt_url))
parsed = json.loads(trigger_cancel_task)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

/api/v3/get_ad_ou

Liste les OU vues par les postes et affichées dans la console WAPT.

```
get_ad_ou =  wgets('https://%s:%s@%s/api/v3/get_ad_ou' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(get_ad_ou)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_ou
```

/api/v3/get_ad_sites

Liste les sites Active Directory

```
get_ad_sites =  wgets('https://%s:%s@%s/api/v3/get_ad_sites' % (wapt_user,wapt_password,wapt_
↪url))
parsed = json.loads(get_ad_sites)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_sites
```

/api/v3/hosts_for_package

Liste les postes qui ont un paquet spécifique d'installé https://srvwapt.mydomain.lan/api/v3/hosts_for_package?package=demo-nompaquet

```
hosts_for_package =  wgets('https://%s:%s@%s/api/v3/hosts_for_package?package=demo-namepackage'
↪% (wapt_user,wapt_password,wapt_url))
parsed = json.loads(hosts_for_package)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/hosts_for_package?package=demo-namepackage
```

/api/v3/host_tasks_status

Liste les tâches d'un poste en particulier.

Exemple avec l'uuid d'un poste: https://srvwapt.mydomain.lan/api/v3/host_tasks_status?uuid=14F620FF-DE70-9E5B-996A-B597E8F9B4AD

```
host_tasks_status =  wgets('https://%s:%s@%s/api/v3/host_tasks_status?uuid=14F620FF-DE70-9E5B-
↪996A-B597E8F9B4AD' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(host_tasks_status)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

Indication: Voici le même exemple avec une simple requête html :

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/host_tasks_status?uuid=14F620FF-DE70-9E5B-
↪996A-B597E8F9B4AD
```

Attention: **Les API ci-après suivent la méthode POST**.

/api/v3/upload_packages

À faire: Tests

/api/v3/upload_hosts

À faire: Tests

/api/v3/change_password

Change le mot de passe du compte admin [ce compte uniquement]. La requête doit être un dictionnaire python *{}*. Les clés doivent être:

- user
- password
- new_password

```
curl --insecure -X POST --data-raw '{"user":"admin","password":"OLDPASSWORD","new_password":
↪"NEWPASSWORD"}' -H "Content-Type: application/json" "https://admin:OLDPASSWORD@srvwapt/api/v3/
↪change_password"
```

/api/v3/login

Initialiser une connexion au serveur.

```
curl --insecure -X POST --data-raw '{"user":"admin","password":"MYPASSWORD"}' -H "Content-Type:␣
↪application/json" "https://srvwapt.mydomain.lan/api/v3/login"

{"msg": "Authentication OK", "result": {"edition": "enterprise", "hosts_count": 6, "version": "1.
↪7.4", "server_domain": "mydomain.lan", "server_uuid": "32464dd6-c261-11e8-87be-cee799b43a00"},
↪"success": true, "request_time": 0.03377699851989746}
```

Indication: Nous pouvons faire une connexion avec un formulaire html plutôt que POST: https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_sites

/api/v3/packages_delete

Supprime un paquet d'une version précise. La requête doit être une liste *[]*. Elle peut prendre plusieurs paquets séparés par des virgules ,.

Exemple:

```
curl --insecure -X POST --data-raw '["demo-libreoffice-stable_5.4.6.2-3_all.wapt"]' -H "Content-
↪Type: application/json" "https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/packages_delete"
```

/api/v3/reset_hosts_sid

Il y a plusieurs possibilités : https://srvwapt.mydomain.lan/api/v3/reset_hosts_sid va réinitialiser toutes les connexions des postes.

Pour la méthode POST:

La syntaxe est : **`--data-raw`** : un dictionnaire avec pour clé les `uuid` et pour valeur l'uuid du poste.

```
curl --insecure -X POST --data-raw '{"uuids":["114F620FF-DE70-9E5B-996A-B597E8F9B4C"]}' -H
↪"Content-Type: application/json" "https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/reset_
↪hosts_sid"

{"msg": "Hosts connection reset launched for 1 host(s)", "result": {}, "success": true, "request_
↪time": null}[
```

Indication: Si vous voulez plusieurs postes:

```
curl --insecure -X POST --data-raw '{"uuids":["114F620FF-DE70-9E5B-996A-B597E8F9B4C","04F98281-
↪7D37-B35D-8803-8577E0049D15"]}' -H "Content-Type: application/json" "https://
↪admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/reset_hosts_sid"

{"msg": "Hosts connection reset launched for 2 host(s)", "result": {}, "success": true, "request_
↪time": null}
```

/api/v3/trigger_wakeonlan

Si les postes ont le WakeOnLan d'activé, cette API est utile.

La syntaxe est : **`--data-raw`** : un dictionnaire avec pour clé les `uuid` et pour valeur l'uuid du poste :

```
curl --insecure -X POST --data-raw '{"uuids":["04F98281-7D37-B35D-8803-8577E0049D15"]}' -H
↪"Content-Type: application/json" "https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/trigger_
↪wakeonlan"

{"msg": "Wakeonlan packets sent to 1 machines.", "result": [{"computer_fqdn": "win10-1809.
↪mydomain.lan", "mac_addresses": ["7e:c4:f4:9a:87:2d"], "uuid": "04F98281-7D37-B35D-8803-
↪8577E0049D15"}], "success": true, "request_time": null}
```

Indication: Si vous voulez plusieurs postes:

```
curl --insecure -X POST --data-raw '{"uuids":["04F98281-7D37-B35D-8803-8577E0049D15","14F620FF-
↪DE70-9E5B-996A-B597E8F9B4AD"]}' -H "Content-Type: application/json" "https://
↪admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/trigger_wakeonlan"

{"msg": "Wakeonlan packets sent to 2 machines.", "result": [{"computer_fqdn": "win10-1803.
↪mydomain.lan", "mac_addresses": ["02:4f:25:74:67:71"], "uuid": "14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD"}, {"computer_fqdn": "win10-1809.ad.alejeune.fr", "mac_addresses": [
↪"7e:c4:f4:9a:87:2d"], "uuid": "04F98281-7D37-B35D-8803-8577E0049D15"}], "success": true,
↪"request_time": null}
```

/api/v3/hosts_delete

```
"""Remove one or several hosts from Server DB and optionnally the host packages

Args:
    uuids (list): list of uuids to delete
    filter (csvlist of field:regular expression): filter based on attributes
    delete_packages (bool): delete host's packages
    delete_inventory (bool): delete host's inventory

Returns:
    result (dict):
"""
```

Si vous voulez supprimer un poste de l'inventaire:

```
curl --insecure -X POST --data-raw '{"uuids":["04F98281-7D37-B35D-8803-8577E0049D15"],"delete_
↪inventory":"True","delete_packages":"True"}' -H "Content-Type: application/json" "https://
↪admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/hosts_delete"

{"msg": "1 files removed from host repository\n1 hosts removed from DB", "result": {"files": ["/
↪var/www/wapt-host/04F98281-7D37-B35D-8803-8577E0049D15.wapt"], "records": [{"computer_fqdn":
↪"win10-1809.mydomain.lan", "uuid": "04F98281-7D37-B35D-8803-8577E0049D15"}]}, "success": true,
↪"request_time": null}
```

Si vous ne voulez pas le supprimer de l'inventaire du serveur:

```
curl --insecure -X POST --data-raw '{"uuids":["04F98281-7D37-B35D-8803-8577E0049D15"],"delete_
↪inventory":"False","delete_packages":"False"}' -H "Content-Type: application/json" "https://
↪admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/hosts_delete"

{"msg": "0 files removed from host repository\n1 hosts removed from DB", "result": {"files": [],
↪"records": [{"computer_fqdn": "win10-1809.mydomain.lan", "uuid": "04F98281-7D37-B35D-8803-
↪8577E0049D15"}]}, "success": true, "request_time": null}
```

/api/v3/trigger_host_action

À faire: Tests

/upload_waptsetup

```
# Upload waptsetup

#Handle the upload of customized waptagent.exe into wapt repository

### NE MARCHE PAS
#curl --insecure -X POST -H  "Content-Type: multipart/form-data" -F 'data=@waptagent.exe'
↪"https://admin:MYPASSWORD@srvwapt.mydomain.lan/upload_waptsetup"
```

/ping

Ping va récupérer les informations générales d'un serveur WAPT.

```
# https://srvwapt.mydomain.lan/ping
# Liste les infos du serveur

ping_wapt =  wgets('https://%s:%s@%s/ping' % (wapt_user,wapt_password,wapt_url))
parsed = json.loads(ping_wapt)
print(json.dumps(parsed, indent=1, sort_keys=True))
```

6.16 Problèmes fréquents

6.16.1 J'ai perdu mon mot de passe SuperAdmin

Il arrive parfois de configurer un serveur WAPT et d'oublier son mot de passe.

Pour réinitialiser le mot de passe *SuperAdmin* de la console WAPT, vous devez relancer le processus de post-configuration sur le serveur WAPT.

Réinitialiser le mot de passe du serveur WAPT Linux

- se connecter au serveur avec SSH ;
- se connecter avec l'utilisateur root (ou utiliser sudo) ;
- lancer le script de post-configuration :

```
/opt/wapt/waptserver/scripts/postconf.sh
```

Attention: Pour éviter de casser la configuration existante du serveur WAPT, acceptez toutes les autres étapes, **NE CRÉEZ PAS une nouvelle clé privée** !

6.16.2 J'ai perdu ma clé privée WAPT

La sécurité et le bon fonctionnement de WAPT s'appuient sur les jeux de clés privés et de certificats publics.

La perte d'une clé privée nécessite donc de regénérer une nouvelle clé et les certifcats associés, et ensuite déployer sur le parc de machines les certificats pour la nouvelle clé.

Donc la procédure n'est pas anodine, même si elle est simple.

Procédure de renouvellement ou de création d'une clé privée

La procédure va être la suivante :

- générer une nouvelle clé privée/certificat public. Vous conserverez alors la clé privée (fichier *.pem*) dans un endroit sûr ;
- déployer le nouveau certificat *.crt* sur vos clients dans le dossier `C:\Program Files (x86)\ssl` manuellement ou en utilisant une GPO ;

Manipulation sur les paquets en conséquence

Les paquets du dépôt local étant signé avec l'ancienne clé, il convient de re-signer l'intégralité des paquets avec la nouvelle clé.

Afin de re-signer tous les paquets WAPT avec la nouvelle clé (les paquets logiciel et les paquets machine), utiliser la commande :

```
wapt-get sign-packages C:\\waptdev\\*
```

6.16.3 Je me suis fait voler ma clé privée

Attention: Toute la sécurité de WAPT repose sur la séquestration de cette clé privée.

WAPT ne gère pas encore la révocation des clés en utilisant une CRL.

La solution consiste à supprimer chaque certificat *.crt* associé à la clé privée volée, situé dans le dossier `C:\Program Files (x86)\wapt\ssl`.

Cette opération peut être effectuée par GPO ou manuellement.

6.16.4 Mon UUID BIOS bogue

- il arrive parfois qu'un problème survienne avec certains BIOS. WAPT utilise l'*UUID* de la machine comme identifiant pour reconnaître les machines ;
- l'*UUID* BIOS est censé être unique, malheureusement chez certains constructeurs et pour certaines séries de machines, les *UUID* des BIOS sont identiques ;
- le PC remontera bien dans la console mais il écrasera le PC déjà présent considérant que l'ordinateur a changé de nom ;

Résolution

WAPT permet de générer un *UUID* aléatoire pour remplacer celui indiqué dans le BIOS.

```
wapt-get generate-uuid
```

6.16.5 Mon WAPTdeploy ne fonctionne pas

Symptôme

L'utilitaire **waptdeploy** n'arrive pas à installer l'agent WAPT.

Résolution

Ajouter l'url pour waptagent.exe

Ajouter l'argument `waptsetupurl` dans les paramètres de la stratégie de déploiement **waptdeploy**.

```
--waptsetupurl=https://monserverserveurwapt/waptagent.exe
```

Lancer WAPTdeploy localement

Lancer **waptdeploy** localement peut mettre en évidence un problème en affichant explicitement les erreurs.

Exemple de commande à lancer :

```
C:\Program Files (x86)\wapt\waptdeploy.exe --
↪hash=2a9971aad083d6822b6e4d1ccfb9886be9429ec58bb13246810ff3d6a56ce887 --minversion=1.4.2.0 --
↪wait=15
```

Dans notre cas le hash n'est pas le bon.

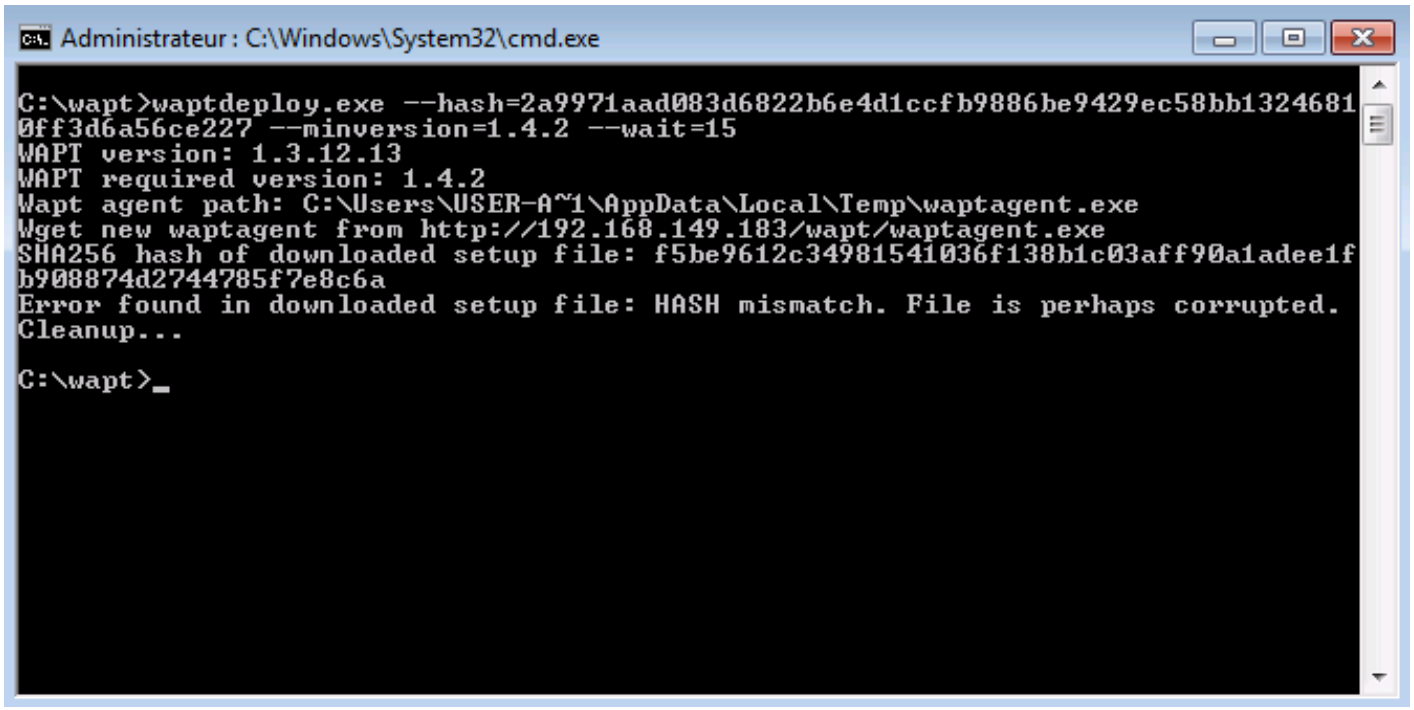


Figure168: Error with WAPTDeploy

Attention: Ne pas oublier de lancer l'invite de commande en tant qu'*Administrateur*.

WAPTdeploy fonctionne manuellement mais ne fonctionne pas via GPO

Vérifier que la GPO est bien appliquée avec la commande :

```
gpresult /h gpo.html & gpo.html
```

Pour forcer l'application des GPO :

```
gpupdate /force
```

Si **waptdeploy** n'apparaît pas, il faut re-vérifier la configuration de la GPO.

 # vous utilisez peut-être une ancienne version de waptdeploy, alors téléchargez la dernière version de **waptdeploy** sur le store WAPT.

 # merci à Emmanuel EUGENE de l'INSERM française qui a soumis cette cause possible du mauvais fonctionnement du **waptdeploy**, si vous répliquez des contrôleurs de domaine, assurez-vous que les GPOs sont correctement synchronisées entre vos DCs et que les ACLs sont appliquées de manière identique sur le `SysVols`.

Windows n'attend pas le réseau au démarrage

Par défaut Windows n'attend pas le réseau au démarrage de la machine.

Cela peut poser soucis pour l'exécution de **waptdeploy** car celui-ci a besoin du réseau au démarrage pour télécharger l'agent WAPT.

Vous pouvez activer la GPO : **Toujours attendre le réseau au démarrage et à la connexion**:

> *Computer → Configuration → Administrative Templates → System → Logon → Always wait for the network at computer startup and logon*

6.16.6 WAPTexit ne se lance pas

Malgré la présence du script dans les stratégies locales d'arrêt de la machine, **waptexit.exe** ne se lance pas à l'extinction du poste.

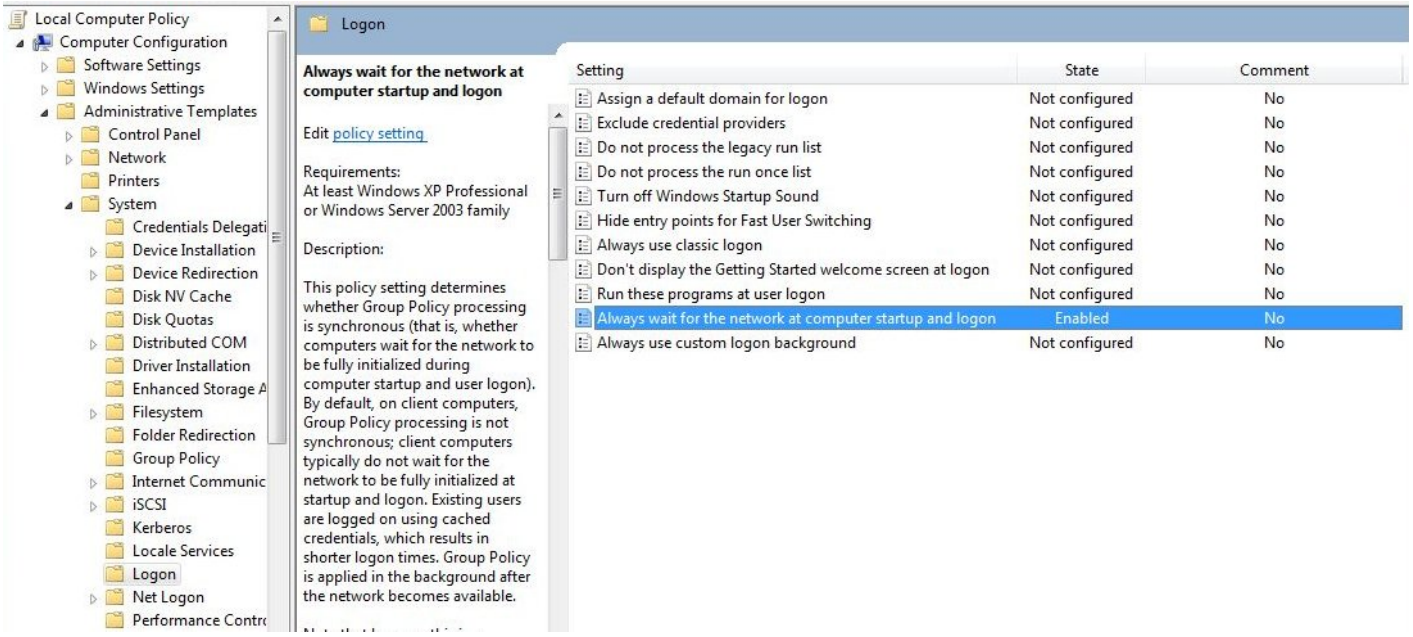
Résolution: Hybrid Shutdown

Il faut désactiver l'arrêt hybride de Windows10, qui cause par ailleurs beaucoup d'autres comportements étranges. La désactivation de l'arrêt hybride rétablit l'exécution des scripts à l'extinction de la machine.

L'arrêt hybride peut être désactivé en précisant une valeur dans le fichier `wapt-get.ini` de l'agent WAPT, voir *waptexit_ini_file*.

Un paquet WAPT existe pour traiter le problème :

- tis-disable-hybrid-shutdown.



Résolution: Windows Home édition

Les GPO n'étant pas présentes sur les versions familiales de Windows, il est normal qu'elles ne s'exécutent pas.

La solution de contournement est d'utiliser une tâche planifiée qui appelle `C:\Program Files (x86)\wapt\wapt-get.exe` avec le paramètre `upgrade`.

Résolution: GPO locale corrompue

Il peut arriver que les stratégies de groupes locales de la machine soient corrompues.

Une des solutions consiste à supprimer les stratégies locales actuelles en supprimant le fichier `C:\windows\system32\GroupPolicy\gpt.ini`, puis en redémarrant la machine, et enfin en relançant l'installation de la tâche d'extinction :

```
wapt-get add-upgrade-shutdown
```

Si le problème se reproduit, cela signifie peut être qu'une autre application manipule également la GPO.

6.16.7 WAPTexit se coupe après 15 minutes et n'achève pas l'installation

Par défaut sous Windows, les scripts d'extinction ne peuvent s'exécuter plus de 15 minutes.

Si à l'arrêt de la machine, un script d'extinction n'a pas rendu la main au bout de 15 minutes, le script est interrompu.

Solution : augmenter le délai d'installation

Pour résoudre le soucis, il faut modifier la valeur `preshutdowntimeout` ainsi que la valeur `max_gpo_script_wait`.

Définissez ces valeurs dans `C:\Program Files (x86)\wapt\wapt-get.ini` pour modifier le comportement par défaut.

```
max_gpo_script_wait=180
pre_shutdown_timeout=180
```

Le paquet tis-wapt-conf-policy embarque cette configuration.

L'autre solution est d'utiliser la GPO `File.ini`.

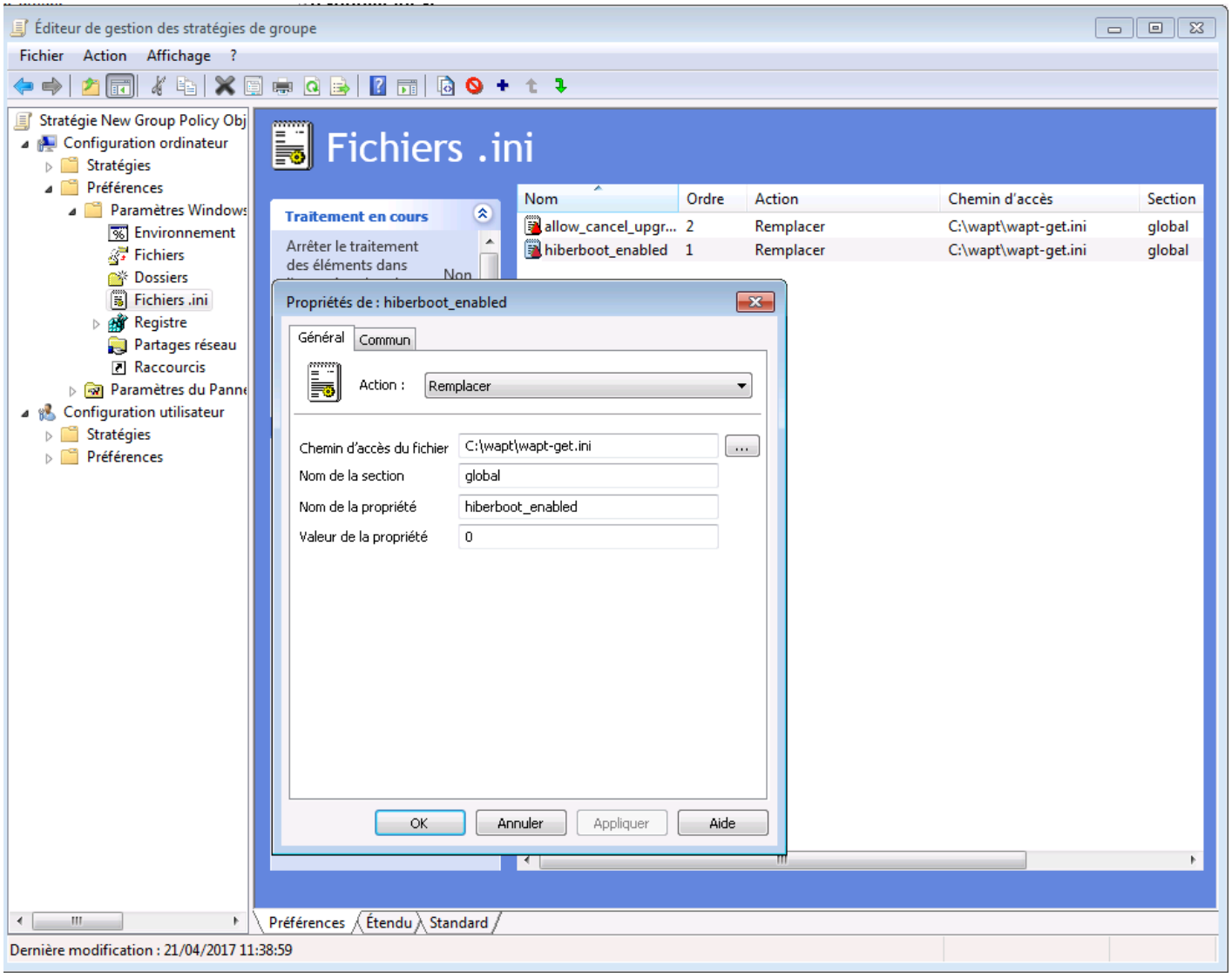


Figure169: GPO ini File

6.16.8 Message d'erreur à l'ouverture de la console

Connection refused

La console WAPT ne parvient pas à joindre le port 443 du serveur.

- vérifier si le service **Nginx** est démarré sur le serveur :

```
ps aux | grep nginx
```

- si **Nginx** n'est pas lancé, relancer **Nginx** :

```
service nginx restart
```

- si **Nginx** ne veux toujours pas démarrer, il faut analyser les fichiers de journalisation dans `/var/log/nginx/` pour Linux ou dans `C:\Program Files (x86)\wapt\waptserverginx\logs` pour Windows.

Service Unavaible

Il est possible que le service *waptserver* soit stoppé.

- vérifier si le service **waptserver** est en cours d'éxécution :

```
ps aux | grep wapt
```

- si la commande ne retourne rien, lancer le **waptserver** en utilisant :

```
service waptserver start
```

Error connecting with SSL ... verify failed

La console ne semble pas réussir à vérifier le certificat HTTPS du serveur.

Attention: Attention, avant toute chose vérifiez que vous n'êtes pas victime d'une attaque *man in the middle* !

Note: Si vous venez de refaire votre serveur WAPT et que vous utilisez un certificat auto-signé, vous pouvez récupérer les anciennes clés de votre ancien serveur wapt dans `/opt/wapt/waptserver/apache/ssl`.

- fermer votre console WAPT ;
- supprimer le dossier `%appdata%\..\Local\waptconsole` ;
- lancer ensuite la commande `wapt-get enable-check-certificate` ;
- assurez-vous que la commande précédente s'est bien déroulée ;
- redémarrer le service WAPT avec `net stop waptservice && net start waptservice` ;
- relancer la console WAPT ;

Dans le cas ou vous ne pratiquez pas le *certificate pinning*, cela signifie que le certificat envoyé par le serveur ne pas être vérifié avec le bundle python **certifi**. Veillez à bien fournir la chaîne complète pour le certificat sur le serveur WAPT.

6.16.9 Problème lors du enable-check-certificate

J'ai le message « certificate CN ### sent by server does not match URL host ### lors du enable-check-certificate »

Cela signifie que le CN envoyé par le certificat du serveur ne correspond pas au *wapt_server* du fichier `wapt-get.ini`.

Deux solutions :

- vérifier le paramètre *wapt_server* dans votre fichier `wapt-get.ini` ;

  Si votre valeur est correcte, cela signifie sûrement qu'une erreur est survenue lors de la génération du certificat autosigné par le post conf, une faute de frappe ...

  Vous pouvez donc regénérer vos certificats autosignés.

- sur le serveur WAPT, supprimez le contenu du dossier `/opt/wapt/waptserver/apache/ssl/`.

  Ensuite, relancez le script de postconf (le même que pendant l'installation avec les mêmes arguments).

  Enfin, vérifiez bien le nom renseigné lors de l'étape *FQDN for the WAPT serveur* est correcte.

- vous pouvez maintenant retenter votre **`enable-check-certificate`**.

6.16.10 Problème avec la création de paquet

Création de paquet via la console

Le glisser-déposer dans le dépôt privé dans la console WAPT ne fonctionne pas :

- cela ne fonctionnera pas si la console n'a pas été lancée en tant qu"*Administrateur Local* ;
- cela ne fonctionnera pas si la console WAPT a été lancée avec UAC ;

  Solution de contournement simple : passer par *Outils* → *Créer un modèle de paquet depuis un installeur* → *Choisir un installeur*.

- la console ne renseigne pas automatiquement les informations dans les champs :
  - cela ne fonctionnera pas si vous avez un accent dans le chemin vers le fichier ;
  - l'installeur ne fournit peut être pas suffisamment d'informations sur ses propriétés ;

Problème de droit avec l'invite de commande Windows

Lors de l'édition d'un paquet, on obtient le message suivant :

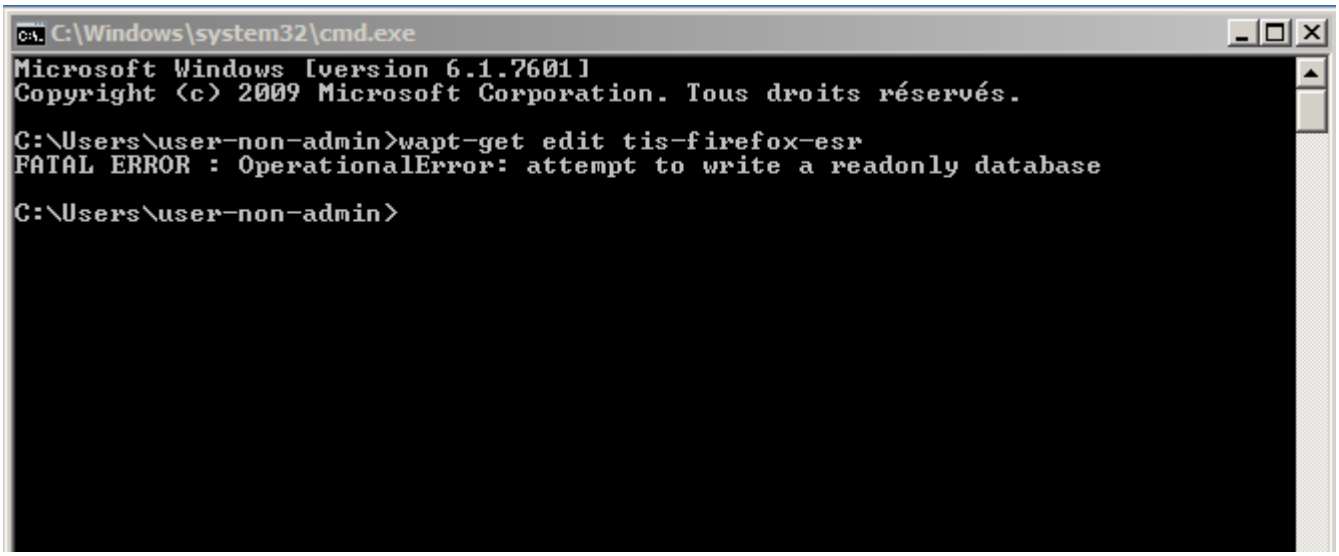


Figure170: OperationnalError: attempt to write a readlony database

Solution

Ouvrir une session en tant qu'*Administrateur Local* et refaire l'opération souhaitée.

Problème de droits avec PyScripter

Lorsqu'on souhaite tester l'installation d'un paquet sur son PC de développement de paquet à partir de **`PyScripter`**, on obtient le message :

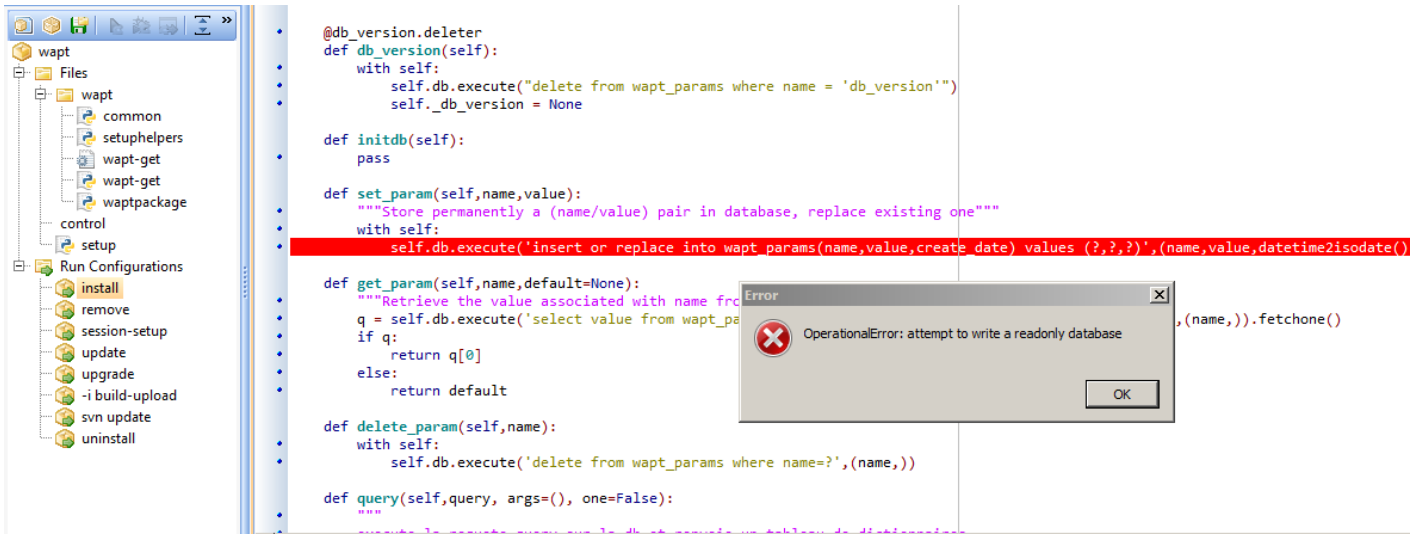


Figure171: OperationnalError: attempt to write a readlony database

Solution

Ouvrir une session en tant qu"*Administrateur Local* et refaire l'opération souhaitée.

Mon paquet WAPT est trop volumineux et je n'arrive pas à l'uploader

Quand un paquet est trop volumineux, il faut en général lancer le builder localement puis l'uploader avec **WinSCP**.

Solution

- *Builder le paquet dans PyScripter ou manuellement*.

> **Indication:** Si l"**upload** a précédemment échoué, vous pourrez trouver le paquet généré dans `C:\waptdev`.

- télécharger et installer WinSCP à l'aide de WAPT :

```
wapt-get install tis-winscp
```

- à l'aide de **WinSCP**, téléversez votre paquet dans le dossier `/var/www/html/wapt/` de votre serveur linux.
- une fois le transfert terminé, il faut regénérer le fichier `Packages` sur votre dépôt :

```
wapt-scanpackages /var/www/wapt/
```

6.16.11 Paquet WAPT en erreur

Problème d'installation

Symptôme

J'ai un paquet en erreur et le logiciel n'est pas installé sur la machine quand je me déplace.

Explication

Une erreur est survenue pendant l'exécution de l'installation définie dans `setup.py`.

Vous pouvez lire et analyser les messages d'erreur retournés dans la console et tenter de les comprendre.

L'installation sera retentée à chaque **upgrade** jusqu'à ce que le paquet ne génère plus d'erreur.

Solution

- si WAPT fournit un code d'erreur, chercher ce code d'erreur sur Internet ;

  Exemple with a MSI : *1618* : Une autre installation est déjà en cours. Un redémarrage devrait solutionner le problème.

  Note: Les différents codes d'erreur MSI sont disponibles ici.

- se déplacer physiquement sur la machine en erreur et relancer l'installation silencieuse en ligne de commande. Vérifier ensuite que le logiciel à bien été installé;

  Attention: Une fois l'installation silencieuse lancée, ne pas intervenir.

  L'objectif est de reproduire le comportement de l'agent WAPT.

- si l'installation fonctionne en mode silencieux, en contexte utilisateur, cela peut signifier que l'installeur ne supporte pas l'installation en compte *SYSTEM* ;
- si cela ne fonctionne toujours pas, lancer l'installation manuellement. Il est possible qu'une erreur apparaisse indiquant explicitement le problème. Exemple (Dépendance manquante: .Net , Java, etc..) ;
- il est possible que l'installeur ne supporte pas l'écrasement d'une installation précédente, alors prévoir la désinstallation des anciennes version avant d'installer la nouvelle version ;

Erreur « timed out after seconds with output "600.0" »

Symptôme

Certains paquets dans la console retournent l'erreur :

```
"Erreur timed out after seconds with output '600.0'"
```

Explication

Par défaut lors d'une installation (avec **`run`**, **`install_msi_if_needed`** ou **`install_exe_if_needed`**) WAPT va attendre 600 secondes que l'installeur lui rende la main.

si l'installeur n'a pas terminé dans ce délai, WAPT coupera l'installation en cours.

Résolution: Installations volumineuses

Si vous tentez d'installer un gros logiciel (Office, Solidworks, Libreoffice . . .), il est possible que l'intervalle de 600 secondes ne soit pas suffisant.

Vous devez alors, augmentez cette valeur avec l'argument *timeout*, ex: *timeout* = 1200 :

```
run('"setup.exe" /adminfile office2010noreboot.MSP',timeout=1200)
```

Erreur « has been installed but the uninstall key can not be found »

Symptôme

Certains paquets dans la console retournent l'erreur :

```
XXX has been installed but the uninstall key can not be found.
```

Explication

WAPT s'appuie sur Windows pour installer les binaires *.msi* avec **install_msi_if_need** et les binaires *.exe* avec **install_exe_if_need**.

Par défaut, WAPT accepte les codes de retour : *0* (OK) et *3010* (redémarrage nécessaire), et il vérifie la clé de désinstallation résultante (*uninstall key*).

Malheureusement, on ne peut pas toujours se fier à ces codes d'erreur, WAPT vérifie enfin que tout s'est bien déroulé :

- il vérifie la présence de la clé de désinstallation sur la machine ;
- il vérifie que la version est bien égale ou supérieure à celle renseignée ;
- si ce n'est pas le cas, il en déduit que le logiciel n'est peut-être pas présent sur la machine ;

La fonction bascule alors volontairement le paquet en erreur. L'installation sera retentée lors de chaque upgrade, jusqu'à ce que le paquet ne génère plus d'erreur.

Solution

Attention: Avant toute chose, il convient de se connecter sur la machine en erreur et de vérifier manuellement **si le logiciel est correctement installé** . Si ce n'est pas le cas, se référer à la documentation sur les *problèmes d'installation d'un paquet*..

- si le logiciel est bien installé, cela signifie peut être que la clé de désinstallation ou la version fournie dans le paquet n'est pas bonne ;
- récupérer la bonne clé de désinstallation et corriger le paquet en conséquence ;
- si l'erreur se produit avec **install_msi_if_needed** cela signifie que l'installeur MSI est mal conçu et renvoie une mauvaise clé de désinstallation ;

Erreur « has been installed and the uninstall key found but version is not good »

Symptôme

Certains paquets dans la console retournent l'erreur :

```
has been installed and the *uninstall key* found but version is not good
```

Explication

Avec les commandes **install_msi_if_needed** et **install_exe_if_needed**, des vérifications supplémentaires sont effectuées pour vérifier que tout s'est bien passé.

Solution

Attention: Avant toute chose, il convient de se connecter sur la machine en erreur et de vérifier manuellement **si le logiciel est correctement installé** . Si ce n'est pas le cas, se référer à la documentation sur les *problèmes d'installation d'un paquet*..

Solution: Avec `install_msi_if_needed`

Les informations étant extraites depuis l'installeur MSI, cela signifie que le fichiers MSI ne renvoie pas la bonne version ou que la clé de désinstallation retournée n'est pas la bonne version.

Vérifier avec la commande :

```
wapt-get list-registry
```

Si la clé retournée n'est pas celle renseignée dans la partie installation du fichier `setup.py` , il n'est pas possible d'utiliser la fonction **install_msi_if_needed**.

Il faut rebasculer l'installation avec un simple **run()** et gérer les exceptions manuellement.

Avec install_exe_if_needed

Cela signifie probablement que la version renseignée dans la fonction **install_exe_if_needed** n'est pas la bonne. Corriger le paquet WAPT en conséquence.

Note: Si l'argument `min_version` n'a pas été renseigné, WAPT va tenter de récupérer automatiquement la version depuis l'installeur exe.

Pour vérifier la clé de désinstallation utilisée et le numéro de version, utiliser la commande :

```
wapt-get list-registry
```

Si aucune version n'est fournie avec la commande **list-registry**, cela signifie que la clé de désinstallation du logiciel ne fournit pas de version.

Deux solutions :

- utiliser l'argument `get_version` pour fournir un chemin vers une autre `uninstallkey` ;

```
def install():

   def versnaps2(key):
       return key['name'].replace('NAPS2 ','')

   install_exe_if_needed('naps2-5.3.3-setup.exe',silentflags='/VERYSILENT',key='NAPS2 (Not␣
↪Another PDF Scanner 2)_is1',get_version=versnaps2)
```

(suite sur la page suivante)

(suite de la page précédente)

- fournir une valeur vide en argument pour `min_version` afin d'indiquer à WAPT qu'aucune version n'est à vérifier ;

```
min_version=' '
```

Attention: Avec cette méthode **on ne vérifiera plus la version lors de mise à jour !**

6.16.12 Problèmes fréquents liés aux Antivirus

Certains Antivirus lèvent des alertes pour des composants de WAPT.

Parmis ceux-ci le composant **`nssm.exe`** est utilisé par WAPT comme utilitaire de service pour l'agent WAPT.

Voici une liste des exceptions possibles à déclarer dans votre interface de gestion centralisé antivirus :

```
"C:\Program Files (x86)\wapt\waptservice\win32\nssm.exe"
"C:\Program Files (x86)\wapt\waptservice\win64\nssm.exe"
"C:\Program Files (x86)\wapt\waptagent.exe"
"C:\Program Files (x86)\wapt\waptconsole.exe"
"C:\Program Files (x86)\wapt\waptexit.exe"
"C:\wapt\waptservice\win32\nssm.exe"
"C:\wapt\waptservice\win64\nssm.exe"
"C:\wapt\waptagent.exe"
"C:\wapt\waptconsole.exe"
"C:\wapt\waptexit.exe"
"C:\Windows\Temp\waptdeploy.exe"
"C:\Windows\Temp\waptagent.exe"
"C:\Windows\Temp\is-?????.tmp\waptagent.tmp"
```

6.16.13 EWaptBadControl: "utf8" codec can't decode byte

Si vous recevez ce message, cela peut signifier que vous n'avez pas mis en place correctement votre environnement de développement. Visitez cette *section de la documentation sur la configuration de l'UTF-8 (pas de BOM)*.

6.16.14 I have a lot more hosts in the console than I have host packages on my server?

Suite à une remarque de Philippe LEMAIRE du Lycée Français Alexandre Yersin à Hanoï, si vous utilisez la version Entreprise du WAPT et que vous faites un usage intensif des paquets *unit* ou *profile packages*, vous pouvez réaliser que vous aurez beaucoup plus d'hôtes dans votre console que de **host packages** sur votre serveur WAPT. **C'est normal**.

En fait, les paquets *unit* et les paquets *profile* ne sont pas explicitement attribués à la machine (c'est-à-dire comme des dépendances dans le paquet *host*) mais sont implicitement pris en compte par le moteur de dépendance de l'agent WAPT lors de la mise à niveau WAPT.

On peut donc ne pas avoir de paquet *host* sur le serveur si seuls des paquets *unit* sont utilisés pour gérer une flotte d'appareils.

6.17 Erreurs courantes

6.17.1 Utiliser un lecteur réseau pour stocker et livrer des paquets WAPT

Le mode de fonctionnement standard de WAPT est avec un serveur Web sécurisé qui fournit les paquets WAPT aux clients WAPT.

Tranquil IT déconseille l'utilisation d'un lecteur réseau pour la livraison de paquets WAPT pour plusieurs raisons :

- un serveur web est extrêmement facile à installer, à sécuriser, à maintenir, à sauvegarder et à surveiller ;
- pour fonctionner correctement, un paquet WAPT doit être autonome. En effet, nous ne savons pas si le réseau sera disponible au moment du lancement de l'installation (par exemple si nous avons un waptexit qui démarre lorsque la station de travail s'arrête sur un réseau avec une authentification utilisateur 802.1x, il n'y aura plus de réseau disponible au moment de l'installation). La nature autonome du WAPT le rend plus déterministe que les autres solutions de déploiement ;
- une congestion du réseau peut résulter du téléchargement de gros paquets sur de grandes flottes d'appareils parce que vous avez moins de contrôle sur la consommation de bande passante ou bien vous ne pouvez pas terminer un téléchargement partiel ;
- cette méthode casse ou au moins affaiblit le cadre de sécurité du WAPT ;
- cette méthode ne vous permet pas d'exposer vos dépôts sur Internet pour votre personnel itinérant ;

Attention: Même si WAPT *peut fonctionner* indépendamment du mode de transport, **Tranquil IT ne supportera pas officiellement l'utilisation d'un lecteur réseau pour stocker et livrer des paquets WAPT**.

6.17.2 Utiliser la fonction register() dans vos scripts d'audit

La fonction register() force l'envoi au serveur WAPT de l'inventaire matériel et logiciel de l'agent WAPT.

Cette fonction est très éprouvante pour les performances du serveur car elle oblige le serveur à analyser un JSON (Java Script Object Notation) BLOB relativement grand et à injecter le résultat dans la base de données PostgreSQL.

La fonction est par défaut déclenchée manuellement ou lorsqu'une nouvelle mise à niveau de paquet est appliquée.

Lorsque vous utilisez la fonction register() dans un script d'audit, elle sera exécutée à chaque fois que le script d'audit est déclenché et chargera le serveur sans bénéfice apparent.

Par conséquent, **nous ne recommandons pas l'utilisation de la fonction register() dans les scripts d'audit**.

6.18 Glossaire

Administrateur

Administrateurs

Développeur de Paquets

Développeurs de Paquets Un **Administrateur** est un individu pouvant signer des paquets, qu'ils intègrent ou non du code python et les charger sur le dépôt principal.

Administrateur Local

Administrateurs Locaux Un **Administrateur Local** est un utilisateur disposant des droits d'administration locaux sur les postes équipés de WAPT.

Déployeur de Paquets

Déployeurs de Paquets Un **Gestionnaire de Déploiement** est un individu pouvant signer des paquets ne contenant pas de code python (en général les paquets de type *group*, *unit* et *host*) et les charger sur le dépôt principal. Il est typiquement un membre d'une équipe informatique locale qui une bonne connaissance des besoins des utilisateurs.

SuperAdmin Le **SuperAdmin** est l'*Utilisateur* dont l'identifiant et le mot de passe est défini lors de la post-configuration du serveur WAPT. Dans la version WAPT Community, il est l'unique *Administrateur* de WAPT.

Utilisateur

Utilisateurs Un **Utilisateur** est un individu qui utilise une machine équipée de l'agent WAPT (WAPT Enterprise et Community).

Organisation

Organisations L'**Organisation** correspond au périmètre de responsabilité dans lequel est exploitée la solution WAPT.

ANSSI **Agence Nationale de la Sécurité des Systèmes d'Information** est un service français en charge d'assurer la sécurité des informations sensibles de l'État Français et d'une mission de conseil et de soutien aux administrations et aux opérateurs d'importance vitale.

DNS **Domain Name System** est un service permettant de traduire un nom de domaine en informations de plusieurs types qui y sont associées, notamment en adresses IP de la machine portant ce nom.

FQDN **Fully Qualified Domain Name** est un nom de domaine complètement qualifié. C'est la notation complète d'un nom de domaine qui révèle la position absolue de la machine dans l'arborescence DNS en indiquant tous les niveaux supérieurs jusqu'à la racine. Exemple de FQDN : wapt.nantes.pdl.organisation.fr.

EPEL **Extra Packages for Enterprise Linux** est un dépôt additionel pour CentOS et RedHat.

GPO Les **Group Policy Objects** ou **Objets de Stratégie de Groupe** sont des objets définissant des stratégies de sécurité dans un environnement Windows. Les stratégies peuvent être définies localement à l'aide de `gpedit.msc` ou définies globalement en domaine Active Directory.

IDE **Integrated Development Environment** (environnement de développement intégré) est un ensemble d'outils qui augmente la productivité des développeurs logiciels. Un IDE permet notamment de déboguer ligne par ligne le code source d'un programme, d'éditer, compiler et exécuter dans une seule interface.

MMC **Microsoft Management Console** est un gestionnaire de console virtuelle incorporé dans Microsoft Windows, qui sert de conteneur pour des interfaces graphiques de configuration.

NAT **Network Address Translation** est un mécanisme qui permet à des machines disposant d'adresses qui font partie d'un intranet de communiquer avec le reste d'Internet en semblant utiliser des adresses externes uniques et routables, au travers d'un routeur.

Setuphelpers **Setuphelpers** est une librairie Python écrite spécialement pour WAPT. Elle contient un ensemble de fonctions et de variables utiles au développement de paquets, pour la manipulation de fichiers, création de raccourcis, etc.

SRV Le champ **SRV** permet de définir un serveur spécifique pour une application, notamment pour la répartition de charge.

virtualhost En informatique, l'**hébergement virtuel** (de l'anglais virtual hosting abrégé **vhost**) est une méthode que les serveurs tels que les serveurs Web utilisent pour accueillir plus d'un nom de domaine sur le même ordinateur, parfois sur la même adresse IP, tout en maintenant une gestion séparée de chacun de ces noms. Cela permet de partager les ressources du serveur, comme la mémoire et le processeur, sans nécessiter que tous les services fournis utilisent le même nom d'hôte.

Websocket **Websocket** est une couche applicative Web bidirectionnelle permettant la communication client-serveur en utilisant une connexion TCP.

UUID Un **UUID** est un identifiant normalisé et réputé unique ; ainsi un UUID dans le contexte de WAPT permet d'identifier de manière unique une machine. Pour en savoir plus, suivre https://en.wikipedia.org/wiki/Universally_unique_identifier.

champ CNAME Un enregistrement **CNAME** ou enregistrement de nom canonique est un type d'enregistrement-ressource dans le Domain Name System (*DNS*) qui spécifie que le nom de domaine est un alias d'un autre nom de domaine canonique.

champ A Un **champ A** met en relation un nom (en général le nom physique d'un serveur) avec une IP.

Autorité de Certification Une CA (Autorité de Certification) est un tiers de confiance permettant d'authentifier l'identité des correspondants.

PKI **Public Key Infrastructure**, ou Infrastructure à Clés Publiques est un ensemble de composants physiques, de procédures humaines et de logiciels destiné à gérer les clés publiques des utilisateurs d'un système.

6.19 Histoire de WAPT

6.19.1 WAPT 0.8 Community (novembre 2013)

- nettoyage et stabilisation du code de la console ;
- passage du waptservice local sur le poste de Lazarus à Python ;
- gestion de l'ouverture de port sur le firewall windows pour l'accès au **`waptservice`** depuis le **`waptserveur`** ;
- affichage de l'état de mise à jour des paquets wapt sur les postes dans la console ;
- possibilité de récupérer un paquet depuis le dépôt TIS et de l'intégrer au dépôt local ;
- embryon du système de build ;
- après la création du **`waptsetup`** customisé dans la **`waptconsole`**, possibilité de l'uploader directement depuis la **`waptconsole`** ;

6.19.2 WAPT 0.9 Community (septembre 2014)

- packaging du waptserver sous Windows ;
- création d'un outil de postconfiguration sous Windows pour faciliter l'installation ;
- passage en https pour toutes les connexions ;
- possibilité de demander la suppression d'un paquet sur une machine ;
- améliorer les retours d'erreur ;
- affichage de la description du paquet lorsqu'on le sélectionne dans les listes dans waptconsole ;
- identification de la machine à travers le compte Machine grâce à kerberos (pas configuré par défaut) ;
- gestion du groupe de waptselfservice pour déléguer des droits d'installation ;
- possibilité de pousser une installation waptsetup sur un poste à travers les MS-RPC (si le firewall ne le bloque pas) ;

6.19.3 WAPT 1.0 Community (février 2015)

La release de cette milestone a été annoncée lors de la conférence FOSDEM à Bruxelles le dimanche 1er février 2015.

- internationalisation de la console, du service, du serveur et des messages de notification ;
- mise en place de documenation scénari ;
- rajout de l'icône de l'application dans le paquet wapt pour un affichage plus joli sur le site web wapt.tranquil.it ;
- installeur Windows plus précautionneux pour éviter des conflits lors de l'installation (vérification des ouvertures de ports, etc.) ;
- correction extensive de bug obscures lié à des spécificités d'installation et de réseau;
- mise en place d'un système de build automatique buildbot ;

6.19.4 WAPT 1.1 Community (février 2015)

- correction de plusieurs anomalies gênantes portées à notre connaissance par des utilisateurs de la 1.0.0 ;
- affichage de l'état joignable d'une machine (machine éteinte / allumée) ;

6.19.5 WAPT 1.3 et 1.5 Community (2016-2017)

- intégration de l'authentification du service local dans le domaine Microsoft Active Directory ou Samba4-AD ;
- ajout d'un wizard de création de paquets intégrés à la console WAPT ;
- différenciation des rôles entre les *Développeurs de Paquets* et les *Déployeurs de Paquets* ;
- passage du serveur à PostgreSQL avec les extensions JSON en remplacement de MongoDB ;
- passage en Websocket ;
- option d'identification de la machine à travers un secret partagé en mode fallback pour les postes non intégrés au domaine, ou bien ne pouvant accéder au serveur Active Directory ou Samba-AD ;

6.19.6 WAPT 1.5 Enterprise (début 2018)

Les fonctionnalités décrites dans cette section concernent la version **Enterprise** de WAPT.

- gestion par Unité organisationelle (OU Machine) ;
- prise en compte de certificat d'autorité pour la signature de paquets, en plus des certificats individuels ;
- authentification Kerberos de l'*Administrateur* sur la console WAPT ;

6.19.7 WAPT 1.6 (Août 2018)

- fonction d'audit périodique de la conformité du poste / référentiel des paquets (**Enterprise**) ;
- (tech preview) gestion des mises à jour OS dans WAPT reproduisant le fonctionnement d'un WSUS (**Enterprise**) ;
- authentification du client WAPT par certificat pour l'accès aux dépôts de paquets, et pour les connexions vers le serveur (inventaire, websockets) ;

6.19.8 WAPT 1.7

- reporting personnalisable intégré à console WAPT (**Enterprise**) ;
- différentiation entre les paquets accessibles en selfservice utilisateur et les paquets disponibles uniquement pour déploiement par les *Administrateurs* (**Enterprise**) ;
- mises à jour globales différenciées suivant la criticité des paquets (**Enterprise**) ;
  - immédiate pour les mises à jour critiques ;
  - avec acceptation de l'utilisateur pour les autres si les process impactés sont lancés ;
- gestion par Unités Organisationnelles (paquets *unit*) (**Enterprise**) ;

6.19.9 WAPT 1.8

- agent WAPT pour Linux Debian, Linux CentOS, Ubuntu et Apple MacOS ;
- réplication de paquets WAPT entre dépôts intégrée ;
- règles de sélection des dépôts intégrées ;

6.20 Appliquer les meilleures pratiques au packaging de logiciels

Note: _benwa est un administrateur système et il a autorisé Tranquil IT à republier son excellente diatribe sur reddit Developers, you can make sysadmins happier.

6.20.1 Variables d'environnement

- Les variables d'environnement existent depuis le DOS. Elles peuvent vous faciliter la vie (et la mienne).

6.20.2 Répertoires des programmes

- Tous les systèmes n'utilisent pas « C: » comme lecteur principal. Certaines entreprises utilisent la redirection de dossiers, et déplacent le dossier Documents. Certains endroits dans le monde ne parlent pas anglais et leurs noms de répertoires reflètent cela. **Utilisez ces variables d'environnement pour que vos programmes « fonctionnent tout simplement «** :
  - `%SystemDrive%` est le lecteur où se trouve `%SystemRoot%`. Vous n'avez probablement pas besoin de le savoir ;
  - Le système d'exploitation Windows est situé dans le répertoire `%SystemRoot%`. Ne vous en souciez pas. Laissez le répertoire Windows tranquille ;
  - `%ProgramFiles%` est l'endroit où vous devez placer vos fichiers de programme, de préférence dans une structure `Company\Program` ;
  - `%ProgramFiles(x86)%` est l'endroit où vous devez placer vos fichiers de programme 32 bits. Veuillez les mettre à jour pour le 64 bits. Le 32-bit ne sera plus supporté dans l'avenir, et les entreprises attendront que vous vous organisiez pour bien plus longtemps que nécessaire ;
  - `ProgramData%` est l'endroit où vous devez stocker les données qui ne sont pas spécifiques à l'utilisateur, mais qui doivent quand même être écrites par les utilisateurs (les utilisateurs n'ont pas non plus d'accès en écriture à ce dossier).

    Votre programme ne devrait pas nécessiter de droits d'administrateur pour s'exécuter, car vous ne devriez pas nous faire écrire dans le répertoire `%ProgramFiles%`. Aussi, ne mettez pas d'exécutables dans ce répertoire.
  - `%Temp%` est l'endroit où vous pouvez traiter des données temporaires. Placez ces données dans un nom de dossier unique (peut-être un GUID généré) afin de ne pas provoquer d'incompatibilité avec un autre programme. Windows fera même le nettoyage à votre place. Ne placez pas de données temporaires dans les dossiers `%ProgramData%` ou `%ProgramFiles%` ;
  - `%AppData%` vous permet de sauvegarder les paramètres de l'utilisateur qui exécute votre programme. C'est un endroit fantastique qui peut être synchronisé avec un serveur et être utilisé pour migrer rapidement et facilement un utilisateur vers une nouvelle machine et conserver tous les paramètres de ses programmes. Ne mettez pas de fichiers géants ou éphémères ici.

    Vous pourriez être à l'origine d'une connexion très lente si vous mettez les mauvais éléments ici et qu'une machine doit les synchroniser. **NE METTEZ PAS VOS FICHIERS DE PROGRAMMES ICI**. C'est l'entreprise qui décide quels logiciels sont autorisés à fonctionner, ce n'est pas à vous de décider, ni aux utilisateurs qui ne savent peut-être pas comment l'environnement de leur entreprise est configuré ;
  - `LocalAppData%` permet de placer des fichiers plus volumineux spécifiques à un utilisateur ou à un ordinateur. Par exemple, personne n'a besoin de synchroniser un cache de vignettes. Elles ne seront pas transférées lorsqu'un utilisateur migrera vers une nouvelle machine, ou se connectera à une nouvelle station VDI, ou à un nouveau serveur de terminal. **NE METTEZ PAS VOS FICHIERS DE PROGRAMME ICI NON PLUS** ;

Note: De plus en plus d'éditeurs de logiciels proposent des versions *portables* de leurs logiciels qui s'installent et s'exécutent à partir de `%AppData%` ou de `%LocalAppData%`. Votre objectif est de permettre aux utilisateurs d'installer des logiciels même s'ils ne sont pas Administrateurs Locaux et vous commercialisez cela comme une fonctionnalité, bien qu'il s'agisse plutôt d'un NOGO de sécurité. Pire encore, vous avez tendance à rendre difficile de trouver le bon *MSI* qui permettrait à vos clients d'installer correctement votre logiciel dans `%ProgramFiles%`. Faites en sorte qu'il soit facile de trouver votre *MSI* qui s'installera dans les `%ProgramFiles%`, de cette façon vous ferez en sorte que les politiques de restriction des logiciels et de verrouillage des applications de vos clients fonctionnent bien et que leurs administrateurs système soient satisfaits.

Vous pouvez aussi bien obtenir ces chemins de répertoires par des appels API si vous n'utilisez pas ou ne pouvez pas utiliser de variables d'environnement.

6.20.3 Logs

- Utilisez le Journal des événements Windows pour la journalisation. Il s'occupera de la rotation pour vous et un administrateur système peut transmettre ces journaux ou faire ce qu'il faut. Vous pouvez même créer votre propre petite zone juste pour votre programme.

6.20.4 Codes d'erreur

- Utilisez les codes d'erreur standard lorsque vous quittez votre programme.

6.20.5 Impression

- Utilisez l'"API d'impression Windows et n'utilisez pas l'impression directe dans votre programme.

6.20.6 Distribution

- Distribuez votre programme en MSI. C'est le standard pour les fichiers d'installation de Windows (même si Microsoft ne l'utilise pas toujours lui-même).
- Signez vos fichiers d'installation et vos exécutables. C'est ainsi que nous savons que votre MSI est valide et que nous pouvons le mettre sur une liste blanche dans AppLocker ou équivalent.

Note: Applocker et Software Restriction Policies peuvent être très efficaces et la **gestion de ces stratégies peut être rendue plus simple avec WAPT**.

6.20.7 Mises à jour

- Vous souhaitez que votre programme se mette à jour ? C'est possible si l'entreprise est d'accord. Vous pouvez créer une tâche ou un service programmé qui s'exécute en mode élevé pour permettre cela sans accorder de droits d'administrateur à l'utilisateur. J'aime la façon dont Chrome Enterprise le fait : il donne une GPO pour définir les paramètres de mise à jour, la version maximale à laquelle elle va se mettre à jour (disons 81.* pour permettre toutes les mises à jour mineures automatiquement et les versions majeures sont manuelles), et un service. Ils ont également une GPO pour empêcher les installations lancées par les utilisateurs ;

Note: WAPT est conçu pour les entreprises qui ne permettent pas à leurs utilisateurs d'exécuter des mises à jour logicielles, c'est la politique souvent choisie par les grandes entreprises consciencieuses vis à vis de la sécurité.

6.20.8 Numéros de version

- Utilisez le versionnage semantique (doit aller dans la propriété de version dans le fichier d'installation et dans la liste Ajout/Suppression de programmes, pas dans le titre de l'application) et ayez un changelog. Vous pouvez également mettre à disposition en téléchargement votre installateur à un endroit prévisible pour permettre l'automatisation. Un chemin de mise à jour publié est également utile ;

Note: Si vous appliquez cette pratique, alors vous rendrez les administrateurs système qui déploient vos mises à jour logicielles en utilisant la fonction *WAPT def_update()* **très heureux** !

6.20.9 GPO

- Les modèles ADMX sont des trucs très moches ;

Note: Nous sommes tout à fait d'accord avec vous _benwa sur ce point chez Tranquil IT. Si les développeurs conseillent à leurs clients d'utiliser des GPO pour déployer leur logiciel ou leur système ou les paramètres des utilisateurs, alors, **ils doivent apprendre que les GPO ne sont pas fiables**.

Au lieu de cela, packagez vos logiciels, votre système et vos configurations utilisateur en utilisant WAPT. Un fichier `setup.py` est beaucoup plus facile qu'un fichier *xml* pour les administrateurs système qui doivent le vérifier avant de le déployer.

Les paquets WAPT peuvent être appliqués récursivement à des arbres d'Unités Organisationnelles, de sorte que votre paquet WAPT se comportera en production exactement comme le ferait une GPO, **juste beaucoup plus facilement**.

6.20.10 Dongles de licences

- Les dongles de licence USB sont un péché. Utilisez une licence de logiciel ordinaire ou une licence activée par réseau. Je suis sûr qu'il y plein de systèmes de gestion des licences sur le marché pour que vous n'ayez pas à réinventer la roue ;

Note: Vous pouvez faire en sorte que votre logiciel accepte une clé de licence comme paramètre dans votre exécutable *msi*.

WAPT peut être utilisé pour attribuer des clés de licence à des postes de travail individuels lors de l'installation en utilisant une méthode *qui garantit que la clé de licence ne peut pas être lue pendant le transport*.

Ensuite, si vous voulez que votre logiciel appelle chez vous pour vérifier la validité de la licence, faites en sorte que votre méthode fonctionne avec des *serveur mandataires*.

6.20.11 Fonctionnement en réseau

- N'utilisez pas ce fichu champ de saisie IPv4 personnalisé. Utilisez des FDQN. L'IPv6 existe depuis 1998 et fonctionnera avec votre logiciel si vous lui donnez une chance ;
- Le pare-feu Windows (je ne peux pas vraiment en dire plus sur les pare-feu tiers) va rester en place. Connaissez la différence entre une règle entrante et une règle sortante. Il est probable que votre serveur aura besoin d'une règle entrante. Vos clients n'auront probablement même pas besoin d'une règle sortante. Configurez-les au moment de l'installation, et non au moment du lancement. Utilisez les groupes de pare-feu pour faciliter le filtrage. N'utilisez aucune règle si vous pouvez. Le but n'est

pas de faire fonctionner le système, mais de le faire fonctionner en toute sécurité. Si vous n'utilisez pas les numéros de version dans vos chemins d'installation, vous n'aurez peut-être même pas à refaire ces règles après chaque mise à jour ;

- Les serveurs mandataires sont bons pour l'hygiène et les serveurs mandataires sont maintenant une caractéristique de sécurité par défaut non seulement dans les environnements informatiques des entreprises, mais aussi sur les petits réseaux. En rendant votre logiciel non compatible avec les proxies, les administrateurs réseau de vos clients devront établir et maintenir des règles spéciales pour leurs pares-feu, et cela rien que pour vos beaux yeux. Il est facile de coder votre logiciel pour qu'il fonctionne avec des serveurs mandataires, alors faites-le !

6.20.12 PDFs

- Ne livrez pas un logiciel qui nécessite d'autoriser le fonctionnement de javascript dans les lecteurs de PDF. La logique métier doit être exécutée avant la sortie au format PDF, pas après.

Note: Le *PDF* est le format de fichier que les gens utilisent par défaut pour échanger des documents. Les lecteurs PDF sont destinés à afficher des documents, et non à exécuter des programmes non signés.

6.21 Stratégie de sortie des mises à jour de WAPT

Les mises à jour WAPT ne sortent pas selon un calendrier fixe.

Au lieu de cela, Tranquil IT sortira une nouvelle version majeure de WAPT lorsque de nouvelles mises à jour fonctionnelles majeures seront intégrées au cœur du produit.

Tranquil IT publiera des versions mineures intermédiaires de WAPT entre les versions majeures afin de corriger les défauts de fonctionnement et de sécurité.

6.22 Délai de publication entre les versions Enterprise et Community

Une nouvelle version majeure sera publiée sous le nom de **Community** et cette même version sera publiée sous le nom de **Enterprise** RC1 (Release Candidate #1). Avant sa sortie, la version Community aura subi des tests internes approfondis pour s'assurer qu'aucune régression ne s'est glissée dans le cœur de WAPT.

Les utilisateurs de la version Community seront encouragés à adopter la nouvelle version majeure.

La version Enterprise passera par plusieurs CR et la version Enterprise finale sera alors disponible entre 4 et 8 semaines après la sortie de la version Community.

Ce délai apportera plusieurs bénéfice au processus de sortie de la version Enterprise :

- accorder plus de temps pour effectuer des tests approfondis des nouvelles fonctionnalités Enterprise tout en évitant les régressions ;
- permettre à Tranquil IT de travailler avec un petit groupe de clients Enterprise sélectionnés afin d'assurer le bon déroulement des procédures de mise à niveau
- donner un peu de temps au forum et à la liste de diffusion pour indexer les questions et les réponses qui seront éventuellement incluses dans la documentation officielle ;
- permettre à l'équipe de documentation de s'appuyer sur une base fonctionnelle figée pour ainsi documenter de manière fiable les fonctionnalités nouvelles ou améliorées ;

- donner à l'équipe de traduction le délai nécessaire pour mettre à jour les traductions ;
- permettre à l'équipe de communication et marketing de s'appuyer sur une base fonctionnelle figée pour ainsi rétro-planifier les annonces, les podcasts vidéo et la promotion générale de WAPT ;

6.23 Contribuer à WAPT

Le projet est maintenu sur github https://github.com/tranquilit/WAPT.

À faire: work still in progress

6.23.1 Recompiler WAPT à partir des sources

Construire l'agent WAPT pour Windows

Exigences liées à WAPT

Environment Python

- Python 2.7.13 ;
- client python libraries in `requirements.txt` ;
- server python libraries in `requirements-server.txt` ;

Environment Lazarus

WAPT utilise les bibliothèques tierces freepascal / lazarus suivantes :

- pl_indy : https://www.pilotlogic.com/sitejoom/index.php/115-wiki/ct-packages/networking/271-pl-indy ;
- pltis_superobject : https://github.com/hgourvest/superobject;
- virtualtrees: https://www.pilotlogic.com/sitejoom/index.php/85-wiki/codetyphon-studio/ct-packages/301-pl-virtualtrees and https://github.com/blikblum/VirtualTreeView-Lazarus;
- pltis_python4delphi : https://github.com/pyscripter/pltis_python4delphi ;
- delphizmq : https://github.com/bvarga/delphizmq ;
- JCL : https://wiki.delphi-jedi.org/wiki/JCL_Installation ;
- thmtlport : https://svn.code.sf.net/p/lazarus-ccr/svn/components/thtmlport ;

Packages TIS

- pltis_python4delphi : https://github.com/tranquilit/pltis_python4delphi ;
- pltis_utils : https://github.com/tranquilit/pltis_utils : sous-ensemble de librairies du projet JEDI JCL adapté à lazarus ;
- pltis_sogrid : https://github.com/tranquilit/pltis_sogrid ;
- pltis_superobject : https://github.com/tranquilit/pltis_superobject ;

Créer un environment de développement virtualenv

Pour une installation propre de zéro sur Windows :

- install python2.7.15 from https://www.python.org/ftp/python/2.7.15/python-2.7.15.msi;
- mettre à jour **`python-setuptools`** :

```
c:\python27\python -m pip install -U pip setuptools
```

- créer l'environment de développement `virtualenv` ;

```
mkdir c:\tranquilit
git clone git@github.com:tranquilit/WAPT.git (ou git clean -fxd ...)
cd c:\tranquilit\wapt init_workdir.bat
```

Créer un environment de développement avec virtualenv

En tant qu'un *Administrateur Local* sur une installation fraîche de Windows 7 :

- installer l'agent WAPT à partir de https://srvwapt.mydomain.lan/wapt/waptagent.exe;
- désactiver les UAC ;
- afficher les fichiers cachés et les extensions de fichiers ;
- augmenter la largeur des fenêtres du CMD et passer en mode édition rapide ;

Installer Lazarus

```
wapt-get install tis-pyscripter tis-tortoisegit tis-7zip tis-python27 tis-notepadplusplus tis-
↪firefox tis-putty tis-lazarus tis-openssh tis-signtool

wget https://www.sqlite.org/2018/sqlite-dll-win32-x86-3250200.zip
unzip sqlite3.dll dans C:\Windows\SysWOW64
md c:\tranquilit

REM git.exe clone --recurse-submodules "https://github.com/tranquilit/WAPT.git" "C:\tranquilit\
↪wapt"
git.exe clone --recurse-submodules "https://github.com/tranquilit/WAPT.git" "C:\tranquilit\wapt"
REM git pull --recurse-submodules=yes --ff-only)
cd  \tranquilit\wapt
init_workdir.bat
```

(suite sur la page suivante)

(suite de la page précédente)

```
git clone https://github.com/tranquilit/pltis_indy.git c:\tranquilit\pltis_indy
git clone https://github.com/tranquilit/pltis_utils.git c:\tranquilit\pltis_utils
git clone https://github.com/tranquilit/pltis_sogrid.git  c:\tranquilit\pltis_sogrid
git clone https://github.com/tranquilit/pltis_superobject.git  c:\tranquilit\pltis_superobject
git clone https://github.com/tranquilit/pltis_python4delphi.git c:\tranquilit\pltis_python4delphi
git clone https://github.com/tranquilit/pltis_virtualtrees.git c:\tranquilit\pltis_virtualtrees
git clone https://github.com/tranquilit/pltis_virtualtreesextra.git c:\tranquilit\pltis_
↪virtualtreesextra
git clone https://github.com/tranquilit/pltis_dcpcrypt.git c:\tranquilit\pltis_dcpcrypt
git clone https://github.com/tranquilit/pltis_luipack.git c:\tranquilit\pltis_luipack
git clone https://github.com/tranquilit/pltis_synapse.git c:\tranquilit\pltis_synapse

c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_dcpcrypt\dcpcrypt_laz.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_indy\indylaz.lpk
c:\lazarus\lazbuild.exe c:\tranquilit\pltis_utils\pltis_utils.lpk
c:\lazarus\lazbuild.exe c:\tranquilit\pltis_superobject\pltis_superobject.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_virtualtrees\pltis_virtualtrees.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_virtualtreesextra\pltis_
↪virtualtreesextra.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_sogrid\pltis_sogrid.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_dcpcrypt\dcpcrypt_laz.lpk
c:\lazarus\lazbuild.exe c:\tranquilit\pltis_synapse\laz_synapse.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_luipack\luicomponents\luicomponents.lpk
c:\lazarus\lazbuild.exe --add-package c:\tranquilit\pltis_luipack\luicomponents\luicomponents.lpk
c:\lazarus\lazbuild.exe --add-package C:\tranquilit\pltis_python4delphi\PythonForDelphi\
↪Components\p4dlaz.lpk
c:\lazarus\lazbuild.exe --add-package C:\lazarus\components\anchordocking\design\
↪anchordockingdsgn.lpk
c:\lazarus\lazbuild.exe --build-ide=
c:\lazarus\lazbuild.exe c:\tranquilit\wapt\wapt-get\pltis_wapt.lpk

REM depending on version, change community to enterprise
waptpython build_exe.py community
```

Installer l'environnement serveur sur Windows

```
cd \tranquilit\wapt
waptpython waptserver\winsetup.py all
```

Créer les installeurs InnoSetup

- installer Innosetup https://jrsoftware.org/download.php/ispack-unicode.exe

Les fichiers `.iss` se situent dans `C:\tranquilit\wapt\waptsetup` ;

L'installeur **`waptsetup`** inclut les bibliothèques python, l'outil en ligne de commande **`wapt-get`**, le webservice local **`waptservice`**, les outils de packaging et la console WAPT **`waptconsole`**.

Le programme **`waptserver`** permet de construire un installeur qui inclut un serveur Nginx en frontal pour le webservice Flask **`waptserver.py`**.

L'installeur waptstarter n'inclut que le code pour le webservice local et l'outil en ligne de commande **wapt-get**, mais pas la console WAPT **waptconsole**, ni les outils de packaging.

Clic-droit sur le fichier iss → *Compiler* crée un installeur avec **InnoSetup**.

ou en ligne de commande :

```
"C:\Program Files (x86)\Inno Setup 5\ISCC.exe" C:\tranquilit\wapt\waptsetup\waptsetup.iss
```

Les paramètres généraux des installeurs sont définis par des *#define* en entête de fichier.

Si vous ne signez pas les installeurs, vous pouvez commenter les lignes `#define signtool ...`

Construire l'agent WAPT pour MacOS

Générer le paquet de l'agent

- si vous n'avez pas accès à la commande **sudo**, vous devrez activer l'utilisateur root ;
- à partir de la racine du répertoire WAPT, naviguer vers waptservice/pkg ;
- exécuter le script *createpkg* avec des droits d'administrateur ;

```
sudo ./createpkg.py
```

Il peut demander des logiciels supplémentaires (les outils de développement en ligne de commande) et les installer après une invite à laquelle vous devrez répondre : guilabel:*Yes* ;

- le paquet de l'agent WAPT aurait dû être généré, sous un nom du type `tis-waptagent-1.7.6.6550-tismacos-fdc24bca.pkg` ;

Installer le paquet de l'agent WAPT

- exécuter la commande suivante sur votre MacOS :

```
sudo installer -pkg tis-waptagent*.pkg -target /
```

Si l'installation est réussie, vous devriez avoir les fichiers wapt dans `/opt` et accéder aux commandes **wapt**, **wapt-get**, **waptpython** et **waptservice**.

- l'agent devrait se lancer au prochain redémarrage, si vous voulez le lancer tout de suite, vous pouvez avec la commande suivante :

```
sudo launchctl load -w /Library/LaunchDaemons/wapt.plist
```

Construire l'agent WAPT pour Linux

Construire l'environnement sur Debian Linux

```
mkdir ~/tranquilit/
cd ~/tranquilit/
git clone git@github.com:tranquilit/WAPT.git
cd ~/tranquilit/wapt/waptserver/deb
python createdeb.py
cd ~/tranquilit/wapt/waptrepo/deb
python createdeb.py
```

Tranquil IT uses various licenses to distribute software and documentation, to accept regular contributions from individuals and corporations, and to accept larger grants of existing software products.

Ces licences nous aident à atteindre notre objectif de fournir des produits logiciels fiables et durables grâce au développement collaboratif de logiciels open source.

Dans tous les cas, les contributeurs conservent tous les droits d'utiliser leurs contributions originales à toute autre fin en dehors de WAPT tout en donnant à WAPT et à ses projets le droit de distribuer et de développer leurs travaux.

6.23.2 Contributor License Agreements

Tranquil IT desires that **all contributors of ideas, code, or documentation** to any Tranquil IT projects **complete**, **sign**, and **submit** via email an Individual Contributor License Agreement (ICLA).**

The purpose of this agreement is to clearly define the terms under which intellectual property has been contributed to Tranquil IT and thereby allow us to defend the project should there be a legal dispute regarding the software at some future time. A signed Individual CLA (ICLA) is required to be on file before an individual is given commit rights to any Tranquil IT project.

For a corporation that has assigned employees to work on a Tranquil IT project, a Corporate CLA (CCLA) is available for contributing intellectual property via the corporation, that may have been assigned as part of an employment agreement.

Note that a Corporate CLA does not remove the need for every developer to sign their own ICLA as an individual, which covers both contributions which are owned and those that are not owned by the corporation signing the CCLA.

Attention: The CCLA legally binds the corporation, so it must be signed by a person with authority to enter into legal contracts on behalf of the corporation.

The ICLA is not tied to any employer you may have, so it is recommended to use one's personal email address in the contact details, rather than an @work address.

Your Full name will be published unless you provide an alternative Public name. For example if your full name is Andrew Bernard Charles Dickens, but you wish to be known as Andrew Dickens, please enter the latter as your Public name.

The email address and other contact details are not published.

6.23.3 Submitting License Agreements

Documents may be submitted by email and signed by hand or by electronic signature. Postal mail hard copy and fax are no longer supported.

When submitting by email, please fill the form with a pdf viewer, then print, sign, scan all pages into a single pdf file, and attach the pdf file to an email to contributors-agreement[at]tranquil[dot]it.

If you prefer to sign electronically, please fill the form, save it locally (e.g. icla.pdf), and sign the file by preparing a detached PGP signature. For example,

```
gpg --armor --detach-sign icla.pdf
```

The above will create a file icla.pdf.asc. Send both the file (icla.pdf) and signature (icla.pdf.asc) as attachments in the same email to contributors-agreement[at]tranquil[dot]it. Please send only one document (file plus signature) per email. Please do not submit your public key to Tranquil IT. Instead, please upload your public key to pgpkeys.mit.edu.

Indication: send to contributors-agreement@tranquil.it

The files should be named icla.pdf and icla.pdf.asc for individual agreements; The files should be named ccla.pdf and ccla.pdf.asc for corporate agreements;

Please note that typing your name in the field at the bottom of the document is not signing, regardless of the font that is used. Signing is either writing your signature by hand on a printed copy of the document, or digitally signing via gpg. Unsigned documents will not be accepted.

From wikipedia.com: A signature is a handwritten (and often stylized) depiction of someone's name or nickname, on documents as a proof of identity and intent.

For answers to frequently asked licensing questions, please consult or post on the Tranquil IT forum located at https://forum.tranquil.it/.

6.24 Changelog

6.24.1 WAPT 1.8 Serie

WAPT-1.8.2.7393 (2021-11-16)

hash : 75a5de09

This is a security release. All Wapt 1.8 version below 1.8.2.7393 are vulnerable.

- [SEC] upgrade babel python module from 2.5.1 to 2.9.1
- **[UPD] python lib upgrades urllib3, and requests** chardet==4.0.0 requests==2.26.0 urllib3==1.26.7

WAPT-1.8.2.7388 (2021-10-07)

This is a security release. All Wapt 1.8 version below 1.8.2.7388 are vulnerable.

Security changelog wapt-1.8.2.7388*

- [SEC] fix for vuln in urllib3 CVE-2021-33503 (CVSS Score : 7.5 High, CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H)
- [SEC] Sanitize filename used when downloading files on local client. (CVSS Score : 7.5 High, CVSS;3.1/AV:L/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H/E:U/RL:O/RC:C) Enforced on wget and local filenames for downloaded packages (chars “' “..” @ | ( ) : / , [ ] < > * ? ; ` n are removed or replaced)
- [SEC] don’t use PackageEntry filename attribute to build target package filename as it is not signed.
- [FIX] Waptconsole config : When retrieving server side https certificate don’t write UTF16 string for in waptconfig. Remove wildcards from CN of certificate to compose cert filename.
- **[UPD] update python modules requirements following urllib3 upgrade** certifi==2021.5.30 chardet==3.0.2 idna==2.8 requests==2.21.0 urllib3==1.24.3

WAPT-1.8.2.7373 (2021-08-10)

hash : e96e569c

This is a security fix version affected by CVE-2021-38608.

Please visit the security bulletin to learn more.

WAPT-1.8.2.7372 (2021-06-21)

WAPTAgent:

- [FIX] fix regression on macos build after dependency upgrade
- [FIX] _update_db : error in for the calculation of next_update_on forpackage attributes valid_until and forced_install_on
- [IMP] be sure to not use waptguihelper when running as system user
- [UPD] add –use-gui for vscode / pyscripter build-upload of package

WAPTServer:

- [FIX] Fix regression on proxy setting for waptserver

Setuphelpers:

- [IMP] add func split_arg_string to split a command line into executable / args list

WAPT-1.8.2.7357 (2021-02-09)

WaptCore:

- [FIX] Be tolerant with target_os = “all” in windows.
- [FIX] in installed_softwares, ignore error when key can not be opened

because of encoding issues (_winreg does not handle unicode, but ansi).

- [IMP] shown “” instead of None in wapt-get tables.
- [FIX] Update timestamping server and openssl hash:

http://timestamp.globalsign.com/scripts/timestamp.dll.

- [FIX] be tolerant if no “id” attribute in installed packages report.
- [FIX] match properly packages with target_os = all.
- [IMP] prepare installed_packages for upgrade to wapt 1.9.
- [IMP] add Timeit class for test purposes.
- [IMP] disable sending unused data waptwua_rules_packages.
- [FIX] waptupgrade regression Bug introduced in Revision:

85686e4d631adb6e13b25146f3a81f3c09ca082d.

- [FIX] CA certificate PEM string stored as utf16 in certificate chain

when creating a certificate signed by a CA (enterprise).

WaptConsole:

- [IMP] increase width of AD-Site combobox.
- [IMP] report packages install_id in console.

WaptAgent:

- [IMP] wapt-get: add –newest-only for search.
- [IMP] waptexit: add ExceptionLogger. Change the way exceptions are handled

in threads to try to fix issues when waptexit hangs and can not be closed.

WaptServer:

- [FIX] don’t actually update the listening websocket session ID

if it is already set in hosts table.

- [IMP] wapttasks : force remove tasks locks at service startup.

WAPT-1.8.2.7334 (2020-12-03)

hash : 2d15afd9

This is a bugfix release. Ubuntu 16.0.4 amd64 and Debian 10 armhf clients are now supported.

Fixes and enhancements

- [FIX] fix base proxy string « » when editing a profile package.
- [FIX] fix « Unable to create file » when editing a profile package.
- [FIX] don't allow to save a self service rules packages without a name.
- [FIX] fix Access violation when importing from file.
- [FIX] fix issue with download_icons.
- [FIX] improve search in waptconsole (search on concatenation

of software name and software version).

- [FIX] fix extract CN from ssl client cert authentication for get_auth_token

when windows client computer has an organization (in this case client csr/cert has a CN=<uuid>,O=<org> subject).

- [FIX] fix regression on wakeonlan introduced by backported code from 1.9.
- [FIX] PostgreSQL DB not correctly migrating from some 1.8.1.X.
- [IMP] Add key param for install_msi_if_needed in setuphelpers_windows.py.
- [FIX] Fix for no_fallback in repositories rules.
- [FIX] Soupsieve python lib is set to 1.9.6 in requirements because later

version are Python3 only.

- [FIX] Patch for SocketIO with proxy.
- [FIX] Fix triggers for repository sync in PostgreSQL who were not correctly

migrated (Enterprise only).

- [IMP] Two new builders for both server and agent : Ubuntu 16.0.4 LTS /

ARM x86 Debian 10 (Enterprise only).

- [IMP] Revert dhparam bits size to 1024 bits in Windows WAPT Server because it

took too much time to generate. It can be generated afterward.

- [IMP] Increase default clockskew for signed action to 6 hours (before it was

only 1 hour).

- [FIX] waptcrypto: security fix: prevent infinite loop in SSLCABundle.certificate_chain

if issuer cert and signed cert have the same subject but one has no authority_key_identifier.

- [FIX] waptcrypto : fix revoke_cert, handle list of DNS names for certificates,

fix AuthorityKeyIdentifier when neregating certificate from CSR.

- [FIX] waptservice fix for verify_cert_ldap in waptagent.

- [FIX] Patch pltis_utils to display properly long integer in WAPTWUA.

The `wsusscn2.cab` file may report KBs with incorrect huge download size up to 1TB.

- [FIX] On a fresh install the admin ACL rights were not properly set up

which required a service restart to get fixed.

- [FIX] Force admin password change on upgrade if the old hash is SHA-1.
- [FIX] minor fixes for uWSGI support.
- [FIX] Fix temporary directories not removed after package import or edit.
- [FIX] Fix duplicated auth_module_ad.py module

in bad waptwaptenterprise directory on windows waptserver.

- [IMP] Warning of wapt licence expiration message changed from 14 days

to 60 days before expiration.

- [FIX] Fix broadcast for wakeonlan.
- [FIX] fix additional server password issues when non ascii character.

Library changes

- [UPD] Update OpenSSL binary from 1.0.2r to 1.0.2u.
- [UPD] Update Python4Delphi lib to 20201020 release.
- [UPD] Build now with Lazarus 2.0.8 and FPC 3.0.4.

WAPT-1.8.2.7269 (2020-06-16)

hash : 757cdc76

- [FIX] Fix db schema upgrade script for upgrade from WAPT version 1.8.1-6742.

Fresh 1.8.2 installation or upgrade from 1.7 or from 1.8.0 or 1.8.1-6758 shouldn't have the issue.

- [IMP] Add key for install_msi_if_needed.
- [FIX] Fix for no_fallback for waptwua (**Enterprise** only).

WAPT-1.8.2.7267 (2020-06-12)

hash : 46f40312

- [FIX] Fix db schema upgrade script for upgrade from WAPT version 1.8.1-6742.

Fresh 1.8.2 installation or upgrade from 1.7 or from 1.8.0 or 1.8.1-6758 shouldn't have the issue.

WAPT-1.8.2.7265 (2020-06-11)

hash : 339f1996

This is mostly a bugfix release. Support for Linux and Mac clients has also been greatly improved.

Notable enhancements

- [IMP] improve support for WaptAgent on Linux and Mac.

Now the support is almost identical on Windows, Linux and MacOS (all versions):

- waptagent installation as a service with kerberos registration.
- waptselfservice gui available on the 3 platforms

(note: support for the latest version of MacOS, Catalina, is expected for 1.8.3).

- waptexit (on Linux an Mac it is not yet started

on system shutdown, it can be triggered by a scheduled task).

- session-setup for configuring user sessions.
- send messagebox to users and propose upgrades (**Enterprise** only).
- OU handling (**Enterprise** only).
- waptselfservice authentication can be delegated

to the waptserver (**Enterprise** only).

- better *setuphelpers* coverage.
- [IMP] new supported platforms. Now WAPT for linux (server and agent)

and MacOS (agent only) supports:

- Ubuntu 18.04 and 20.04;
- Debian 8, 9 and 10;
- Centos7 (CentOS 8 as a preview);
- MacOS Sierra, HighSierra, Mojave (note: support for MacOS Catalina

expected for WAPT 1.8.3).

- [IMP] streamlining of development environment

for packaging on Linux using VSCode.

- [FIX] better handling of websocket cleanup when a host

is not properly registered. Should improve stability on large WAPT installations.

- [IMP] selfservice can now be configured for external authentication

for desktops that are not in a AD Domain.

- [IMP] selfservice users can now authenticate on waptserver

even when out of the corporate network.

- [IMP] The session setup in run for all packages immediately

after upgrade or install, so that new packages are already configured in the context of each logged in users (no need to logout / login) (**Enterprise** only).

- [IMP] If secondary repositories are defined in `waptconsole.ini`,

additional packages can be selected when editing hosts, groups or self-service packages.

- [IMP] When editing group or self-service packages,

one can define the Target OS of the package.

- [IMP] Remote message to logged in users is using the same custom dialog box

for Windows, Linux and macOS.

- [IMP] Remote message to logged in users can display the same custom logo

as self-service (**Enterprise** only).

- [IMP] The IP/Subnet match in repository access rules is based on the « main IP »

of the host (source IP from which the host is reaching the server, if the server is public, this is usually the external IP of the router) (**Enterprise** only).

- [IMP] Added Remote host Shutdown and remote host Reboot from Waptconsole

if enabled in wapt-get.ini (`allow_remote_shutdown` and `allow_remote_reboot`) (**Enterprise** only).

- [IMP] Add a *no fallback* checkbox in repositories access rule

to prevent host using main repository in case secondary ones are not reachable (when main repository bandwidth is limited, having all hosts reaching the main repository can slow down access to the main site) (**Enterprise** only).

- [FIX] Make sure WUA install task are executed after packages install

(**Enterprise** only).

Other enhancements

- [IMP] Cmd Console is hidden when session-setup is running,

to limit annoyance for users.

- [IMP] WUA direct download option in waptconsole (**Enterprise** only).
- [IMP] can now use microsoft url for WUA in rules (**Enterprise** only).
- [FIX] Improved background icons loading in self-service.
- [FIX] better inventory of `lastboottime` and `get_domain_info`.
- [FIX] better handling of other local install of Python

on client computer (eg. conflict with local Anaconda Python installation).

- [IMP] allows to have multiple private repo content displayed in waptconsole.
- [IMP] remote repository: it is now possible to prevent a fallback.
- [FIX] better handling of icons in selfservice.
- [IMP] improved support for VSCode.
- [FIX] better handling of ipv6 in console and inventory.

- [IMP] `wapt_admin_filter`: local admin can be filtered out

like normal user in selfservice.

- [IMP] add a larger support for setuphelpers on macOS.
- [FIX] waptserver logs are properly redirected

to `/var/log/waptserver.log`.

- [FIX] package caching: packages are deleted after each successful installation

(rather than at the end of the whole upgrade) to better keep local disk space.

- [IMP] allows usage of url for changelog in control file.
- [IMP] better support for Windows Update download directly

from Microsoft if WAPTServer is not reachable.

- [FIX] better handling of upgrade from Community version

to Enterprise version.

- [IMP] improved local store skin and translations.
- [FIX] bugfixes and minor gui improvements.

Library changes in WAPT-1.8.2.7165

- [CHANGE] replaced **`python-ldap`** with **`ldap3`**.
- [FIX] upgraded **`ujson`** on waptagent and waptserver on Linux.

Removed features with WAPT-1.8.2.7165

- [REMOVED] autoconfiguration of repositories based on SRV DNS fields

(it was not working anymore anyway).

Caveats when using WAPT-1.8.2.7165

- [CAV] WaptExit is not run automatically on shutdown

on Linux or MacOS (current issue with **`systemd`** / launched integration).

- [CAV] WaptTray is not yet available on Linux and macOS.
- [CAV] MacOS Catalina is supported by the WaptAgent,

however WAPTSelfService and WaptExit are not yet supported.

WAPT-1.8.2.7265 RC2 (2020-05-29)

hash git : 339f1996

This is a Release Candidate version for testing and evaluation only and should not be installed on production system.

This is mostly a bugfix release. Support for Linux and Mac clients has greatly improved.

Notable enhancements over 1.8.2 RC1

- [IMP] the session setup in run for all packages immediately after upgrade

or install, so that new packages are already configured in the context of each logged in users (no need to logout / login) (**Enterprise** only).

- [IMP] if secondary repositories are defined in waptconsole.ini,

additional packages can be selected when editing hosts, groups or self-service packages.

- [IMP] when editing group or self-service packages,

one can define the target OS of the package.

- [IMP] remote message to logged in users is using the same custom dialog box

for windows, linux and macOS.

- [IMP] remote message to logged in users can display the same custom logo

as self-service (**Enterprise** only).

- [IMP] the IP / Subnet match in repository access rules is based

on the *main IP* of the host (source IP from which the host is reaching the server, if the server is public, this is usually the external IP of the router) (**Enterprise** only).

- [IMP] added remote host shutdown and remote host reboot from Waptconsole

if enabled in wapt-get.ini (`allow_remote_shutdown` and `allow_remote_reboot`) (**Enterprise** only).

- [IMP] added a *no fallback* checkbox in repositories access rule

to prevent hosts using main repository in case secondary repositories are not reachable (when main repository bandwidth is limited, having all hosts reaching the main repository can slow down access to the main site) (**Enterprise** only).

- [FIX] make sure WUA install task are executed

after packages install (**Enterprise** only).

Other enhancements over 1.8.2 RC1

- [IMP] cmd Console is hidden when session-setup is running,

to limit annoyance for users.

- [IMP] WUA direct download option in waptconsole (**Enterprise** only).
- [IMP] can now use Microsoft url for WUA in rules (**Enterprise** only).
- [IMP] improved background icons loading in self-service.

Removed features

None

Caveats

Same as RC1

WAPT-1.8.2.7165 RC1 (2020-05-29)

hash git : 1387b38f

This is a Release Candidate version for testing and evaluation only and should not be installed on production system.

This is mostly a bugfix release. Support for Linux and macOS clients has greatly improved.

Notable enhancements in WAPT-1.8.2.7165 RC1

- [IMP] improve support for WaptAgent on Linux and Mac.

Now the support is almost identical on Windows, Linux and MacOS (all versions):

- waptagent installation as a service with kerberos registration.
- waptselfservice gui available on the 3 platforms

(note: support for the lastest version of MacOS, Catalina, is expected for 1.8.3).

- waptexit (on Linux an Mac it is not yet started

on system shutdown, it can be triggered by a scheduled task).

- session-setup for configuring user sessions.
- send messagebox to users and propose upgrades (**Enterprise** only).
- OU handling (**Enterprise** only).
- waptselfservice authentication can be delegated

to the waptserver (**Enterprise** only).

- better setuphelpers coverage.
- [IMP] add new supported platform. Now WAPT for linux (server and agent)

and MacOS (agent only) supports:

- Ubuntu 18.04 and 20.04;
- Debian 8, 9 and 10;
- Centos7 (CentOS 8 as a preview);
- MacOS Sierra, HighSierra, Mojave (note: support for MacOS Catalina

expected for WAPT 1.8.3);

- [IMP] streamlining of development environment

for packaging on Linux using VSCode.

- [FIX] better handling of websocket cleanup when a host

is not properly registered. Should improve stability on large WAPT installation.

- [IMP] selfservice can now be configured for external authentication

for desktops that are not in an Active Directory Domain.

- [IMP] selfservice users can now authenticate on selfserver

even when out of the corporate network.

Other enhancements in WAPT-1.8.2.7165 RC1

- [FIX] better inventory of `lastboottime` and `get_domain_info`.
- [FIX] better handling of other local install of Python

on client computer (eg. conflict with local Anaconda Python installation).

- [IMP] allows to have multiple private repo content displayed in waptconsole.
- [IMP] remote repository: it is now possible to prevent a fallback.
- [FIX] better handling of icons in selfservice.
- [IMP] improved support for VSCode.
- [FIX] better handling of ipv6 in console and inventory.
- [IMP] `wapt_admin_filter`: local admin can be filtered out

like normal user in selfservice.

- [IMP] add a larger support for setuphelpers on macOS.
- [FIX] waptserver logs are properly redirected

to `/var/log/waptserver.log`.

- [FIX] package caching: packages are deleted after each successful installation

(rather than at the end of the whole upgrade) to better keep local disk space.

- [IMP] allows usage of url for changelog in control file.
- [IMP] better support for Windows Update download directly

from Microsoft if WAPTServer is not reachable.

- [FIX] better handling of upgrade from Community version

to Enterprise version.

- [IMP] improved local store skin and translation.
- [FIX] bugfixes and minor gui improvements.

Library changes in WAPT-1.8.2.7165 RC1

- [REF] replaced **python-ldap** with **ldap3**.
- [FIX] upgraded **ujson** on waptagent and waptserver on Linux.

Removed featured with WAPT-1.8.2.7165 RC1

- autoconfiguration of repositories based on SRV DNS fields

(it was not working anymore anyway).

Caveats when using WAPT-1.8.2.7165 RC1

- [CAV] WaptExit is not run automatically on shutdown

on Linux or MacOS (current issue with systemd / launched integration).

- [CAV] WaptTray is not yet available on Linux and MacOS.
- [CAV] MacOS Catalina is supported by the WaptAgent,

however WAPTSelfService and WaptExit are not yet supported.

WAPT-1.8.1-6758 (2020-03-06)

(hash bb93ce41)

On server:

- [REF] refactoring for postconf.py / remove old migration from MongoDB;
- [REF] refactoring for winsetup.py / create now a `dhparam`

for **nginx** on Windows;

- [REF] refactoring for repositories: change repo_diff by remote_repo_diff /

add param `remote_repo_websockets` (by default to True) on server;

- [IMP] disable cache on **nginx** for Windows and Linux on wapt packages / exe;

On agents:

- [REF] change param `waptservice_admin_auth_allow`

by `waptservice_admin_filter`;

- [REF] delete resync functions for remote repo;
- [IMP] param `local_repo_sync_task_period` by default to « 2h »;
- [FIX] wapt-get / waptservice debug when download a package on linux

when not sudo;

- [FIX] fix for **plist** in macOS;
- [IMP] can now have relative path for packages/directories

in **wapt-get**;

- [IMP] templates have by default setup_uninstall / update etc...
- [IMP] improvements with templates for vscode;

On waptconsole:

- [IMP] add possibility of template packages for deb / rpm / pkg;
- [FIX] Fix for msi, exe, etc in PackageWizard explorer;
- [IMP] Can now choose `editor_for_packages` directly in waptconsole config;
- [UPD] Some cosmetic / translations improvements for GUI to deploy waptagent;

WAPT-1.8.1-6756 (2020-02-17)

(hash 43394f3b)

Bug fixes and small improvements

- [IMP] waptconsole: improve the refresh of hosts grid when a lot of hosts

are selected (improved by a factor of around 5)

- [FIX] waptserver Database connections management: don't close DB on teardown

as it should not occur, and seems to trigger some issue when triggering a lot of tasks on remote hosts (error db is closed)

- [FIX] waptconsole: Don't « force » install when triggering the upgrade

on remote hosts, to avoid reinstalling softwares when already up to date.

- [IMP] use *ldap auth* only if session and admin fail (avoid waiting for timeout

when ldap is not available but one wants to login with plain admin user);

- [FIX] wapt-get upload: encode user and password in `http_upload_package`

to allow non ascii in admin password;

- [IMP] waptconsole: Disable auto search on keywords;
- [IMP] use DMI `System_Information.Serial_Number` information

for serialnr Host field instead of `Chassis_Information.Serial_Number` because System_Information is more often properly defined;

- [IMP] waptconsole: add `uuid` in the list of searched fields

when only "host" is checked in filters;

- [IMP] nginx config: disable caching;
- [IMP] fixes for **vscode** project template;

WAPT-1.8.1-6742 (2020-02-12)

(hash 80dbdbe7)

Major changes

- waptconsole: Added a page to show packages install status summary (merge)

of all selected hosts, grouped by `package`, `version`, `install status`, with count of hosts;

Context menu allow to apply selectively the pending actions. On enterprise, one can apply safely the updates (only packages for which there is no running process on client side);

- Prevent users from saving a host package if targeted host(s) do not accept

their personal certificate. (Checked on waptconsole when editing / mass updating host packages, and on server when uploding packages);

The personal certificate file `.crt` must contain at first the personal certificate, followed by the issuer CA certificates, so that wapt can rebuild the certificate chain and check intersection with host's trusted certificates;

Important note about SSL client side authentication

In your nginx configuration, be sure to reset the headers `X-Ssl-Authenticated` and `X-Ssl-Client-DN` as waptserver *trusts* these headers if ssl cient side auth is enabled in `waptserver.ini`;

If SSL client side auth is setup these headers can be populated by `proxy_set_header` with result of `ssl_verify_client` as explained in ./wapt-security/security-configuration-certificate-authentication.html#enabling-client-side-certificate-authentication;

Fixes and detailed changelog

- Security fix: update waitress module to 1.4.3

(CVE-2020-5236);

- Security fix: blank `X-Ssl*` headers in default **`nginx`** templates;
- Fix: regression: **`kerberos register_host`** did not work anymore;
- On server, :file:"<repository root>/wapt/ssl" dir is moved automatically

on winsetup / postconf to (per default) :file:"<repository root>/ssl", a `/ssl` location is added;

This `/ssl` should be accessible from clients at the location specified by the server parameter `clients_signing_crl_url` (in `waptserver.ini`);

- Improved logs readability. Log count of used DB connections

from pool on waptserver to troubleshoot DB connection issues. Log level can be specified by subcomponent with loglevel_waptcore, loglevel_waptserver, loglevel_waptserver.app, loglevel_waptws, loglevel_waptdb defined in `waptserver.ini`;

- Reworked explicit DB Open/close on waptserver to not get

a DB connection from pool if not useful. It prevents exhaustion of DB connections;

- waptwinsetup: don't create unused directories `wapt-group`

and `waptserverlog`;

- Added *`.msu`* and *`.msix`* extensions

for Package wizard setup file dialog;

- Fallback with os._exit(10) for waptservice restart.

Added a handler in **`nssm.exe`** configuration to honor the restart;

- Increased waitress threads to 10 on waptservice;
- Lowered the default number of pooled DB connections (`db_max_connections`)

to 90, to be lower than postgresql default of 100;

- waptserver: allow kerberos or ssl auth check in waptserver

only if enabled in `waptserver.ini` config file;

- waptconsole: Allow update of host package only if user certificate

is actually allowed on the host (based on last update of host status in database);

- waptconsole / build waptagent: checkbox to specify to include or not

non certificate authority certificates in build. The normal setup would be to uncheck this, to not deploy non CA certificates, on wapt root CA;

- [IMP] Add and option to disable automatic hiding of panels...
- [IMP] Add explicit AllowUnauthenticatedRegistration task to waptserversetup windows
- waptsetup: Remove explicit VCRedistNeedsInstall task. Use /VCRedistInstall=(0/1)

if you need to force install or force not install vcredist VC_2008_SP1_MFC_SEC_UPD_REDIST_X86;

- [FIX] **`wapt-get.exe`**: use wapt-get.ini for :command:"scan-packages"

and :command:"update-packages" wapt-get actions;

- [FIX] **`wapt-get`**: auth asked when checking if server is available (ping)

and client ssl auth is enabled;

- [IMP] WAPT client: if client ssl auth failed with http error 400,

retry without ssl auth to be able to ask for new certificate signing;

- [FIX] waptserver register behavior: revert over rev 6641: sign host certificate

if an authenticated user is provided or data is signed with a key which can be verified by existing certificate in database for this host uuid;

- [IMP] waptserver register behavior: when receiving 401 from server when registering,

retry registering without ssl auth;

- [IMP] wapt client: be sure to have proper host private key saved

on disk when receiving signed certificate from server;

- [IMP] waptconsole: advanced filters for selected host packages status.

Filter on *Install status* and *Section + keyword*. *Pending* button to show only pending installations / removes;

- [ADD] wapt-get make-template / edit package: Add .vscode directory.

Add template project for vscode;

- [FIX] waptconsole: fix ssl auth for mass package dependencies

/ conflicts updates;

- [FIX] waptconsole: fix import packages from external repos with ssl auth;
- [IMP] backports from master:
- target OS in import packages;
- choose editor for packages in linux in cmdline;
- [IMP] backports from master:
- refactoring for `HostCapabilities.waptos`;
- add new `target_os` unix for mac and linux;
- so `target_os`: windows, darwin (for mac), linux or unix;
- [FIX] `WAPT.wapt_base_dir`;
- [FIX] makepath in linux/macOS;
- [IMP] refactoring / fixes for setuphelpers;
- [FIX] for `rights_to_check` in repo-sync client;
- [FIX] for repo-sync;
- [ADD] two setuphelpers for linux: type_debian and type_redhat

indent the local sync.json;

- [IMP] use `get_os_version` and `windows_version_from_registry`

instead of `windows_version`;

- [IMP] use `windows_version_registry` for `get_os_version` on windows;
- [IMP] backport `host_capabilities.os` from master
- [FIX] for **`make-template`** for malformed *`.exe`* installer;
- [ADD] automatic maintenance of a CSR for client auth certificates

signed by server:

- default CSR lifetime to 30 days;
- check renewal of client cert CSR every hour;
- added a parameter for the next update time of crl;
- added `clients_signing_crl_url`, `clients_signing_crl_days`,

`known_certificates_folder` waptserver parameters;

- added a `/ssl` location in nginx templates;
- added `crl_urls` in client auth signed certificates;
- added a scheduled task to renew server side crl;
- added `clients_signing_crl` waptserver parameter to add client cert

to server crl when host is unregistered;

- added **`revoke_cert`** method to SSLCRL class;
- added a `authorityKeyIdentifier` to the client auth CSR;
- force restart if windows task is broken;
- waptservice: use `sys._exit(10)` to ask **`nssm`** to restart service

in case of unhandled exception in waptservice (loops, etc.);

- wapt client: don't log / store into db Wapt.runstatus if not changed;
- waptserver postconf: fix for rights on some wapt directories;
- Add mutual conflicts to deb/rpm packages for waptagent/waptserver

to avoid simultaneous install;

WAPT-1.8.0-6641 (2020-01-24)

(hash 3dbb3de8)

Major changes

- [ADD] client Agent for Linux Debian 8, 9 , 10, Linux Centos 7, Ubuntu 18, 19

and MacOS. The packages are named wapt-agent and available in https://wapt.tranquil.it/wapt/releases/latest/;

- [IMP] repository access rules defined in waptconsole. Depending of client IP,

site, computername, one can define which secondary repository URL to use (**Enterprise** only);

As a consequence, the DNS query method (with SRV records) is no more supported for repositories

- [IMP] the package and signature process has been changed to be compatible

with **`python3`**. Serialization of dict is now sorted by key alphabetically to be deterministic across python versions. WAPT agents prior to version 1.7.1 will not be able to use new packages. (see git hash SHA-1: f571e55594617b43ed83003faeef4911474a84db);

- [NEW] a WAPT agent can now be declared as a secondary remote repository.

Integrated syncing with main server repository is handled automatically. (**Enterprise** only);

- [NEW] waptconsole can now run without elevated privileges.

The build of waptagent / waptupgrade package are done in a temporary directory. **When editing a package from waptconsole, :program:`PyScripter` should be launched with elevated privileges**;

One could deploy the agent with GPO without actually rebuilding a waptagent. Command line options are available on stock waptsetup-tis.exe to configure repo url (`/repo_url=`), server url (`/wapt_server=`), server certificate bundle location (`/CopyServersTrustedCA=`), packages certificates checking (`/CopyPackagesTrustedCA=`), `/use_random_uuid`, `/StartPackages`, `/append_host_profiles`, `/DisableHiberBoot`, `/waptaudit_task_period`;

Some options are still missing and may be added in a future release;

- [IMP] package filename now includes a hash of package content to make it easier

to check if download is complete and if package has been scanned (improved speed for large number of packages);

- [SEC] the WAPT admin password must be regenerated (with postconf);

if it is not *pbkdf2* based. See in your `waptserver.ini` file, `wapt_password` must start with **$pbkdf2-**;

Fixes and detailed changelog

- [SEC] waptagent can optionally be digitally signed,

if (1) Microsoft **signtool.exe** is present in <wapt>utils` and (2) if there is a pkcs#12 :mimetype:.p12` file with the same name as the personal certificate *.crt* file, and (3) the certificate is encrypted with the same password;

- [IMP] wapt-get.py can be run on linux and macos in addition to windows;
- [IMP] waptconsole host's packages status reporting: now displays current version

with *NEED-UPGRADE*, *NEED-REMOVE*, *ERROR* status and future version with *NEED-INSTALL* status;

The status is stored in server's DB HostPackagesStatus so it can be queried for reporting;

- [IMP] setuphelpers: there now different setuphelpers

for each operating system family;

- [ADD] waptconsole: added an action to safely trigger upgrades on remote hosts

only if associated processes (impacted_process control attribute) are not running, to avoid disturbing users (**Enterprise** only);

- [ADD] **wapt-get --service upgrade**: added handling of --force,

--notify_server_on_start=0/1, notify_server_on_finish=0/1 switches;

- [IMP] package signature's date is now taken in account when comparing packages;
- [ADD] host_ad_site key in [global] in wapt-get.ini to define

a *fake* Active Directory site for the host;

- [ADD] waptconsole / packages grid: if multiple packages are selected,

the associated *show clients* grid shows the status of packages for all selected clients (**Enterprise** only);

- [ADD] waptagent build: added checkbox to enable repository rules lookup

when installing agent (**Enterprise** only);

- [ADD] waptconsole / import packages: don't reimport existing dependencies.

Checkbox to disable import of dependencies;

- [IMP] wapt-scanpackages speed optimizations: don't re-extract certificates

and icon for skipped package entries. use md5 from filename if supplied when scanning.

- [FIX] waptexit: fix arguments to waptexit for only_if_not_process_running

and install_wua_updates (bool);

- [FIX] waptagent / waptwua fix wapt wua enabled setting reset to *False*

when upgrading with waptagent and enabled=don't touch;

- [FIX] waptserver / waptwua repository: all cabs files are now

in root directory instead of microsoft original file tree. The files are moved when upgrading to 1.8;

- [IMP] waptupgrade package: increment build number if building

a new waptagent of the same main wapt version;

- [NEW] waptserver parameter trusted_signers_certificates_folder:

Path to trusted signers certificate directory. If defined, only packages signed by this trusted CA are accepted on the server when uploading through server;

- [NEW] waptserver parameter `remote_repo_support`: if true,

a task is scheduled to scan repositories (`wapt`, `waptwua`, `wapt-hosts`) that creates a `sync.json` file for remote secondary repositories;

- [IMP] when building waptagent, don't include non CA packages certificates

by default in waptagent. A checkbox is available to still enable non CA certificates to be scanned and added;

- [IMP] when building waptagent, one can add or remove certificates

in the grid with `Ctrl+Del` or drag and drop;

- [FIX] waptconsole / host packages status grid: fixed `F5` refresh;
- [IMP] waptconsole / build agent: build an enterprise agent even

if no valid licence (**Enterprise** only);

- [FIX] `forced_update_on` control attribute: don't take into account

for `next_update_on` if in the past;

- [IMP] waptconsole: try to accept waptserver password with non ASCII characters;
- [REMOVED] waptstarter: remove *socle* from default host profile;
- [IMP] waptagent build: rework of server certificate path relocation

when building / installing;

- [SEC] don't sign agent certificate if no valid human authentication

(admin, passwd or ldap) or kerberos authentication has been provided:

- be explicit on authentication methods;
- store registration authentication method in db only

if valid human authentication or kerberos authentication has been provided;

- when registering, be sure we trust an already signed certificate

with CN matching the host;

- store the signed host certificate in server DB on proper registration;
- [IMP] some syntax preparation work for future python3;
- [IMP] some preparation work for detailed ACL handling (**Enterprise** only);
- [FIX] don't enable client ssl auth by default in waptserver as nginx reverse

proxy server is perhaps misconfigured;

Python libraries / modules updates

- use **waitress** for waptservice wsgi server

instead of unmaintained **Rocket**`;

- **Flask-SocketIO 3.0.1** -> **Flask-SocketIO 4.2.1**;
- **MarkupSafe 1.0** -> **MarkupSafe 1.1.1**;
- **python_ldap-2.4.44** -> **python_ldap-3.2.0**;

6.24.2 WAPT 1.7 and older

(hash 1c00cefd)

- [FIX] waptserver: add fix to workaround

flask-socketio bug (AttributeError: "Request" object has no attribute "sid");

- [IMP] waptserver: be sure db is closed before trying to open it

(for dev mode);

- [IMP] waptserver: add logs messages when an exception message

is sent back to the user;

(hash ad237eee)

- [IMP] waptserver: upgrade **peewee** DB python module to 3.11.2.

Explicit connection handling to DB to track potential limbo connections (which could lead to db pool exhaustion);

- [FIX] waptwua: trap exception when pushing WU to Windows cache to allow

valid updates to be installed even if some could not be verified properly;

(hash2090b0e6d52cecfb04f8fa4c279e7c0a0252d6e2

- [FIX] **wapt-get session-setup**: fix bad print in **session_setup**.

Regression introduced in b30b1b1a550a4 (1.7.4.6229);

(hash 391d382f)

- [IMP] return server git hash version and edition in ping and `usage_statistics`;
- [IMP] be sure to have `server_uuid` on windows when during setup;
- [FIX] *`.git`* partially included in built package `manifest`;

(hash b30b1b1a)

- [FIX] 100% cpu load on one core on waptserver even when Idle;
- **python-engineio** upgrade to 3.10.0;
- **python-socketio** upgraded to 4.3.1;
- [IMP] don't try run **session_setup** on packages

which don't have one defined;

- [IMP] limit text output on console (for faster output);

(hash 86ddeaa2d)

- [FIX] Newlines in packages installs logged output;
- [FIX] Allow nonascii utf8 encoded user and password for server basic auth;
- [UPD] waptconsole: Default package filtering to x64 and console locale

to avoid mistakes when importing;

- [IMP] waptconsole: increase default Port Socket listening test timeout

(for rdp, remote service access etc..) to 3s instead of 200ms;

- [IMP] waptconsole: sort OU

by description in treeview:

Right click changes current row selection in OU treeview;

- [NEW] option to set `waptservice_password` = **NOPASSWORD**

in waptstarter installer;

- [FIX] grid sorting for package / version / size of packages;
- [FIX] don't create waptconsole link for starter;
- [NEW] **`wapt-scanpackages`**: add an option to update

the local packages DB table from `Packages` file index;

- [FIX] regression introduced in previous build: `maturities` = **PROD**

and `maturities` = **"** are equivalent when filtering allowed packages;

- [FIX] waptconsole: grid headers too small for highdpi;
- [UPD] waptupgrade package filename: keep old naming

without *all* arch (for backward compatibility);

- [IMP] `waptservice_timeout` = **20** seconds now;
- [FIX] AD auth for waptconsole with non ASCII chars;
- [IMP] missing french translations for columns

in *Import packages* grid;

- [FIX] be sure to terminate output threads in waptwinutils.run;
- [IMP] avoid showOnTop flickering for VisLoading;
- [IMP] setuphelpers.run_powershell!

add `$ProgressPreference` = **SilentlyContinue** prefix command;

- [SEC] waptservice: protect test of `host_cert` date if file is deleted

outside of service scope;

- [IMP] WaptBaseRepo class:
- packages cache handling when repo parameters (filters...) are changed;
- allow direct setting of cabundle for WaptBaseRepo;
- keep a fingerprint of input config parameters;

- [UPD] set a fallback calculated `package_uuid` value in database

for compatibility with old package status reports;

(hash f9cb3ebd)

- [IMP] revert package naming of waptupgrade to previous one to ease upgrade

from previous wapt;

- [IMP] increase `waptservice_timeout` to 20 seconds per default;
- [FIX] AD auth when there are non ascii chars (encoding);
- [FIX] missing french translations for columns in Import packages grid;
- [IMP] set a fallback calculated `package_uuid` in database

for old package without `package_uuid` attribute in db status report;

- [NEW] **`wapt-scanpackages`**: add an option to update

the local Packages DB table from Packages file index;

- [NEW] option to filters `maturities`;

(hash 3e00ac6688)

- [SEC] update python modules **`python-engineio`** and **`werkzeug`**

to fix vulnerability CVE-2019-14806

GHSA-j3jp-gvr5-7hwq

- [UPD] Python modules:
- **`eventlet 0.24.1`** -> **`eventlet 0.25.1`**;
- **`flask 1.0.2`** -> **`flask 1.1.1`**;
- **`greenlet 0.4.13`** -> **`greenlet 0.4.15`**;
- **`itsdangerous 0.24`** -> **`itsdangerous 1.1.0`**;
- **`peewee 3.6.4`** -> **`peewee 3.10`**;
- **`python-socketio 1.9.0`** -> **`python-socketio 4.3.1`**;
- **`python-engineio 3.8.1`** -> **`python-engineio 3.9.3`**;
- **`websocket-client 0.50`** -> **`websocket-client 0.56`**;
- [UPD] default `request_timeout` = **15s** for client websockets;
- [FIX] when building packages, excluded directories (for example *`.git`*

or *`.svn`*) were still included in `manifest` file;

- [UPD] don't canonicalize package filenames by default when scanning

server repository to ease migration from previous buggy wapt;

- [FIX] package filename not rewritten in `Packages` when renaming package;
- [NEW] **`wapt-scanpackages`**: added explicit option to trigger rename

of packages filenames which do not comply with canonic form;

- [NEW] **`wapt-scanpackages`**: added option to provide proxy;

- [UPD] return **OK** by default in package's audit skeleton;
- [IMP] waptconsole cosmetic: minheight 18 pixels for grid headers
- [FIX] waptserver database model: bad default datatype in `model.py`

for `created_by` and `updated_by` (were not used until now);

- [FIX] `ensure_unicode` for *`.msi`* output: try *cp850*

before *utf16* to avoid Chinese garbage in run output;

- [NEW] added `connected_users` to `hosts_for_package` provider;
- [FIX] use **`win32api`** to get local connected IPV4 IP address

instead of socket module. In some cases, socket can't retrieve the IP;

- [FIX] **`wapt-get unregister`** command not working properly;
- [NEW] Waptselfservice: added option in `wapt-get.ini`

to disable unfiltered packages view of local admin;

- [IMP] Waptselfservice: 4K improvements;
- [FIX] Waptselfservice:
- packages *restricted* were shown in selfservice / now corrected;
- if the repo have no packages segmentation error / now corrected;
- if the repo have changed segmentation error / now corrected;

(hash f153fab4)

- [NEW] added **`unregister`** action to wapt-get;
- [UPD] improvements with the alt logo in the self-service;
- [UPD] use version to build the package name of unit, groups

and profile type package, like for base packages;

- [UPD] added logs to **`uwsgi`**;
- [FIX] bugfixes with the icons of the app self-service;
- [FIX] bugfixes with the logos in the self-service;
- [UPD] waptexit: don't cancel tasks on CloseQuery;
- [UPD] patch `server.py` earlier to avoid *execute cannot be used

while an asynchronous query is underway*;

- [FIX] fix waptexit doing nothing if `allow_cancel_upgrade` = **0**

and `waptexit_disable_upgrade` = **0**;

- [FIX] fix issue with merge of wsus rules (can cause memory errors

if more than one wsus package is applied on a host) (**Enterprise** only);

- [FIX] fix wua auto `install_scheduling` issue;
- [FIX] waptexit: add a watchdog to workaround

some cases where it hangs (threading issue ?);

(hash da870a2c)

- [IMP] wapt self service application is now fully usable.

It is available in `<wapt>waptself.exe`;

- [ADD] option to set a random UUID instead of BIOS UUID at setup.

This is to workaround for bugged BIOS with duplicated ids;

- [IMP] better Sphinxdocs for WAPT Libraries;
- [UPD] behavior change: Use computer FQDN from tcpip registry entry

(first NV Hostname key) then fixed domain then DHCP;

- [FIX] inverted Zip and signature steps in package build operations

to workaround issue with Bad Magic Number when signing already zipped big packages;

- [NEW] Add `use_ad_groups` wapt-get `[global]` parameter to activate groups

from AD (this is a time consuming task, so better not activate it...);

- [FIX] appendprofile infinite loop during setup;
- [FIX] read forced uuid from `wapt-get.ini` earlier to avoid loading

a bad host certificate in memory if changing from bios uuid to forced uuid;

- [FIX] setting `use_random_uuid` in `waptagent.iss`;
- [FIX] waptstarter setup: force deactivate server, hostpackages;
- [FIX] include waptself in waptstarter, don't include innosetup in waptstarter;
- [FIX] `ensure_unicode`: add *utf16* decoding test before *cp850*;
- [FIX] add `ensure_unicode` for tasks logs to avoid unicode decode errors

in **`get_tasks_status`** callback;

- [NEW] host status: add `boot_count` attribute;
- [FIX] fix potential float / unicode error when scanning windows updates

(**Enterprise** only);

- [FIX] handles properly excluded files in package signatures;
- [FIX] waptexit: avoid some work after checking if waptservice is running

if it is not running;

- [FIX] a case where WAPTLocalJsonGet could loop forever if auth fails;
- [FIX] `setup.pyc` in `manifest` but not in zipped package:
- exclude exactly [”*.svn*”,”*.git*”,

“*.gitignore*”,”`setup.pyc`”] when signing and zipping;

- **`inc_build`** before signing;
- [UPD] add `use_ad_groups` setting in waptagent build.

Default to *False* (**Enterprise** only);

- [FIX] better detection of `waptbasedir` for `python27.dll` loading;
- [FIX] allow to sign source package directory to workaround a bug

in python zipfile (bad magic number);

- [NEW] added a `htpasswd` password file method for restricted access

to only **`add_host`** method:

allows **`add_host`** if provided host certificate is already signed by server and content can be verified;

- [FIX] **`wapt-get.exe`** crash with « can not load... «

when python 3.7 is installed from MS store;

- [FIX] load `private_dir` conf parameter earlier;
- [UPD] put a *rnd-* in front of randomly generated uuid;

added a checkbox to use random uuid (if not already defined in `wapt-get.ini`);

- [UPD] SSL CA certifi library;
- [IMP] utf8 decode user /password in localservice authentication;
- [UPD] allow authentication on local waptservice with token;
- [NEW] filter packages on hosts based on the `valid_from`

and `valid_until` control attributes;

force update sooner if `valid_from` or `valid_until` or `forced_install_on` is sooner than regular planned `update_period`;

- [FIX] events reporting from service tasks;
- [FIX] **`waptexit`** not closing of writing for running tasks

but auto upgrade has been disabled;

- [ADD] added `waptexit_disable_upgrade` option to **`waptexit`**

to remove the triggering of upgrade from waptexit, but keep the waiting for pending and running tasks:

"running_tasks" key in waptservice checkupgrades.json. Was not reflecting an up to date state;

- [NEW] add new packages attributes: `name`, `valid_from`,

`valid_until`, `forced_install_on`;

- [FIX] regression on *profile* packages not taken in account;

(hash 38e08433)

- [FIX] **`waptexit`** not closing if waiting for running tasks

but auto upgrade has been disabled;

- [FIX] events reporting from service's tasks;
- [ADD]] new packages attributes: `name`, `valid_from`, `valid_until`,

`forced_install_on`;

- [ADD] `waptexit_disable_upgrade` option to **`waptexit`** to remove

the triggering of upgrade from waptexit, but keep the waiting for pending and running tasks;

- [IMP] added `running_tasks` key in waptservice checkupgrades.json.

Was not reflecting an up to date state.

- [IMP] waptself:
- early support of high DPI;
- loading of icons in the background;

(hash 5b6851ae)

- [FIX] takes *profile* packages (AD based groups)

into account (**Enterprise** only)

(hash 4be40c534c4627)

- [FIX]] regression on waptdeploy unable to read current `waptversion`

from registry;

- [FIX] be more tolerant to broken or inexistent *wmi* layer

(for waptconsole on **`wine`** for example);

(hash 95a146c002)

- [IMP] **`waptself.exe`** preview application updated.

Loads icons in the background.

Known issues:

- does not work with repositories behind proxies and client side auth;
- https server certificate is not checked when downloading icons);
- High DPI not handled properly;
- Cosmetic and ergonomic improvements still to come;
- [IMP] waptserver setup on windows: open port 80 on firewall in addition to 443;
- [IMP] waptserver on Debian. add *www-data* group to wapt user

even if user wapt already exists;

- [IMP] waptserver on CentOS. add waptwua directory

to SELinux `httpd_sys_content_t` context;

- [FIX] waptserver client auth: comment out `ssl_client_certificate`

and `ssl_verify_client`;

By default because old client's certificate does not have proper `clientAuth` attribute (error http 400);

- [FIX] problem accessing to 32bit uninstall registry view from 32bit wapt

on Windows server 2003 x64 and Windows server 2008 x64:

it looks like it is not advisable to try to access the virtual Wow6432Node virtual node with disabled redirection;

- [FIX] setuphelpers `installed_softwares` regular expression search on name;

https://github.com/tranquilit/WAPT/issues/7

- [IMP] waptservice: for planned periodic upgrade, use single WaptUpgrade task

like the one used in websocket;

- [IMP] waptexit: cancel all tasks if closing waptexit form;
- [FIX] wapt-get: wapt-get service mode with events:

refactor using uWAPTPollThreads;

- [FIX] **veyon** cli executable name updated;
- [IMP] wapt-get: check *CN* and *subjectAltNames* in lowercase

for **enable-check-certificate** action;

(todo: doesn't take wildcard in account)

(hash 5ef3487)

- upgrade **urllib3** to 1.24.2 for

CVE-2019-11324 (high severity);

- upgrade **jinja2** to 2.10.1 for

CVE-2019-10906;

- [NEW] Wapt self service application preview;
- [IMP] propose to copy the newly created CA certificate

to ssl local service dir, and restart waptservice. Useful for first time use;

- [FIX] `sign_needed` for wapt-signpackages.py;
- [FIX] missing *StoreDownload* table create;
- [FIX] bug in fallback `package_uuid` calculation.

It didn't include the version;

(hash 4cdcaa06c83b)

- [UPD] handling of *subjectAltName* attribute for https server certificates

checks in waptconsole (useful when certificate is a multi hostname commercial certificate). Before, only CN was checked against host's name;

- [UPD] client certificate auth for waptconsole;
- [UPD] versioning of wapt includes now the Git revision count;
- [FIX] replace openssl command line call with waptcrypto call

to create tls certificate on linux server wapt install;

- [FIX] add dnsname *subjectAltName* extension

to self signed waptserver certificate on linux wapt nginx server configuration;

- [FIX] pkcs12 export;
- [NEW] handling of *SubjectAlternativeName* in certificates

for server X509 certificate check in addition to CN:

Added a *SubjectAltName* when creating self signed certificate on linux wapt nginx server in postconf;

For old installation, the certificate is not updated. It should be done manually;

- [FIX] fix **`check_install`** returning additional packages

to install which are already installed (when private repository is using `locale` or `maturities`):
Added missing attributes in waptdb.installed_matching;

- [NEW] added client certificate path and client private key path

for waptconsole access to client side ssl auth protected servers;

- [FIX] fix regression with **`wapt-get edit <package>`**:

made `filter_on_host_cap` a global property of Wapt class instead of a function parameter;

- [FIX] regression if there are spaces in OU name.

Console was stripping space for https://roundup.tranquil.it/wapt/issue911 and https://roundup.tranquil.it/wapt/issue908;

- [IMP] allow "0"..”9”, “A”..”Z”, “a”..”z”, “-“,” _”,”=”,”~”,”.” in package names

for OU packages. Replaces space with ~ in package names and “,” with “_”;

- [IMP] make sure we have a proper package name in packages edit dialogs;
- [IMP] waptservice config: allow `waptupdate_task_period` to be empty

in `wapt-get.ini` to disable it in waptservice;

- [FIX] waptutils: fix regression on wget() if user-agent is overridden;
- [FIX] waptwua: fix an error in install progress % reporting for wua updates;
- [IMP] wapttray: refactor tray for consistency.

Makes use of *uwaptpollthreads* classes;

- [IMP] waptexit: some changes to try to fix cases

when it does not close automatically;

- [IMP] build: add git Revcount (commit count) to exe metadata;
- [FIX] waptconsole: fix hosts for package grid not refreshed if not focused;
- [FIX] internal: use synapse httpsend for waptexit / wapt-get / wapttray

local service http queries to workaround auth retry problems with **`indy`**;

- [ADD] **`wapt-get.exe`**: added `--locales`

to override temporarily locales form `wapt-get.ini`;

- [ADD] **`wapt-get.exe`**: added *WaptServiceUser*

and *WaptServicePassword* / *WaptServicePassword64* command line params:
fix timeout checking in checkopenport;

- [ADD] core: added logs for self-service auth;
- [ADD] waptservice: added /keywords.json service action;
- [ADD] waptservice: added filter keywords (csv) on packages.json provider;
- [IMP] waptconsole: replace tri-state checkbox by a radio group

for wua enabled setting in *create waptagent* dialog;

- [IMP] waptservice local webservice: temporary workaround

to avoid costly icons retrieval in local service;

- [FIX] simplify `installed_wapt_version` in waptupgrade package

to avoid potential install issues;

- [IMP] waptconsole layout: anchors for running task memo;
- [FIX] Makefullyvisible for main form:

avoid forms outside the visible area when disconnecting a second display;

- [FIX] layout of tasks panel for Windows 10;
- [FIX] add `token_lifetime` server side

(instead of using clockskew for token duration);

- [UPD] default unit **days** instead of **minutes**

for wua scan download install and install_delay;

- [ADD] optional export of key and certificate as `PKCS12` file

in *create key* dialog. (to check SSL client auth in browsers...);

- [FIX] winsetup.py fix for backslashes in **`nginx`**;
- [FIX] wapt-get json output / flush error;
- [IMP] cache `host_certificate_fingerprint` and issuer id in local db

so that we don't need to read private directory to get `host_capabilities`. It allows to use **`wapt-get list-upgrade`** as normal user;

- [UPD] don't make DNS query in waptconsole Login / waptconfig

to avoid DNS timeout if domain dns server is not reachable;

- [FIX] warning message introduced in previous revision

when adding a new ini config on login (**Enterprise** only);

- [FIX] waptwua: handles redirect for wsusscn2 head request

(**Enterprise** only);

- [UPD] Report only 3 members on the `wapt_version` capability attribute;
- [IMP] core: refactor WaptUpgrade task: check task to append

and then append them to tasks queue in WaptUpgrade.run instead of doing it in caller code. Avoid timeout when upgrading;

- [IMP] core: self service rules refactoring;
- [IMP] core: notify server when audit on waptupgrade;
- [IMP] core: fix `update_status` not working

when old packages have no `persistent_dir` in db;

- [IMP] core: tasks, events waptservice action: timeout in milliseconds

instead of seconds for consistency;

(hash 92ccb177d5c)

- [FIX] waptconsole: use repo specific ca bundle

to check remote repo server certificate (different from main wapt repo);

- [FIX] waptconsole / hosts for packages: fixed `F5` to do a local refresh;
- [FIX] improved update performance with repositories with a lot of packages;
- [FIX] improved wapttray reporting:

fix faulty inverted logic for `notify_user` parameter;

- [FIX] waptconsole: fixed bad filtering of hosts for package

(**Enterprise** only);

- [FIX] waptexit: fixed waptexit closes even if Running task

if no pending task / no pending updates;

- [FIX] waptexit: fixed potential case where waptexit remains running

with high cpu load;

- [FIX] waptconsole: fixed HostsForPackage grid not filtered properly

(was unproperly using Search expr from first page);

- [FIX] waptservice: None has no `check_install_is_running` error

at waptservice startup;

- [FIX] core: set `persistent_dir` and `persistent_source_dir` attributes

on setup module for install_wapt;

- [FIX] core: fixed bug in guessed `persistent_dir` for dev mode;
- [FIX] core: fixed error resetting status of stuck processes

in local db (check_install_running);

- [FIX] waptservice: trap error setting runstatus in db in tasks manager loop:

Don't send runstatus to server each time it is set;

- [UPD] core: define explicitly the `private_dir` of Wapt object;
- [UPD] server: don't refuse to provide authtoken if FQDN has changed

(this does not introduce specific risk as request is signed against UUID);

- [UPD] core: if `package_uuid` attribute is not set

in package's `control` (old wapt), it is set to a reproductible hash when package is appended to local waptdb so we can use it to lookup packages faster (dict);

- [NEW] waptconsole: added audit scheduling setup

in waptagent dialog (**Enterprise** only):

added `set_waptaudit_task_period` in innosetup installers;

- [IMP] setuphelpers: add win32_displays to default wmi keys for report;
- [IMP] server setup: create X509 certificate / RSA key

for hosts ssl certificate signing and authentication during setup of server;

- [IMP] waptexit: add sizeable border and icons;

- [IMP] show progress of long tasks;
- [IMP] waptservice: process update of packages as a task instead of waiting

for its completion when upgrading (to avoid timeout when running upgrade waptservice task):

added `update_packages` optional (default True) parameter for upgrade waptservice action;

- [NEW] added audit scheduling setup in waptagent compilation dialog

(**Enterprise** only);

- [NEW] setuphelpers: added `get_local_profiles` setuphelpers;
- [IMP] waptserver: don't refuse to provide authtoken

for websockets auth if FQDN has changed;

- [IMP] flush stdout before sending status to waptserver;
- [IMP] waptcrypto handle alternative object names in

CSR build;

- [IMP] wapt-get: `--force` option on **`wapt-get.exe`** service mode;
- [NEW] use client side authentication for waptwua too;
- [CHANGE] server setup: nginx windows config: relocate logs and pid;
- [ADD] added conditional client side ssl auth in nginx config;
- [CHANGE] waptconsole: refactor wget, wgets WaptRemoteRepo WaptServer

to use requests.Session object to handle specific ssl client auth and proxies:

Be sure to set privateKey password dialog callback to decrypt client side ssl auth key;

- [IMP] waptcrypto: added waptcrypto.is_pem_key_encrypted;
- [IMP] waptconsole: make sure waptagent window is fully visible;
- [IMP] waptconsole: make sure Right click select row on all grids;
- [ADD] waptconsole: import from remote repo: add certificate

and key for client side authentication;

(hash ec8aa25ef)

- [UPD] upgraded **`OpenSSL`** dlls to 1.0.2r

for https://www.cert.ssi.gouv.fr/avis/CERTFR-2019-AVI-080/ (moderate risk);

- [IMP] much reworked wizard pages embedded in **`waptserversetup.exe`**

windows server installer. Install of waptserver on Windows is easy again:

- register server as a client of waptserver;
- create new key / certificate pair;
- build waptagent.exe and waptupgrade package;
- configure package prefix;
- [NEW] if client certificate signing is enabled on waptserver

(waptserver.ini config), the server will sign a CSR for the client when the client is first registered. See *Configurer l'authentification par Certificat sur le Client*.

- [NEW] wapt-get: added new command create-keycert to create a pair

of RSA key / x509 certificate in batch mode. Self signed or signed with a CA key/cert:

(options are case sensitive...)

- option /CommonName: CN to embed in certificate;
- options /Email, /Country, /Locality, /Organization,

/OrgUnit: additional attributes to embed in certificate;

- option /PrivateKeyPassword: specify the password

for private key in clear text form;

- option /PrivateKeyPassword64: specify the password for private key

in base64 encoding form;

- option /NoPrivateKeyPassword: ask to create

or use an unencrypted RSA private key;

- option /CA = **1** (or 0)): create a certification authority

certificate if 1 (default to 1);

- option /CodeSigning = **1** (or 0)): create a code signing

certificate if 1 (default to 1);

- option /ClientAuth = **1** (or 0): create a certificate

for authenticating a client on a https server with ssl auth. (default to 1);

- option /CAKeyFilename: path to CA private key to use for signing

the new certificate (defaults to %LOCALAPPDATA%waptconsolewaptconsole.ini [global] default_ca_key_path setting);

- option /CACertFilename: path to CA certificate to use for signing

the new certificate (defaults to %LOCALAPPDATA%waptconsolewaptconsole.ini [global] default_ca_cert_path setting);

- option /CAKeyPassword: specify the password for CA private key

in clear text form to use for signing the new certificate (no default);

- option /CAKeyPassword64: specify the password for CA private key

in base64 encoding form to use for signing the new certificate (no default);

- option /NoCAKeyPassword: specify that the CA private to use

for signing the new certificate is unencrypted;

- option /EnrollNewCert: copy the newly created certificate

in <wapt>ssl to be taken in account as an authorized packages signer certificate;

- option /SetAsDefaultPersonalCert: set personal_certificate_path

in configuration inifile [global] section (default %LOCALAPPDATA%waptconsolewaptconsole.ini);

- [NEW] wapt-get: added new commands `build-waptagent`

to compile a customized waptagent in batch mode:

- copy **`waptagent.exe`** and pre-waptupgrade locally

(if not `/DeployWaptAgentLocally`, upload to server with https);

- option `/DeployWaptAgentLocally`: copy the newly built **`waptagent.exe`**

and prefix-waptupgrade_xxx.wapt to local server WAPT repository directory `.\waptserver\repository\wapt\`;

- [NEW] `wapt-get register`: added options for easy configuration of wapt

when registering:

- `--pin-server-cert`: pin the server certificate.

(check that CN of certificate matches hostname of server and repo);

- `--wapt-server-url`: set `wapt_server` setting in `wapt-get.ini`;
- `--wapt-repo-url`: set `repo_url` setting in `wapt-get.ini`.

(if not provided, and there is not `repo_url` set in `wapt-get.ini`, extrapolate `repo_url` from `wapt_server url`);

- [NEW] wapt-get: added check-valid-codesigning-cert /

CheckPersonalCertificateIsCodeSigning action;

- python libraries updates
- **`cryptography from 2.3.1`** -> **`cryptography 2.5.0`**;
- **`pyOpenSSL 18.0.0`** -> **`pyOpenSSL 19.0.0`**;
- [FIX] don't reset host.server_uuid in server db

when host disconnect from websocket. Set host.server_uuid in server db when host gets a token;

- [FIX] modify isAdminLoggedIn to try to fix cases

when we are admin but function return false;

- [FIX] ensure valid package name in package wizard (issue959);
- [FIX] regression when using python cryptography 2.4.2 openssl bindings

for windows XP agent (openssl bindings of the python cryptopgraphy default WHL >= 2.5 does not work on Windows XP);

- [FIX] trap exception when creating db tables from scratch fails,

allowing upgrade of structure;

- [FIX] reduce the risk of *database is locked* error;
- [FIX] deprecation warning for verifier and signer when checking crl signature;
- [FIX] `persistent_dir` calculation in package's call_setup_hook

when package_uuid is None in local wapt DB (for clients migrated from pre 1.7 wapt, error None has no len() in audit log);

- [FIX] regression don't try to use host_certificate / key

for client side ssl authentication if they are not accessible;

- [IMP] define proxies for crl download in **`wapt-get scan-packages`**;
- [IMP] fixed bad normalization action icon;

- [IMP] paste from clipboard action available in most packages editing grid;
- [IMP] propose to define package root dev path, package prefix, waptagent

or new private key / certificate when launching waptconsole;

- [IMP] remove the need to define waptdev directory

when editing *groups* / *profiles* / *wua packages* / *self-service* packages;

- [IMP] grid columns translations in French;
- [IMP] waptexit responsiveness improvements. Events check thread

and tasks check thread are now separated.

- [NEW] added ClientAuth checkbox when building certificate in waptconsole;
- [NEW] added `--quiet -q` option to `postconf.py`
- [MISC] add an example of client side cert auth
- [ADD] added clientAuth extended usage to x509 certificates (default True)

for https client auth using personal certificate;

- [NEW] use of ssl client cert and key in waptconsole for server authentication;
- [FIX] ssl client certificate auth not taken in account

for server api and host repository;

- [ADD] added `is_client_auth` property for certificates;
- default *None* for `is_client_auth` certificate /

CSR build;

- don't fallback to host's client certificate authentication

if it is not clientAuth capable (if so, http error 400);

- [MISC] waptcrypto: added SSLPKCS12 to encapsulate

pcks#12 key / certificate in certificate store;

- [MISC] added splitter for log memo in Packages for hosts panel;
- [FIX] store fixes;
- [FIX] be tolerant when no `persistent_dir` in *waptwua* packages;
- min wapt version 1.7.3 for self service packages and *waptwua* packages,
- [FIX] WsusUpdates has no attribute `downloaded`;

(hash 373f7d92)

- [FIX]] softs normalization dialog closed when typing F key

(**Enterprise** only);

- [IMP] include waptwua in nginx wapt server windows locations

(**Enterprise** only);

- [FIX] force option from service or websockets not being taken in account

in **`install_msi_if_needed`** or **`install_exe_if_needed`**;

- [IMP] improved win updates reporting (uninstall behavior)

(**Enterprise** only);

- [ADD] added uninstall action for winupdates in waptconsole

(**Enterprise** only);

- [FIX] reporting from dmi « size type » fields with non integer content

(**Enterprise** only);

- [IMP] waptexit: allow minimize button;
- [IMP] waptexit: layout changes;
- [IMP] AD Auth: less restrictive on user name sanity check

(**Enterprise** only);

- [IMP] handling of updates of data for winupdates

with additional download urls (**Enterprise** only);

- [ADD] added some additional info fields to WsusUpdates table

(**Enterprise** only);

- [ADD] added filename to Packages table for reporting and store usage

(**Enterprise** only);

- [ADD] added uninstall win updates to waptconsole (**Enterprise** only);
- [ADD] added windows updates uninstall task capabilities (**Enterprise** only);
- [ADD] added filename to Packages table;
- [IMP] increased default clockskew tolerance for client socket io;
- [FIX] regression in package filenames (missing _);
- [FIX] mismatch for waptconsole `[global] waptwua_enabled` setting;
- [FIX] default waptconsole *EnableWaptWUAFeatures* to True;
- [FIX] waptexit: fixed install of and empty list of Windows Updates

(**Enterprise** only);

- [FIX] wapt-get.exe WaptWUA commands: fixed import of waptwua client module

for waptwua-scan download install (**Enterprise** only);

- [FIX] `install_delay` for Windows Updates stored

as a time_delta in waptdb (**Enterprise** only);

- [ADD] versioning on group packages filenames;
- [ADD] button to create AD Host profiles

(package automatically installed/removed based on AD Grouo memberships)

- [IMP] reduce wapttray notifications occurrences.

`notify_user` = **0** per default

- [FIX] waptexit: fixed details panel does not show the pending packages

to install;

- [FIX] always install the missing dependencies in install

(even if upgrade action should have queued dependencies installs before) for some corner known cases;

- [FIX] get server certificate chain popup action when building the waptagent;
- [ADD] action to create a key / certificate in waptconsole conf;
- [IMP] hide inactive / disabled WaptWUA actions in Host popup menu;
- [ADD] checkbox to display newest only for groups;
- [ADD] waptconsole config parameter `licences_directory`

to specify the location (directory) of licenses (**Enterprise** only);

- [IMP] waptagent build dialog: Removed the *Append host's profiles*

option;

- [IMP] remove waptenterprise directory if waptsetup community is deployed

over a waptenterprise edition;

- [IMP] Core:
- better support for `locales`, `maturities` and `architecture`

packages filtering;

- [NEW] Self service rule packages (**Enterprise** only):
- Package to define which packages can be installed / remove

for groups of users;

- WAPT Windows Updates rules packages (**Enterprise** only);
- [NEW] package to define which Windows Updates are allowed / forbidden

to be deployed by Wapt WUA agents;

- **waptagent** build:
- [ADD] option for `use_fqdn_as_uuid` when building waptagent.exe;
- [ADD] option to define the profile package to be deployed

upon Wapt install on hosts;

- [ADD] options to enable WaptWUA (Windows updates with Wapt)

(**Enterprise** only);

- Host Profile packages (**Enterprise** only):
- [IMP] specific packages (like Group packages) which are installed

or removed depending of `wapt-get.ini [global] host_profiles` ini key;

- [NEW] if a *profile* package name matches Computer's AD Groups,

it is deployed automatically;

- Reporting (**Enterprise** only):
- [NEW] import / export queries as json files;

- [IMP] softwares names normalization as a separate dialog;
- **waptexit**:
- [IMP] reworked to make it more robust;
- [IMP] takes in account packages to remove;
- [IMP] takes in account Wapt WUA Updates (**Enterprise** only):
- command line switch: /install_wua_updates;
- wapt-get.ini setting: [waptwua] `install_at_shutdown` = **1**;
- checkbox in waptexit to skip install of Windows Updates;
- **waptconsole** Custom commands:
- [NEW] ability to define custom popupmenu commands which are launched

for the selection of hosts. Custom variables {uid};

- Other improvements:
- [IMP] French translations fixes;
- [NEW] Reporting (**Enterprise** only):
- basic SQL reporting capability;
- duplicate action / copy paste for reporting queries;
- [ADD] setuphelpers: added helpers `processes_for_file`

and `get_computer_domain`;

- **`python 2.7.15`** on Windows;
- **`openssl-1.0.2p`**;
- upgraded to **`python-requests 2.20.0`** (Security Fix);
- [IMP] don't refresh GridHostsForPackage if not needed

(**Enterprise** only);

- [IMP] don't add a newline to log text output for LogOutput;
- [IMP] improved handling of update_host_data hashes to reduce amount of data

sent to server on each **`update_server_status`**;

- [IMP] set python27.dll path in wapt-get and **`waptconsole.exe`**

(fix cases with multiple python installations);

- [FIX] removal of packages when upgrading host via websockets;
- [IMP] don't get host capabilities if not needed when updating;
- [IMP] don't check package control signatures in wapt-get

when loading list of packages for development tasks;

- [IMP] Moved static waptserver assets to a /static root

split base.html and index.html templates for blueprints;

- [FIX] selective pending wua install or downloads (**Enterprise** only);

- [FIX] wua updates filter logic (**Enterprise** only);
- [IMP] uninstall host packages if `use_hostpackages` is set to false:
- add a forced update in the task loop

when host capabilities have been changed;

- include `use_host_packages` and `host_profiles` in host's capabilities;
- [FIX] regression not removing implicit packages.
- [IMP] more tolerant to unicode errors in **`update_host_data`**

to avoid hiding actual exception behind an encoding exception.

- [FIX] order of columns not kept when exporting reports (**Enterprise** only)
- [IMP] `install_msi_if_needed`, `install_exe_if_needed`:

check if `killbefore` is not empty or None

- [IMP] changed tasks's progress and runstatus to property
- [FIX] Audit aborted due to exception: "NoneType" object

is not iterable (**Enterprise** only)

- [ADD] setuphelpers: Add `get_app_path` and `get_app_install_location`
- add fix_wmi procedure to re-register WMI on broken machines
- some wmi fallbacks to avoid unregistered machines when WMI is broken on them
- [ADD] Online wua scans (**Enterprise** only)
- [ADD] random `package_uuid` when signing a package metadata

which could be used later as a primary key:

- creates a random `package_uuid` when installing in DEV mode;
- creates a random `package_uuid` when installing

a package without `package_uuid`;

- [IMP] moved and renamed EnsureWUAUServRunning to setuphelpers;
- [ADD] `pending_reboot_reasons` to inventory;
- [IMP] display package version for missing packages;
- [ADD] **`wapt-get sign-packages`**: added setting `maturity`

and inc version in sign-packages action;

- [ADD] WindowsUpdates's host History grid below WindowsUpdate grid

(**Enterprise** only);

- [IMP] store Host Windows update history in server DB (**Enterprise** only);
- [IMP] keep selected or focused rows in grids;
- [IMP] updates Packages table when uploading a Package / Group.

This table is meant mainly for reporting purpose;

- [IMP] disable indexes for some BinaryJson fields;

- [FIX] windows update `install_date` reporting (**Enterprise** only);
- [ADD] checkbox to enable `use_fqdn_as_uuid`

when building **`waptagent.exe`**;

- [IMP] change default value for `upgrade_only_if_not_process_running`;
- [IMP] changed naming of organizational *unit* packages to remove ambiguity

with comma in package name and comma to describe the list of packages depends / conflicts:

Replace "," with "_" when editing package (**Enterprise** only);

- [ADD] waptexit: added priorities and `only_if_not_process_running`

command line switches;

- [IMP] waptupgrade: changed `windows_version` and Version;
- [ADD] setuphelpers `windows_version`: added `members_count`;
- [IMP] waptutils.Version: strip members to `members_count` if not *None*;
- [ADD] control attributes editor keywords license homepage `package_uuid`

to local waptservice db;

- [ADD] short fingerprint to repr of SSLCertificate;
- [IMP] be sure password gui is visible even if parent window is not;
- [ADD] gui for private key password dialog if `--use-ggui`;
- [ADD] `--use-gui` to **`wapt-get.exe`** command line arg

to force use of waptguihelper for server credentials when registering;

This is a bugfix release for 1.6.2.5:

- [FIX] *waptexit*: changed the default value of

upgrade_only_if_not_process_running parameter to *False* instead of *True*:

if *upgrade_only_if_not_process_running* is *True*, the install tasks for packages with running processes (*impacted_process*) are skipped;

if *upgrade_only_if_not_process_running* is *False*, the install tasks for packages with running processes may impact the user if the installer kills the running processes;

- [FIX] *waptwua*: take in account Windows Updates *RevisionNumber* attribute

to identify uniquely an Update in addition to UpdateID field (**Enterprise** only). This fixes the 404 error when downloading missing windows updates on a client.

This is a bugfix release for 1.6.2.5:

- [FIX] WAPTServer Enterprise on Windows: added proper upgrade path from

`PostgreSQL 9.4` (used in WAPT 1.5) to **`PostgreSQL 9.6`** which is required for WAPT-Windows Update:

- new database binary and data directory path are suffixed with -9.6;
- old data is suffixed with -old after migration;
- [FIX] upgrade script for **`MongoDB`** upgrade (WAPT 1.3)

to **`PostgreSQL`** used since WAPT 1.5;

- [FIX] regression on WMI / DMI inventory which may be not properly

sent back to the server;

[NEW] Main new features if you are coming from 1.5:

- per package *Audit* feature (**Enterprise** only);
- *WAPT managed Windows Updates* tech preview (**Enterprise** only);
- wizards to guide post configuration

of Windows server and first use of **waptconsole**;

- **waptconsole/** private repo page: added a grid which shows

the computers where the selected package is installed;

It includes numerous changes over the 1.5.1.26 version.

- [NEW] per package audit feature:
- def audit() hook function to add into package's `setup.py`.

By default, check *uninstall key* presence in registry:

- **wapt-get audit**;
- **wapt-get -S audit**;
- **wapt-get audit <packagename>**;
- right click in waptconsole on machines or installed

packages/ Audit package;

- synthetic audit status for each machine;
- for each installed package: *last_audit_status*, *last_audit_on*,

last_audit_output, *next_audit_on*;

- scheduled globally with `wapt-get.ini` parameter `[global]`:

```
waptaudit_task_period = 4h
```

or in package's `control` file:

```
audit_schedule = 1d
```

- audit log displayed in **waptconsole** below installed package grid

if *Audit Status* column is focused;

- [UPD] updated python modules
- [IMP] build with **Lazarus 1.8.2** instead of **CodeTyphon 2.8**

for the Windows executables:

- better strings encoding handling and easier to setup for the development;
- **PostgreSQL 9.6** is required for WAPT WUA tech preview

(Debian Jessie not supported);

- WAPT 1.6 includes one more security layer in the agent to server connection.

After server upgrade, the client desktops won't be able to connect to the server as long as they have not been upgraded themselves. If you require to be able to remotely manage the WAPT agent while the agent has not yet been upgraded, it is necessary to set *allow_unauthenticated_connect* to *True* in `waptserver.ini`;

- [FIX] add AD Groups as Hosts dependencies in **`waptconsole`**;
- [FIX] remove image on reachable column if no status has been sent yet;
- [FIX] Organizational Units WAPT packages not being installed

when there are spaces in DN;

- [FIX] Operational error when host are trying

to reconnect but are not registered;

- [FIX] fill in *created_on* db fields on win updates data;
- [IMP] debian server postinst: remove old `pyc` files;
- [IMP] Improved WAPT console setup Wizard;
- [ADD] *allow_unauthenticated_connect* defaults to

allow_unauthenticated_registration if it is not explicitly set in `waptserver.ini` file (This will ease migration from 1.5 to 1.6);

- [IMP] `Escape` key on password edit of login moves focus

to configuration combo;

- [IMP] PackageEntry.asrequirement(): removed space between package name

and version specification;

- [IMP] missing *install_date* in *insert_many* for some updates;
- [ADD] add force arg for WAPTUpdateServerStatus action;
- [IMP] don't includes `setup.py` in initial host's

packages inventory, and full inventory;

- [IMP] allow to use installed **`waptdeploy.exe`**

without retry/ignore dialog;

- [IMP] be sure error is reported properly in **`socketio`**;
- [IMP] added *package_uuid* and homepage package attributes;
- [IMP] added installed on columns for host wsus updates;
- [FIX] WUA grid layout saving;
- **`PostgreSQL 9.6`** is required for WAPT WUA tech preview

(Debian Jessie not supported);

- the authentication of client connections to the WAPT websockets server

is not compatible with pre-1.6.2 wapt clients. During migration, if you want to keep the connection with clients, you have to disable the authentication with the parameter: *allow_unauthenticated_connect* = 0 in server's configuration file `waptserver.ini`. When all clients have migrated, this can be removed;

- [NEW] wizard for the initial configuration of **waptserver** on Windows;
- [ADD] wizard for the initial configuration of **waptconsole**

connection parameters;

- [ADD] **Enterprise only**: waptconsole/ private repo page: added a grid

which shows the computers where the selected package is installed;

- [NEW] **Enterprise only**: WAPT WUA Windows Updates management

technical preview:

- activate with `waptwua_enabled` = **1** in `wapt-get.ini` file

on the client;

- scan of updates on Windows clients with the IUpdateSearcher Windows API

and the `wsusscan2` cab file from Microsoft;

- additional page in *WAPTconsole* host inventory for

Windows updates status reported (HostWsus model);

- additional page in *WAPTconsole* for the consolidated view

of all updates reported by hosts (WsusUpdates model);

- periodic task on server to check and download newer version

of `wsusscan2` cab file from Microsoft (daemon/ service wapttasks);

- periodic Task on server to download missing windows updates files

as reported by Windows client after scan:

- missing files are downloaded if one of the client should install

it and has not yet a copy in its local windows update cache;

- downloads are logged in *WsusDownloadTasks* model;
- [ADD] field in hosts table to keep the hashes of sent host data,

so that clients can send only what needs to be updated;

- [ADD] *db_port server* config parameter if **posgresql** server

is not running on standard port 5432;

- [ADD] editor optional attribute for package control, used

in *register_windows_uninstall* helper if supplied;

- [IMP] websocket authentication with a timestamped token obtained

from server with client SSL certificate on server with client SSL certificate;

- [IMP] json responses from **waptserver** are gzipped;
- [IMP] forced host uuid;
- [IMP] forced computer AD Organizational unit;
- [IMP] public certs dir;
- [FIX] caching of negative result for certs chain validation;

- [IMP] refactoring of server python modules (*config*, *utils*, *auth*, *app*,

common, *decorators*, *model*, *server*) for the enterprise modularity;

- [FIX] timezone file timestamp handling for http download;
- upgrade to **`peewee 3.4`**;
- upgrade to **`eventlet==0.23.0`**;
- upgrade to **`huey 1.9.1`**;
- **`eventlet 0.20.1`** -> **`eventlet 0.22.1`**;

0.22.1:

- [IMP] event: Event.wait() timeout=None argument to be

compatible with upstream CPython;

- [IMP] greendns: Treat /etc/hosts entries case-insensitive.

Thanks to Ralf Haferkamp;

0.22.0:

- [IMP] dns: reading /etc/hosts raised DeprecationWarning for universal lines

on Python 3.4+. Thanks to Chris Kerr;

- [IMP] green.openssl: Drop OpenSSL.rand support.

Thanks to Haikel Guemar;

- [IMP] green.subprocess: keep CalledProcessError identity.

Thanks to Linbing@github;

- [IMP] greendns: be explicit about expecting bytes from sock.recv.

Thanks to Matt Bennett;

- [IMP] greendns: early socket.timeout was breaking IO retry loops;
- [IMP] GreenSocket.accept does not notify_open.

Thanks to orishoshan;

- [IMP] patcher: set locked RLocks" owner only when patching existing locks.

Thanks to Quan Tian;

- [IMP] patcher: workaround for monotonic « no suitable implementation ».

Thanks to Geoffrey Thomas;

- [IMP] queue: empty except was catching too much;
- [IMP] socket: context manager support.

Thanks to Miguel Grinberg;

- [IMP] support: update **`monotonic 1.3`** (5c0322dc559bf);
- [IMP] support: upgrade bundled to **`dnspython 1.16.0`** (22e9de1d7957e)

https://github.com/eventlet/eventlet/issues/427;

- [FIX] websocket leak when client did not close connection properly.

Thanks to Konstantin Enchant;

- [IMP] websocket: support permessage-deflate extension.

Thanks to Costas Christofi and Peter Kovary;

- [IMP] wsgi: close idle connections (also applies to websockets);
- [IMP] wsgi: deprecated options are one step closer to removal;
- [IMP] wsgi: handle remote connection resets.

Thanks to Stefan Nica;

0.21.0

- [IMP] new timeout error API: .is_timeout=True on exception object.

It's now easy to test if network error is transient and retry is appropriate. Please spread the word and invite other libraries to support this interface;

- [IMP] hubs: use monotonic clock by default (bundled package);

Thanks to Roman Podoliaka and Victor Stinner

- [IMP] dns: EVENTLET_NO_GREENDNS option is back, green is still default;
- [IMP] dns: hosts file was consulted after nameservers;
- [IMP] wsgi: log_output=False was not disabling startup and accepted messages;
- [IMP] greenio: Fixed OSError: [WinError 10038] Socket operation on nonsocket;
- [IMP] dns: EAI_NODATA was removed from RFC3493 and FreeBSD;
- [IMP] green.select: fix mark_as_closed() wrong number of args;
- [NEW] added zipkin tracing to eventlet;
- [IMP] db_pool: proxy Connection.set_isolation_level();
- **`Flask-socketio 2.9.2`** -> **`Flask-socketio 3.0.1`**;
- **`python-engineio 2.0.1`** -> **`python-engineio 2.0.4`**;
- **`python-socketio 1.8.3`** -> **`python-socketio 1.9.0`**;
- upgrade to **`websocket-client 0.47`**;
- [ADD] def audit() optional hook in package is called periodically

to check compliance. Log and status is reported in server DB and displayed in console (**Enterprise**).

- [ADD] WSUS tech preview: based on local Windows update engine and `WSUSSCAN2`

cab Microsoft file. WAPT server act as a caching proxy for updates. Scanning for, downloading and applying Windows updates can be triggered from console on workstations (**Enterprise**). A new wapttasks process is launched on the server to download updates and wsusscan cab from Internet.

- [IMP] better utf8 handling;
- [IMP] **`wapt-get make-template`** from a directory creates a basic installer for portable apps;
- [IMP] wapt-get, waptexit: Removed ZeroMQ message queue on the client,

replaced by simple http long polling to monitor tasks status;

- [IMP] waptconsole: Replaced blocking timer based http polling for tasks

status by threaded http long polling;

- [IMP] waptconsole: Filter hosts on whether current personal certificate signature

is authorized for remote tasks (**Enterprise**). If same server is used for several organizations, it allows to focus on own machines. This supposes that different CA certificates are deployed depending on the client host's organization. In this release, the filtering is not enforced and not cryptographically authenticated;

- [CHANGE] renamed waptservice.py to service.py and waptserver.py to server.py,

activated absolute import for all python sourced absolute import for all python sources;

- [REMOVED] *use_http_proxy_for_template* parameter

(setting is now in `[wapt-templates]` repo);

- [ADD] handling of WUA tasks (Scan, download, apply updates) (**Enterprise**);
- [ADD] handling of auditing tasks;
- [ADD] tasks queue (**Huey**) for the WSUS background tasks

(**Enterprise**);

- [IMP] gzip compression activated on the **nginx** configuration;
- [ADD] option in `wapt-get.ini` to hide some items:
- `hidden_wapttray_actions`: comma separated list of:

LaunchWAPTConsole, *register*, *serviceenable*, *reloadconfig*, *cancelrunningtask*, *cancelalltasks*, *showtasks*, *sessionsetup*, *forceregister*, *localinfo*, *configure*;

- [CHANGE] use long polling instead of **zmq**;
- [IMP] stop/ start/ query waptservice using a thread to avoid gui freeze;
- [FIX] waptguihelper: be sure to load the proper python27.dll;
- [FIX] core: forward *force* argument from console

to `setup.py` install() hook;

- [FIX] overwrite `psproj` package file when editing a package

to fix path to WAPT python virtualenv and add new debug actions;

- [UPD] GUI Binaries are built with **Lazarus 1.8.2 / fpc 3.0.4**

instead of **CodeTyphon 2.8**;

- [UPD] **peewee 3.0.4**;
- [UPD] **eventlet 0.23.0**;
- [UPD] **huey 1.9.1**;
- [UPD] **pywin32** rev 223;
- [UPD] **Flask-socketio 2.9.6**;
- [UPD] **engineio.socket 2.0.4**;
- [UPD] **websocket-client 0.47**;
- [UPD] **pyOpenSSL 17.5.0**;
- [UPD] **request 2.19.1**;

- *unit* type of packages (with AD DN style names) are not well handled

by local WAPT self service, because of commas in name.

- [FIX] av potential cause in wapttray;
- [IMP] buffer LogOuput;
- [FIX] wait task result loop in waptserver;
- [FIX] bad acl on waptservice;
- [FIX] repo timeout not taken in account;
- [FIX] bad parameter for `repo_url` and `[wapt-host]` section;
- [FIX] waptexit AV potential cause;
- [FIX] make isAdmin non blocking as a workaround for false positive checks;
- [FIX] use timeout parameter when importing external package;
- [FIX] pass timeout parameter when importing;
- [FIX] bad `repo_url` config naming;
- [FIX] calc hash when compiling if file does not exist;
- [FIX] repo timeout is float;
- [FIX] custom zip corruption when signing a package with non ascii filenames;
- [FIX] check wapt_db is assigned when rollbacking;
- [IMP] logging in events;
- [FIX] installed packages section is incorrectly reported as *base*

instead of *unit* or *host* in waptconsole;

- [IMP] ensure manual service wua is running when using command line;
- [UPG] Python modules updates:
- upgrade to **`peewee 3.4`**;
- upgrade to **`eventlet==0.23.0`**;
- upgrade to **`huey 1.9.1`**;
- [CHANGE] replace eventprintinfo with LogOutput;
- [ADD] `waptwua_enabled` config parameter;
- [IMP] missing `ensure_list` waptwua_enabled config parameter;
- [IMP] default *waptwua_enabled* to None to avoid wuauserv

service configuration change;

- [ADD] missing columns for window updates;
- [ADD] action in waptconsole to show help on KB;
- [IMP] wapttray cosmetic: hide duplicated separators

in tray popup menu when some actions are hidden;

- [ADD] http_proxy ini setting for the server external download operations;

- [IMP] wapttray: Start and stop WAPTservice using a thread to avoid gui freeze;
- [IMP] Pure FPC PBKDF2 password hash calc for postconf;
- [IMP] refactor server code to share app and socketio instances;
- [FIX] forward the « force » argument (command line and through the websockets)

to the install() setup.py hook;

- [FIX] do not display all missed events at tray startup in wapttray;
- [FIX] no default `audit_period`;
- [REMOVED] **`zeromq`**, replaced by long http polling between wapttray,

wapt-get and waptservice;

- [IMP] revert monkey_patch for server on windows. No reason to exclude thread;
- [ADD] `allow_unauthenticated_connect` server config (default *false*);
- [FIX] CRITICAL update_host failed UnboundLocalError(« local variable "result"

referenced before assignment »,);

- [FIX] https://roundup.tranquil.it/wapt/issue951;
- [FIX] https://forum.tranquil.it/viewtopic.php?f=13&t=1160ix;
- [FIX] https://forum.tranquil.it/viewtopic.php?f=13&t=1160;
- [FIX] `init_workdir.bat`;
- [FIX] returns a token when updating host data for websocket authentication;
- [IMP] rewrite package psproj when editing (to fix wapt basedir paths);
- [FIX] %s -> %d format string for expiration warning message;
- [FIX] host_certificate not found for waptstarter;
- [ADD] some dev build scripts;
- [FIX] zipfile python library bug for packages which contains files

with non-ascii filenames. Signed WAPT packages were corrupted in this case;

- [FIX] deadlocks on server database when simultaneous DB connections

is larger than 100 (default maximum connections configured by default on postgresql);

- [FIX] waptconsole crash on warning message when license

is about to expire (**Enterprise** only);

- [FIX] %s -> %d format string for expiration warning message;
- [FIX] `host_certificate` not found for waptstarter;
- [FIX] waptserversetup.iss to include enterprise modules (**Enterprise**);
- [FIX] download link to waptsetup and waptdeploy

on server index page for Windows;

- **`requests 2.19.1`**;
- **`Rocket 1.2.8`** - Don't try to resurrect connections that timeout.

Increase the timeout ... to decrease the likelihood:

- handle PyPi only supports HTTPS/TLS downloads now;
- fix the problem that when body is empty no terminating;

chunk is sent for chunked encoding.

- avoid sending the terminating chunk in case it's a HEAD request;
- fix the problem that when body is empty no terminating

chunk is sent for chunked encoding;

- explicitly set the log level to warning;
- fix bug « Threadpool grows by negative amount when max_threads = 0 »;
- don't try to resurrect connections that timeout. Increase the timeout

to decrease the likelihood;

- [IMP] waptexit: display a custom PNG logo if one

is created in `%WAPT_HOME%\templates\waptexit-logo.png`;

- [IMP] nssm.exe is signed with Tranquil IT code signing key;
- waptconsole: Add locale and maturity columns in packages status grid;
- waptconsole: wapagent wizard; be sure to get a relative path

when checking cert validity;

- waptsetup: Add /CopyPackagesTrustedCA and /CopyServersTrustedCA command line

parameters to allow deployment of wapt with specific certificates with GPO for wapt without recompiling waptsetup;

Example:

```
C:\tmp\waptdeploy --hash=e17c4eddd45d34000df0cfe64af594438b0c3e1ee9791812516f116d4f4b9fa9
--minversion=1.5.1.23 --waptsetupurl=http://buildbot/~tisadmin/wapt/latest/
waptsetup.exe --setupargs=/CopyPackagesTrustedCA=c:\tmp\tranquilit.crt --setupargs=/
CopyServersTrustedCA=c:\tmp\srvwapt.mydomain.lan.crt --setupargs=/verify_cert=ssl\
server\srvwapt.mydomain.lan.crt --setupargs=/repo_url=https://srvwapt.mydomain.lan/wapt
--setupargs=/waptserver=https://srvwapt.mydomain.lan --setupargs=/DIR=c:\wapt
```

- [FIX] waptconsole: regression introduced in 1.5.1.22. Unable to login if server

has not a FQDN;

- [FIX] setuphelpers: winstartup_info fallback when `COMMON_STARTUP`

folder does not exist, preventing a client to register properly;

- [FIX] version/ revision in wapttray dispkay the git hash instead

of old svn revision number;

- [FIX] waptconsole: update French translation for certs bundle hint;
- [FIX] waptconsole: compare properly packages when number of version

members differs 1.3 -<> 1.3.1 for example;

- [FIX] add Active Directory groups;

- [FIX] newest only with `locale`, `architecture` and `maturity`;
- [FIX] Import from external repository with mixed `locale`,

`architecture` and `maturity`;

- [ADD] `--setupargs` to **`waptdeploy`**;
- [FIX] RPM;
- [FIX] Enterprise build (**Enterprise** only);
- [IMP] different icons for WAPT Community and Enterprise editions;
- [IMP] switch to Community features when no licence instead of aborting

(**Enterprise**);

- some up to date Installed Packages marked as upgradable because

of bad comparison `maturity` None/ maturity;

- [IMP] `depends` and `conflicts` fields of HostsPackagesStatus table limited

to 800 chars -> type changed to ArrayField to handle unlimited number of dependencies;

- [NEW] git python module added as part of WAPT libraries;
- [IMP] list organizational *unit* packages in group package table

(**Enterprise**);

- [FIX] MongoDB to PostgreSQL database upgrade script;
- [FIX] licence/ hosts count/ expiry check (**Enterprise**);
- [FIX] relative path for *verify_cert*;
- When waptserver is searched with DNS SRV query (dnsdomain param),

kerberos register authentication is not working.

- [IMP] multiple languages for description of packages. English, French, German,

Spanish, Polish are handled as a start point. More to be added in the future;

- [IMP] the description columns in waptconsole displays either languages depending

on `language` setting in `waptconsole.ini`. In packages, `description_fr`, `description_en`, etc... have been added;

- [IMP] when renaming hosts, old host package (matching previous host uuid)

is now « removed » instead of forgotten;

- [NEW] Handle AD organizational unit packages (**Enterprise** only;)
- [NEW] package attributes:
- `locale` attribute: A computer can be configured to accept

only packages with a specific locale;

- `maturity` attribute: stores status like *DEV*, *PREPROD*, *PROD*

to describe the level of completion of the package. Computers can be configured to accept packages with specified maturities. Default packages maturity of computer is both the empty one and *PROD*;

- `impacted_process` attribute: csv list of process names which

would be killed before install (**install_msi_if_needed**, **install_exe_if_needed**) and uninstall (by the mean of uninstallkey list). Could be used too in the future for « soft » upgrade remote action which upgrade softwares while they are not running;

WAPTupgrade package:

- [IMP] increased lifetime for upgrade task windows scheduler trigger

for computers which are down for many days when upgrading;

- [ADD] trigger at start of the computer;
- [IMP] display of the list of embedded trusted packages certificates

when building the custom waptagent installer;

- [FIX] handle unicode filepaths for Packages Wizard;
- [IMP] work in progress improvement of unicode handling globally in WAPTconsole;
- [FIX] use proxy if needed for « download and edit » from external repo;
- [FIX] bug in **create_programs_menu_shortcut** and

create_user_programs_menu_shortcut. Shortcuts were created in `startup` and not `startup/programs`.

There is now some additional support for packages localization.

In Package `control` file, the *description_fr*, *description_en*, *description_de*, *description_pl*, *description_es* can be used to give description in respective french, english, german, polish languages.

If not set, the base description is used.

There is a significant internal change on how python libraries are managed inside WAPT. This has implications on the way python scripts are launched. This change is only relevant for peoples launching WAPT processes manually.

We have removed the (not clean) sys.path manipulations inside wapt python scripts sources. The consequence is that all python scripts must be run with prior setting `PYTHONHOME` and `PYTHONPATH` pointing to WAPT home directory (`/opt/wapt` on Linux).

Failing to do so results in scripts claiming that libraries are missing.

On Linux waptserver, libs are now in the default `/opt/wapt/lib/python2.7` location instead of using non standard former one.

- [IMP] WAPT has its own full python environment for libraries,

even when debugging. Before, system wide python27 installation was needed for **PyScripter** to run.

Now, **PyScripter** can be started with a special batch file `waptpyscripter.bat` which sets the environment variables for python (`PYTHONHOME` and `PYTHONPATH`) and run **PyScripter** with python dll path set to wapt own copy.

- [NEW] Command line scripts with proper environment:
- *wapt-serverpostconf* on Linux server to start server postconf.py
- *wapt-scanpackages*
- *wapt-signpackages*
- [NEW] debugging commandline tools which setup python environment

properly before running the python script.py before running the python script:

- to debug waptservice, launch in cmd as admin: *runwaptservice.bat*;
- to debug waptserver, launch in cmd: *runwaptserver.bat*

or under linux: *runwaptserver.sh*;

- to launch **PyScripter** without the need for local

system wide python27 install, run **waptpyscripter.bat**;

- [IMP] Add local wapt-get.ini settings *packages_whitelist*

and *packages_blacklist* to restrict accepted packages from repository based on their package's name;

- [IMP] More detailed reporting off host's repositories configuration

(now includes dnsdomain, proxy, and list of trusted certificates);

- [FIX] fixed display in the Windows task bar of the login window

(to allow in particular the autofill of the password by password managers); waptagent failing to compile if keys/ certificates already exist but the certificate had been removed from `C:\wapt\ssl`;

- [NEW] Handle AD organizational unit packages (Enterprise edition)
- [IMP] Fallback to basic auth when a host is registering on waptserver

if kerberos is enabled but authentication fails.

- [IMP] for **wapt-get.exe**, allow to designate configuration

`wapt-get.ini` file with *–config* option with base name of user waptconsole ini file (without ini extension) instead of full path. Handy when switching between several configurations. Same behavior as for waptconsole. Example:

`wapt-get -c site3 build-upload c:\\waptdev\\test-7zip-wapt`;

- [FIX] Be sure to not loop for ever in websockets retry loop if something

is wrong in host waptserver or websocket configuration.

- [FIX] Update PyScripter project template to use project directory as parameter

for debug actions, and use relative paths for filenames.

- [FIX] incorrect package version comparison. Return True when comparing 1.2-1

to 1.2.1-3 (note: this is not homogeneous with the Version() class behavior. todo: merge both);

- [FIX] waptsetup: register and update must be launched with elevated

privileges. So remove *runasoriginaluser* option.

- [NEW] Introduced attributes target_os and impacted_process for package's

`control` file. They are not yet taken in account.

- [NEW] Introduced machinery to handle X509 client certificates authentication

for repositories and waptserver (specially for public servers);

- [NEW] Introduced classes to generate X509

CRL;

- [UPD] setuphelpers.removetree: Try to remove readonly flag when remove_tree

reach a Access Denied error;

- [FIX] unicode handling in shell startup shortcuts;
- [IMP] waptutils.wget can check sha1 or sh256 hashes in addition to md5,

and can cache and resume partial downloads;

- [NEW] action in WAPTconsole to plan in near future

a restart of waptservice on selected hosts;

- [IMP] mass host update/upgrade in waptconsole actions are now launched

in single shot instead of one host at a time;

- [NEW] allow to force a host_dn in `wapt-get.ini`

when host is not in a domain (**Enterprise** only);

- [NEW] added timeout parameter for setuphelpers

service_start, *service_stop* and *service_restart*;

- [IMP] *group filter list* box is now editable, and one can type

a partial group match and press enter to filter on all matching groups. Separator is comma (,). Handle * at the end of search to find all occurrences even if one group matches exactly;

- [ADD] bat script migrate-hosts.bat to set environment

for `migrate-hosts.py`;

- [ADD] trigger_action.py script to trigger action on pre 1.5 hosts with

reachable 8088 waptservice port from 1.5 server;

- [FIX] `registration_auth_user` reset to None when reusing host certificate

for re-register;

- [IMP] removed unnecessary dependencies krb5-user, msktutil, python-psutil

for waptserver package;

- [IMP] increase client_max_body_size for http post on nginx

for large update/ upgrade trigger:

- fix `signature_clockskew` waptserver config parameter

not taken in account;

- unified loggers for server;
- have waptserver ask wapt clients to update status using websockets

if websocket connection is up but database is not aware of given SID (case where waptserver is restarted but **`nginx`** is kept up, and restart of waptserver service is fast enough to not trigger a reconnection of the clients);

- [FIX] disable proxy for migrate-hosts;
- waptservice: if a system account level http proxy is defined in registry

on the windows host, websocket client library tries to use it and fails to connect to the server. Workaround: make an exception for waptserver;

- waptconsole: if a http proxy is defined in `waptconsole.ini`,

section `[global]`, key *http_proxy*, it is used by the waptconsole even if setting *use_proxy_for_xxx* is False Workround: set *http_proxy* to an empty string in `waptconsole.ini`;

- when using a not self-signed personal certificate, depending of th issuer,

the certificate file <private_dir>mine_cert.crt can contain the full chain (own certificate, intermediate CA, and root CA). When waptconsole asks if the certificate should be put in authorized client certificate directory (<wapt-dir>ssl), the full crt file is copied as this. This means that all certificates in crt file are authorized, and not only the personal one. This is perhaps not desired;

Workaround: check if the personal pem encoded crt file contains the full certificates chain. If this is the case, copy in <wapt-dir>ssl only the parts of the PEM file matching the certificates you want to trust;

- SNI is not properly handled by waptconsole code, leading to incorrect

error about certificate validation on https server with virtual hosts;

- Certificates CSR updates

(periodical signature, ...) must be managed manually using tools like easy-rsa. Only CSR accessible by a URL are supported;

- proxies are not supported on the server, so

CRL can not be updated properly (as far as Distribution Point is defined in certificates) if the server has no direct http access to the distribution points;

- https certificates are verified on the clients using the bundle defined

by the verify_cert ini settings. If this setting is simply *True*, the bundle supplied with python libraries is used to check issuers. This bundle is not updated unless WAPT is upgraded, so new issuers or no more trusted issuers are taken in account only at this point. So it is better to deploy your own CA bundle along with wapt and define the verify_cert path.

- for 1.5.1.18 rc1, on the linux server, there are broken symbolic links

in lib/python2.7 folder. Next RC does not exhibit this problem;

- [NEW] Historize in *wapt_localstatus* PostgreSQL table the dependencies

and conflicts of installed packages (to provide an easy way to warn when conflicting package will be installed or should be removed);

- [FIX] load fill certificate chain from host packages to check control

(as it is the case for other types of packages);

- [SEC] regression: check host package control signature

right after downloading (it is checked too when starting install);

- [FIX] regression: don't install host package if version is lower

than installed one;

- [FIX] don't raise an exception during session-setup if package

has no setup.py;

- [FIX] intermediate CA pinning:

Allow to deploy intermediate CA as authorized package CA without root CA (segragation of rules between entities);

- [FIX] old style print statement (without parentheses)

raising an error in *setup-session* or *uninstall* **setup.py** functions;

- [IMP] Add *cache_dir* parameter to **wget** function;
- [IMP] renamed *cabundle* parameter to *trusted_bundle*;
- [NEW] Add python methods to create certificate

from CSR;

- [ADD] checkbox in create waptagent to sign with sha1 in addition to sha256

for old wapt client upgrades;

- [IMP] force host package version to be at least equal to already installed

host package (when host package is deleted, version was starting again at 0);

- [FIX] regression: check existing host package signature before editing it;
- [FIX] Force waptserver DB structure upgrade at each server startup;
- [ADD] *db_connect_timeout* parameter for pool of

waptserver DB connections;

- [NEW] Store *depends* and *conflicts* attributes in waptserver

HostPackagesStatus PotsgreSQL table;

- SNI is not properly handled by waptconsole code, leading to

incorrect error about certificate validation on https server with virtual hosts;

- certificates CSR updates

(periodical signature, ...) must be managed manually using tools like easy-rsa. Only CSR accessible by a URL are supported;

- Quelques fallback pour permettre l'utilisation de la console WAPT sous Wine
- Ebauche architecture plugins dans waptconsole.
- Interface GUI pour entrer les mots de passe dans PyScripter
- Action make-template dans installeur crée un paquet vide
- Inclusion de la chaine de certificats du signataire dans le paquet

au lieu du seul certificat final

- IMPROVE: gestion des certificats signés par une autorité intermédiaire

pour les actions de la console Wapt

- Ajout option pour spécifier fichier de configuration pour waptconsole.
- [FIX] SNI pour la récupération de la chaine de certificats dans waptconsole.
- [ADD] added actions to launch mass updates/ upgrades, offer updates

to the users (WAPT Enterprise);

- `F5` rafraîchit la liste des paquets
- Changement à distance de la description de l'ordinateur
- Possibilité de configurer plusieurs instances de serveurs Wapt

sur un serveur/ VM.

- chunked http upload pour pouvoir uploader des gros paquets

sans passer par du scp.

- Ajout installation forcée d'un paquet sur un poste dans la console.
- Ajout option pour masquer les actions avancées

(simplication affichage console)

- CN du Certificat / clé machine sont nommés comme l'UUID.
- Si une ou plusieurs dépendances d'un paquet ne peuvent pas être installées,

le paquet parent n'est pas installé et est marqué en erreur.

- Memory leak sur le serveur
- Gestion timezone pour validité de certificats
- [SECURITY] prend tous les fichiers en compte dans la vérification des hashes,

pas seulement ceux dans le répertoire racine (régression apparue en 1.5 mais non présente en 1.3)

- [NEW] the host packages are now named with the BIOS *UUID*

of the machine instead of the *FQDN* (it is possible to use the FQDN as the UUID with the parameter *use_fqdn_as_uuid* but it may create duplicates in the console);

- le service **`waptservice`** écoute sur l'adresse de loopback,

port 8088 et non plus sur toutes les interfaces. Cela réduit la surface d'attaque potentielle si un attaquant spoofe l'adresse IP du serveur WAPT;

- le service **`waptservice`** crée au démarrage

une connexion Websockets (Socket.IO) vers le serveur pour permettre à la console de déclencher les Update/ Upgrade / Install/ Remove ; On ne pass plus par le port 8088 du service;

- [NEW] the Websocket requests from the WAPT console to the WAPT agents are now

signed with the key of the *Administrator*. Before, security relied on source IP restriction and the validation of the Administrator's login/ password;

- la base de données d'inventaire est maintenant une base PostgreSQL

en remplacement de MongoDB. Cela facilite le requêtage pour un reporting personnalisé, le langage SQL étant mieux connu des administrateurs système;

- l'affichage dans la console d'un grand nombre de machines a été amélioré.

L'affichage de plusieurs milliers de machines n'est plus un problème;

- modifier la configuration d'un grand nombre de machines

a été rendu largement plus performant;

- la reprise d'un téléchargement partiel de paquet est

maintenant possible (interruption lors de l'arrêt . . .);

- les clés privées doivent maintenant obligatoirement être protégées

avec un mot de passe;

- passage en Websockets;
- gestion des écrans de haute résolution (ex: écrans 4k);
- modernisation des jeux d'icônes dans la console;
- changement à la volée de la description du poste;
- option pour changer le mot de passe d'une clé;
- la présence du fichier `setup.py` est optionnelle (plus particulièrement,

il n'est pas nécessaire pour les paquets groupes et machines qui ne contiennent que des dépendances);

- [NEW] if the package contains a `setup.py` file, it MUST be signed with a

Code Signing certificate, otherwise the package WILL NOT be installed. The roles are now differenciated between the role of the *Package Deployer* (allowed to sign group and host packages) and the role of *Package Developer* (allowed to sign group, host AND base packages);

- lors de la signature du paquet, le certificat du signataire est ajouté

dans le paquet (`WAPT/certificate.crt`);

- le fichier `manifest` est renommé `manifest.sha256` au lieu de

`manifest.sha1` et `signature.sha256` au lieu de `signature`;

- ajout des attributs suivants au fichier `control`:
- `signed_attributes`: pour la fiabilité de la vérification;
- `min_wapt_version`: le paquet est ignoré (et ne s'installe pas)

si wapt n'est pas au moins à cette version;

- `installed_size`: le paquet ne s'installe pas s'il n'y a pas au moins

cet espace disponible sur le disque système;

- `max_os_version`: le paquet est ignoré si Windows

a une version supérieure à cet attribut;

- `min_os_version`: le paquet est ignoré si Windows

a une version inférieure à cet attribut;

- `maturity`: *PROD*, *PREPROD*, *TEST*;
- `locale`; *fr*, *en*, etc ;
- section explicite `[wapt-host]` pour le dépôt des paquets machines

sinon l'url est déduite de <repo_url>+"-host";

- section explicite `[wapt]` pour le dépôt principal,

sinon <repo_url> est pris en compte;

- vérification des certificats activée par défaut

pour toutes les connexions https;

- signature avec du sha256 au lieu de sha1;
- prise en compte de paquets signés avec des certificats délivrés

par une autorité, déploiement uniquement du certificat de l'autorité;

- utilisation de l'UUID du client pour le nom des paquets machine

au lieu du FQDN;

- possibilité d'utiliser le FQDN comme UUID au lieu de l'UUID du Bios.

(paramètre *use_fqdn_as_uuid*) (ou uuid forcé: paramètre *forced_uuid*);

- lorsqu'on signe, on désigne le signataire par son certificat et

non sa clé privée. La clé privée est recherchée par wapt dans le même répertoire que le certificat personnel. On incite à avoir un certificat par personne agissant sur WAPT;

- possibilité de prendre en compte la révocation de certificats

(la CSR est fournie aux poste lors de l'update, dans le fichier Packages);

- re-signature possible sous Linux avec

la commande **wapt-signpackage.py**;

- installation dans `Program Files(x86)` par défaut;
- *running_as_admin*, *running_as_system*;
- correctif sur **add_shutdown_script**;
- ajout paramètre *remove_old_version* pour **install_msi_if_needed** et

install_exe_if_needed;

- ajout fonction **update-package-sources** qui lance

la fonction optionnelle **update_package()** du paquet;

- remplacement de l'option *–private-key* par l'option *–certificate*

pour désigner le certificat à utiliser pour signer le paquet. La clé privée est recherchée dans le même répertoire que le certificat;

- remplacement du fichier `WAPT/wapt.psproj` à chaque édition d'un paquet

(pour mettre à jour le chemin vers les modules WAPT suivant l'installation dans `C:\wapt` ou `C:\Program Files (x86)\wapt`);

- vérification du certificat serveur lors du **enable-check-certificate**

pour éviter de mauvaises configurations;

- ajout options

```
```

–if-needed –message-digest –scan-packages –message-digest

```
```

Usage: wapt-signpackages -c crtfile package1 package2

Re-sign a list of packages

Options: -h, –help show this help message and exit -c PUBLIC_KEY, –certificate=PUBLIC_KEY Path to the PEM RSA certificate to embed identitiy in control. (default:) -k PRIVATE_KEY, –private-key=PRIVATE_KEY Path to the PEM RSA private key to sign packages. (default:) -l LOGLEVEL, –loglevel=LOGLEVEL Loglevel (default: warning) -i, –if-needed Re-sign package only if needed (default: warning) -m MD, –message-digest=MD Message digest type for signatures. (default: sha256) -s, –scan-packages Rescan packages and update local Packages index after signing. (default: False)

- [NEW] all actions sent to the hosts are signed with the Administrator's key;
- [NEW] generation of a key / certificate pair signed by

a Certificate Authority (WAPT Enterprise);

- option de créer un certificat **Code Signing** ou non (version Enterprise);
- option pour changer le mot de passe d'une clé RSA;

- option de vérification des certificats lors de la

création du **waptagent**;

- lancement TISHelp (version Enterprise);
- limitation du nombre de machines retournées dans la console;
- ajout filtre *reachable* = poste connecté au serveur WAPT;
- possibilité de changer la description du poste
- authentification sur une base LDAP (version Enterprise);
- utilisation des Websockets pour les actions;
- le Webservice http de **waptservice** écoute uniquement

sur la loopback 127.0.0.1 (donc plus de vérification si port 8088 ouvert sur firewall..);

- le **waptservice** se connecte en websocket au serveur WAPT

si le paramètre *waptserver* est présent dans `wapt-get.ini`;

- le paramètre *websockets_verify_cert* active la vérification SSL du certificat

pour la connexion websockets;

- affichage de liste des certificats / CA autorisés pour les paquets;
- affichage signataire paquet;
- [NEW] *allow_user_service_restart* parameter allows a standard user to restart

the WAPT service on her computer;

- lancement de **tishelp** en mode service par URL /tishelp;
- suppression installation **msvcrt**;
- restent uniquement 2 options: installer le service et lancer

wapttray;

- options pour une installation silencieuse:
- *dnsdomain* pour la recherche auto wapt et waptserver
- *wapt_server*
- *repo_url*
- **waptupgrade** fait systématiquement une installation complète

(pas d'installation incrémentale);

- `setup.py` pas obligatoire pour uninstall;
- chemin unicode pour édition de paquets;
- corrigé la recherche de dépots en s'appuyant sur les DNS;
- corrigé \0000 pour PostgreSQL;
- introduit une option pour avoir une double signature sha1 et sha256;
- vérification https pour upload **waptagent**;
- option *–if-needed* dans **wapt-signpackages**;

- fix proxy dans import paquets;
- gestion des révocations de certificats (CSR);
- fix attributs requis dans signature actions;
- *max_clients*;
- fix option sans serveur (**waptstarter**);
- ajout lancement **tishelp**;
- force update à l'installation;
- pas de release officielle;
- [NEW] migration sur la base PostgreSQL à la place de MongoDB;
- régression: Package files content check was skipped if signature of `manifest`

and `Packages` index file checksum was ok. This regression affects all 1.3.12 releases, but not WAPT <= 1.3.9 and >= upcoming 1.5. In order to exploit this bug, one would need to tamper the `Packages` files either through a MITM (if you do not have valid https certificate check) or a root access on the WAPT server.

- compatibility with packages signed with upcoming WAPT 1.5.

With WAPT 1.5, package are signed with sha256 hashes. An option allows to sign them with sha1 too so that they can be used with WAPT 1.3 without signing them again.

- new package certificate for Tranquil IT packages.

previous certificate for package on store.wapt.fr has expired. all packages on store.wapt.fr has been signed again with new key / certificate with both sha1 and sha256 hashes, and WAPT 1.5 signature style (control data is signed as well as files)

- fix for local GPO add_shutdown_script() function (thanks jf-guillou!)
- fix for **waptsetup.exe** postinstall actions (**update** / **register**)

when running **waptsetup.exe** installer without elevated privileges: added *runascurrentuser* flag

- remove needless python libraries to make install package slimmer
- [NEW] Assistant de création de paquets à partir d'un fichier `MSI` ou d'un `Exe`;
- [NEW] Option dans le menu *Outils* ou par drag drop dans l'onglet dépôt privé;
- [NEW] Découverte des options silencieuses;
- [NEW] Utilisation des fonctions **install_exe_if_needed** et **install_msi_if_needed**

au lieu d'un simple **run()** pour les exes et les MSI (plusieurs templates de `setup.py` dans `C:\wapt\templates`);

- [NEW] Amélioration significative de la vitesse de modification en masse des paquets machines;
- [NEW] Vérification optionnelle de la signature des paquets que l'on importe d'un dépôt extérieur.

La liste des certificats autorisés se trouve par défaut dans `%APPDATA%\waptconsole\ssl` et peut-être précisée dans les paramètres de la **waptconsole**. Le paramètre ini se nomme *authorized_certs_dir*. Sinon, les certificats autorisés sont ceux dans `C:\wapt\ssl`;

- [NEW] Vérification optionnelle du certificat https pour les dépôts extérieurs dans la console;
- [NEW] Vérification de la signature des paquets machines, groupes et logiciels

avant leur modification dans la console ou dans **PyScripter**;

- [NEW] Lors de l'import d'un dépôt extérieur, possibilité d'éditer le paquet

pour inspection plutôt que de le charger directement sur le dépôt de production;

- [NEW] Changement des URL relatives à la documentation. https://www.wapt.fr/en/doc/;
- [NEW] Possibilité d'actualiser le certificat sans recréer la paire de clés RSA

(en particulier pour préciser un Common Name correct, qui apparaît comme le signataire des paquets);

- [NEW] HTTPS par défaut pour les URL de dépot.
- [FIX] Paramètre *AppNoConsole:1* pour NSSM (**waptservice** / **waptserver**)

pour permettre le fonctionnement sur Windows 10 Creators Updates;

- [FIX] Problème de fichier Zip qui restent verrouillés si une erreur est déclenchée;
- [FIX] Suppression répertoire temporaire lors de l'annulation d'édition d'un groupe;
- [FIX] Gestion espace dans les fichiers de projet PyScripter;
- [FIX] Gestion utf8 / unicode pour certaines fonctions;
- [FIX] Fix gestion encoding quand **run_not_fatal()** renvoie une errreur;
- [FIX] remplacement librairie mongo.bson par json natif de python ,
- [FIX] bug dans la synchro des groupes AD avec les paquets WAPT;
- [FIX] bug « La clé privée n'existe pas » la première fois qu'elle est renseignée

si on ne redémarre pas la console;

- [FIX] bug « redémarrage service wapt » (merci à QGull);
- [FIX] possibilité d'avoir des majuscules dans les noms de paquet

(toutefois pas recommandé, les noms des paquets sont sensibles à la casse);

- [FIX] quelques actualisation des exemples de configuration `wapt-get.ini.tmpl`
- [FIX] la compilation du **waptagent** échoue si les clés / certificats

existent déjà mais que le certificat a été supprimé de `C:\wapt\ssl`;

- [FIX] affichage dans la barre des tâches de la fenêtre de login

(pour permettre en particulier l'autofill par des gestionnaires de mot de passe);

- [FIX] Argument *shell = True* was not explicitly passed to the underlying

function as it occurred on previous versions.

- [FIX] update code to follow more PEP8 recommandations;
- [FIX] upgradedb locks sqlite database issue;
- [FIX] Fix broken DNS SRV record discovery;
- [FIX] Fix unicode handling of signer / CN / organization in certificates;
- [FIX] Unzipped netifaces module;
- [NEW] Expands wildcards args for **install**, **show**,

build-package, **sign-package**;

- [FIX] Fix **show-params** wapt-get command;
- [FIX] Fix **register** with description not working on some computers;

- [FIX] Fix broken *-c –config* option;
- [NEW] **reg_key_exists**;
- [NEW] **reg_value_exists**;
- [NEW] **run_powershell**;
- [NEW] **remove_metroapp**;
- [NEW] **local_users_profiles**;
- [NEW] **get_profiles_users**;
- [NEW] **get_last_logged_on_user**;
- [NEW] **get_user_from_sid**;
- [NEW] **get_profile_path**;
- [NEW] **wua_agent_version**;
- [NEW] **local_admins**;
- [NEW] **local_group_memberships**;
- [NEW] **local_group_members**;
- [IMP] command:*run*: explicit default values for **run** command help in **PyScripter**.

Added *return_stderr argument* (overloaded str object);

- [FIX] **run_notfatal**: fix unicode issue in use wmi module for **wmi_info_basic**

instead of **wmic** shell command;

- [IMP] **make_path**: improved when first argument is a drive.

Be smart if an argument is a callable;

- [FIX] **CalledProcessError**: restored command:*CalledProcessError* alias;
- [ADD] **host_infos**: added *profiles_users*, *last_logged_on_user*,

local_administrators, *wua_agent_version* attributes;

- [IMP] **ensure_unicode**: return None if None, for bytes strings,

try utf8 decoding before system locale decoding;

- [FIX] restore allowed lowercase/uppercase package naming;
- [ADD] 4 host popup menu actions:
- *Computer Mgmt*;
- *Computer Users*;
- *Computer Services*;
- *RemoteAssist*;
- [FIX] fixed other issues in the WAPT console:
- Don't search host while typing;
- utf8 search (accents. . .);
- utf8 compare;

- try to get localized versions of special folders;
- [ADD] **waptpythonw.exe** binary in distribution for console less python scripts

(to avoid having **cmd.exe** windows poping up when invoking a python script);

- [FIX] change default wapt templates URL to https://store.wapt.fr/wapt;
- [FIX] when upgrading, (full **waptagent.exe** install) remove stalled

waptagent.exe installs;

- [SEC] Fix inheritance of rights on wapt root folder for Windows 10 during setup

when installed in `C:\wapt`. On Windows 10, **cacls.exe** does not work and does not remove « Authenticated Users » from `C:\wapt`. **cacls.exe** has been replaced by **icacls.exe**:

- on pre-wapt 1.3.7 systems, you can fix this by running the following command,

or upgrade to wapt 1.3.8 (you may check `icacls.exe c:\wapt /inheritance:r`) * This can be achieved with a GPO, or a wapt package

- [IMP] in next versions of WAPT, the default install path of wapt will be changed

from root folder `C:\wapt` to a more standard `C:\Program Files (x86)\wapt`.

- [IMP] By default, **waptsetup.exe** / **waptsetup-tis.exe** do not

distribute certificates to avoid to deploy directly packages from Tranquil IT. **waptagent.exe** by default distributes the certificates that are installed on the mangement desktop creating the **waptagent**.

- [IMP] The database structure has changed between 1.3.8 and 1.3.8.2 to include

additional attributes from packages: *signer*, *signer_fingerprint*, *locale*, and *maturity*. *signer* and *signer_fingerprint* are populated when signing the package to identify the origin. This means local WAPT database is upgraded when first starting WAPT 1.3.8.2 and this is not backward compatible;

- [IMP] Installers have a limited set of options, the most common use of WAPT is privileged;
- [ADD] 3 new parameters for the **waptexit** policy behavior: *hiberboot_enabled*,

max_gpo_script_wait, *pre_shutdown_timeout*. These parameters are not set by default and should be added to `wapt-get.ini` *[global]* section if needed;

- [IMP] Use user's `waptconsole.ini` configuration file instead of `wapt-get.ini`

for the commands targeted to package development (*sources*, *make-template*, *make-host-template*, *make-group-template*, *build-package*, *sign-package*, *build-upload*, *duplicate*, *edit*, *edit-host*, *upload-package*, *update-packages*. This avoids the need to write these parameters in `wapt-get.ini` on the development workstation. These parameters are not shared across multiple users on same machine. One use case is to allow multiple profiles (key, upload location) depending on the maturity of package (development, test, production...);

- [ADD] helper functions **dir_is_empty**, **file_is_locked**,

service_restart and **WindowsVersions** class

- [IMP] Added referer and *user_agent* in **wget** and **wgets**
- [IMP] run function: define stdin as PIPE to avoid lockup process waiting for input

or error like unable to duplicate handle when using for example powershell

- [IMP] Version class: try to compare version using at least Version.members_count
- [FIX] encoding fixes for registry functions, fix encoding

for registry_setstring key name

- [FIX] **install_exe_if_needed**: don't check uninstall_key

or min_version if not provided

- [FIX] **install_exe_if_needed** and **install_msi_if_needed**

version check if *–force*

- [UPD] Check version and uninstall key after install with **install_exe_if_needed**

and **install_msi_if_needed**

- [UPD] inventory includes informations from WMI.Win32_OperatingSystem
- [ADD] **get_disk_free_space** helper function
- [UPD] check free disk space when downloading with **wget**.

Check http status before.

- [UPD] Version class: Version("7")<Version("7.1") should return True
- [ADD] 2 commands to get server SSL certificate and activate the certificate checking

when using https with waptserver

- [FIX] **get_sources** to allow svn checkout of a new package project
- [FIX] **register** problems with some BIOS with bitmaps
- [UPD] Check uninstall key after package install if uninstallkey is provided
- [FIX] added compatibility OS in `manifest` file for **wapt-get**

and **waptconsole** version windows

- [FIX] erroneous error messages for **session-setup** in the WAPT console
- [UPD] add « pattern » parameter to all_files function
- [FIX] Install Date incorrectly registered by **register_uninstall**
- [ADD] **user_local_appdata** function
- [ADD] add the *signer* CN and *signer_fingerprint*

to `control` file when building package

- [ADD] add control attributes *min_wapt_version* to trigger an exception

if `Package` requires a minimum level of libraries. The version is checked againts **setuphelpers.py** "s __version__ attribute.

- [ADD] *authorized_certificates* attribute is sent to the WAPT server.

It contains the list of host's signer certificates distributed on the host

- [FIX] When signing, check if WAPT zip file has already a `signature` file.

(python zipfile can not replace the file inline)

- [ADD] Show *All Versions* checkbox in *Available Packages* page
- [UPD] Skin updated
- [ADD] *Filter* searchbox for available packages
- [ADD] Add *NOT* checkbox for keywords search in **waptconsole**

to search for hosts NOT having a specific package or software...

- [FIX] fix integer limit for grid display of package size, use int64

for size of packages in **waptconsole**.

- [UPD] don't list packages of section « restricted » in local webservice

available packages list

- [UPD] *Common Name* attribute should be populated now, so that signer identity

is not None in package `control` file.

- [ADD] signer's identity column in packages grid
- [FIX] escape quotes in package's description
- [ADD] Check **waptagent.exe** version against **waptsetup-tis** version

at **waptconsole** startup.

- [UPD] try to display a *progress* dialog

at **waptconsole** startup

- [FIX] company not set when building customized **waptagent.exe**
- [ADD] initialize Organization in **waptagent.exe** build with CN

from certificate.

- [UPD] some text introduction changes
- [NEW] Limit trayicon balloon popup when Windows version is above Windows 7

or if *notify_user = 0* in `wapt-get.ini`

- [UPD] Use broadcast address on interface for wakeonlan call
- [FIX] remove the check of wapt server password which prevents

the proper registration of **waptserver** on Windows.

- [UPD] when upgrading, reuse existing `waptserver.ini` file if it already exists,

don't overwrite server_uuid and ask for password reset if it already exists

- [FIX] **waptdeploy** not working on WinXP removed

DisableWow64FileSystemRedir on **runtask**.

- [FIX] **waptupgrade**: Missing quotes for system account on Windows XP
- [ADD] BeautifulSoup for wapt packages auto updates tasks
- [UPD] **winsys** library update to "1.0b1"
- [ADD] *UUID* parameter for direct requests to hosts from the WAPT Server;
- [ADD] allow host to refuse request if not right target (if ip has changed

since last **update_status** for example)

- [ADD] fallback on waptserver usage_statictics if mongodb lacks aggregate support
- [IMP] register host on server in postconf using **waptservice** http

instead of command line **wapt-get**

- [ADD] **reset-uuid** and **generate-uuid** for

https://roundup.tranquil.it/wapt/issue421 duplicated *UUID* issues

- [IMP] mass hosts delete, added delete hosts package action. server >=1.2.2 only:

https://roundup.tranquil.it/wapt/issue433

- [ADD] read the docs theme for sphinx setuphelpers API documentation. WIP

https://roundup.tranquil.it/wapt/issue427

- [IMP] doc updates
- [ADD] api/v1/hosts_delete method
- [ADD] **need_install**, **install_exe_if_needed**,

install_msi_if_needed functions to setuphelpers

- [ADD] parameters for **waptdeploy**.
- [ADD] combobox for filtering on groups in **waptconsole**.
- [ADD] *Add ADS Groups as packages* action

to WAPT host selection popup menu

- [ADD] **cleancache** action to clean local waptconsole packages cache
- [ADD] added **notify_server** on network reconfiguration

if **waptserver** is available;

- [IMP] column *groups* shows only host's direct dependencies with package's

section == « group » instead of all direct dependencies.

- [ADD] optional anonymous statistics (nb of machines, nb of packages, age of updates. . .)

sent to Tranquil IT to document the communication around WAPT (sent by **waptconsole** at most every 24h)

- [IMP] improved mass hosts delete,
- [ADD] delete hosts package action. server >=1.2.2 only:

https://roundup.tranquil.it/wapt/issue433

- [IMP] big packages uploads (write uploaded packages by chunk)

(but still some issues on 32bits servers due to **uwsgi**)

- [IMP] display version of mismatch when editing package
- [FIX] host's packages not saved when some dependencies don't exist anymore
- [FIX] restore working *Cancel running task* button
- [FIX] canceling subprocesses not working in freepascal apps

(when waiting for **InnoSetup** compile for example)

- [ADD] **reset-uuid** and **generate-uuid** for

https://roundup.tranquil.it/wapt/issue421 duplicated *UUID* issues

- [IMP] **find_wapt_repo_url** processus to avoid waiting for all repos

if one repo is ok (improved response time in buggy networks)

- [IMP] windows DNS resolver in wapt client (python part) instead of pure python resolver.

Should reduce issues when multiple network cards or inactive network connections.

- [IMP] changed priority of server discovery using SRV dns records.

-> first priority ascending and weight descending. -> comply with standards.

- [FIX] solved some issues with **SQLite** and threads

in local **waptservice**

- [IMP] explicit transaction handling and *isolation_level = None*

for local waptDB (to try to avoid locks)

- [IMP] teardown handler for **waptservice** to commit

or rollback thread local connections

- [FIX] for waptrepo detection in freepascal parts: same processus as python part.
- [FIX] for **edit_package** when supplying a wapt filename

instead of package request

- [ADD] read the docs theme for sphinx setuphelpers API documentation.

WIP https://roundup.tranquil.it/wapt/issue427

- [ADD] _all_ list to avoid importing unecessary names

in **setup.py** modules. Now only functions defined in **setuphelpers** are available when importing **setuphelpers**. This can break some WAPT packages if names were indirectly imported through **setuphelpers** module.

- [ADD] **need_install**, **install_exe_if_needed**,

install_msi_if_needed functions to **setuphelpers**

- [ADD] **local_desktops** function
- [FIX] version class instances accept to be compared to str
- [REM] **processnames_list** which is unused in **setuphelpers**
- [ADD] **add_ads_groups** and **get_computer_groups**

to **waptdevutils.py**

- [FIX] **run** helper
- [FIX] on_write callback not working
- [FIX] TimeoutExpired not formatted properly
- [FIX] use closure for registry keys
- [IMP] **waptdeploy** with more command line options

(in particular tasks to merge to default innosetup selected tasks)

- [FIX] waptrepo detection using dns records
- [FIX] **waptagent** upload error on windows
- [FIX] debian packages should work for Jessie
- [IMP] **copytree2** for **waptupgrade**

- [FIX] trap exception for version check on copy of `exe` and `dll`
- [FIX] **`mongodb-server`** version should be >= 2.4
- [IMP] the loading of the main grid has been optimized; only configured

columns are displayed;

- [IMP] the WAPT server detects the hosts whose **`waptservice`** is listening.

Their *Reachable* status is shown with a green / grey indicator;

- [IMP] the WAPT package to upgrade WAPT on hosts (???-waptupgrade.wapt)

is generated by the WAPT console at the same time as the WAPT agent installer (**`waptagent.exe`**), the two files are then uploaded on the WAPT server;

- [ADD] the package dependencies of each host are displayed in the grid.

This allows to see what hosts have no package;

- [ADD] possibility to trigger available package upgrades on hosts

that are listening from the WAPT console. In that case, the host sends its status to the WAPT server after the upgrade;

- [ADD] possibility to filter hosts in the WAPT console according to their upgrade status

or whether they are « reachable » or not,

- [ADD] when packages are flagged for install but are not yet installed on a host,

they appear with a blue « + » indicator. It is then possible to force the immediate install of the package with a right-click;

- [ADD] cleaning of the cache on the hosts after each successful upgrade;
- [ADD] the versions of the WAPT agent, WAPT Server are shown in the main web page

of the WAPT Server (with a red indicator if there is a problem);

- [ADD] functions to **`setuphelpers`** to manage shortcuts:
- **`remove_desktop_shortcut`**;
- **`remove_user_desktop_shortcut`**;
- **`remove_programs_menu_shortcut`**;
- **`remove_user_programs_menu_shortcut`**;
- [IMP] verification of used ports during the post-configuration of WAPT Server on a Windows machine;
- [IMP] the **`waptserver`** no longer listen on 8080 port by default.

The Apache frontal web server listens in HTTP and HTTPS and relays action calls to the python **`waptservice`** that only listens locally.

It is therefore necessary to update `wapt-get.ini` files on WAPT agents and to replace *wapt_server* = http://srvwapt.mydomain.lan:8080 with *wapt_server* = https://srvwapt.mydomain.lan

If you can not make that change to your WAPT agents, it is possible to return to the previous behavior.

On Debian, edit the file `/opt/wapt/waptserver/waptserver.ini`, and in the `[uwsgi]` section, put:

http-socket = 0.0.0.0:8080

On Windows, edit `C:\waptwaptserver\waptserver.ini` and replace:

server = Rocket((“127.0.0.1”, port), “wsgi”, {« wsgi_app »:app})

with:

server = Rocket((“0.0.0.0”, port), “wsgi”, {« wsgi_app »:app})

The repository may stay in HTTP on port 80.

The calls to the WAPT Server are authenticated, but it is advized to restrict access to authorized sub-networks with a firewall.

- [IMP] json calls to the webservice of the WAPT Server are now standardized;
- [IMP] when launching command:*update* / command:*upgrade* / command:*remove*

/ command:*forget* / command:*tasks_status* actions from the WAPT console, the IP address of the host is no longer sent, but instead its *UUID*, and it is the WAPT Server that finds the IP address and the port to use; et c’est le serveur wapt qui s’occupe de déterminer quelle IP / port utiliser;

- [ADD] verification in the WAPT console that the version of the WAPT Server is sufficient;
- [ADD] the timeout to connect to WAPT agents and read the data are configurable in `waptserver.ini`;
- [ADD] first public version of WAPT

6.25 Licences des composants externes utilisés dans WAPT

Le développement du logiciel WAPT a commencé en mars 2012 ; il est porté en très grande partie par l’équipe de Tranquil IT.

Les développements réalisés dans le cadre du projet WAPT Community sont distribués sous licence GNU Public Licence v3.0.

Les extensions incluses dans la version Enterprise de WAPT sont propriétaires.

Table31: Licences des composants externes utilisés dans WAPT

Composant WAPT	Licence
Python	Python Software License
Librairies Python	Licenses OpenSource diverses
Lazarus	GNU Public Licence
Composants Lazarus	GNU Lesser General Public License
Librairies Lazarus	Licenses OpenSource diverses
OpenSSL	Openssl License
Redistr. Microsoft Visual C++	Microsoft Software License Terms
PostgreSQL	PostgreSQL License
NSSM	Public Domain
Nginx	2-clause BSD-like license

6.26 Contacter l'éditeur de WAPT

Contactez-nous pour plus d'informations :

- Tranquil IT : https://www.tranquil.it/
- Twitter : https://twitter.com/tranquil_it
- Linkedin : https://www.linkedin.com/company/tranquil-it
- Viadeo: https://fr.viadeo.com/fr/company/tranquil-it-systems/
- Forum en français : https://forum.tranquil.it/
- Forum en anglais: https://www.reddit.com/r/WAPT
- Liste de diffusion : https://lists.tranquil.it/listinfo/wapt

CHAPTER 7

Index et tables

- genindex
- *Glossaire*
- search