WAPT Documentation

Release 1.8.2

Tranquil-IT Systems

Jan 10, 2023

PRESENTATION

Welcome to WAPT's official documentation by Tranquil IT, last compiled on 2023-01-10.

Click here for a PDF version of the complete documentation.

WAPT is a software deployment tool whose core set of features is licensed under the GPLv3.

WAPT exists in two flavors, *WAPT Community and WAPT Enterprise*.

CHAPTER

ONE

SECURITY CERTIFICATION

Fig. 1: Security Visa from ANSSI dated 14th of February 2018 for WAPT Enterprise Edition 1.5.0.13
Cybersecurity solutions are many and diverse, but not all of them offer the same level of effectiveness, robustness and trust.
Security Visas that the French cyberdefense agency, ANSSI, delivers allow to identify more easily the most reliable security solutions. They are recognized as such after having been evaluated by approved laboratories following a rigorous and recognized methodology.

CHAPTER

TWO

SECURITY CERTIFICATION FROM FRENCH CYBERDEFENSE AGENCY ANSSI

Following its first level security certification obtained on 14 February 2018, WAPT has been prized with a higher level certification from ANSSI.

CHAPTER THREE

MAIN FEATURES

For System Administrators:

- install software and configurations silently;
- maintain up to date an installed base of software and configurations;
- configure software at the system and user level to reduce the load on support teams;
- remove unwanted or out of cycle software and configurations silently;
- give *Users* more autonomy to install software safely and reliably;
- reduce as much as possible the consumption of bandwidth on remote sites to preserve it for productive uses;

For IT Security Officers

- pilot the software installed base to converge to a security standard acceptable to the Organization;
- prepare your enterprise for the coming GDPR and help your DPO keep his register of data processing, because you two will become close colleagues;
- to no more tolerate machines operating in *Administrator* mode;
- reduce the level of exposure to software vulnerabilities and lateral movement attacks;
- bring up audit indicators for a better knowledge of the state of installed IT devices and their global security level;
- be prompt to deploy updates to react to cyber attacks like Wannacry or notPetya;

For End-Users

- have your software configured to work well in the context of your Organization and trust that they will work correctly;
- reduce your need for support by your IT teams, whose reaction times are often long because of their workloads;
- have better working and more predictable IT systems because of standard software configurations;

CHAPTER

FOUR

SOURCE CODE REPOSITORY

You may access the source code by visiting Tranquil IT's GitHub repository located at https://github.com/tranquilit/WAPT.

CHAPTER

FIVE

HOW TO CONTRIBUTE?

You may want to have a look at our *contribution guide*.

CHAPTER

SIX

WAPT SUPPORT AND TRAINING

- commercial support: https://www.tranquil.it/
- forum: https://forum.tranquil.it/
- mailing list: https://lists.tranquil.it/listinfo/wapt

6.1 Introduction to WAPT

6.1.1 For what purpose is WAPT useful?

WAPT installs, updates and removes software and configurations on Windows devices. Software deployment (Firefox, MS Office, etc.) can be carried out from a central server using a graphical console. WAPT is taking many ideas from Debian Linux apt package management tool, hence its name. WAPT **Community** Edition is distributed under the **GPLv3** license. **Enterprise** Edition is distributed under a proprietary license.

WAPT is intended to help IT administrators manage their deployed base of computer desktops, laptops, tablets running a Microsoft Windows client (from XP to 10), their deployed base of Windows servers (from 2003 to 2019) or their deployed base of Windows Intel tablets.

Private companies of all sizes, Colleges, Schools, Universities, research labs, local and state governments, Hospitals, city hall, state ministries are successfully using **WAPT**.

WAPT exists in two versions, **Community** and **Enterprise**.

WAPT is very efficient to address **recurrent Flash and Java update needs** and it is often to cover that basic need that WAPT is initially adopted; it then becomes a tool of choice for the sysadmin's daily tasks.

If you're a developer, WAPT can help you configure your development environment on your computer as Chocolatey / NuGet can do.

6.1.2 The genesis of WAPT

Our assessment after 15 years of IT management

Managing large IT installed bases of Microsoft Windows computers is today a difficult task in a secured environment:

- common ghosting methods (*Clonezilla* or *Ghost*) are efficient on homogeneous IT infrastructures with roaming user profiles;
- deployment software (*OCSInventory* or *WPKG*) can broadcast software but does not easily allow software level or user level customizations that are useful to prevent or limit user support requests;

- software from smaller vendors often need *Local Administrator* rights to run properly;
- currently available solutions to address theses 3 problems are either too expensive or too inefficient, and they are in every case too complex;

WAPT development hypotheses and motivations

The development of WAPT is motivated by these two principles:

- what is **complicated** should be made **simple**;
- what is **simple** should be made **trivial**;

WAPT relies on a small set of fundamental hypotheses:

- sysadmins know script languages and WAPT has chosen Python for the depth and breadth of its libraries;
- sysadmins who have little experience with scripting languages must find inspiration in simple and efficient examples that they'll adapt to fit their needs;
- sysadmins must be able to communicate on the efficiency of their actions to their superiors and report process gaps to internal or external auditors;
- sysadmins must be able to collaborate with their IT team in full trust; thereby WAPT local repositories provide signed packages they can trust to be deployed on their network. Alternatively, they can choose external repositories providing them the security guarantees they consider sufficient;
- sysadmins are aware that user workstations serve business purposes and some customizations must be possible. The adaptation of the infrastructure to the business needs is facilitated by the notion of groups; it allows to select a large number of machines to customize their configuration;

6.1.3 Fundamental principles

Package/ Repository principle

WAPT packages

A WAPT package structure is similar to Debian Linux **.deb** packages. Each WAPT package includes the binaries to be executed and the other files it needs.

A package is easily transportable.

Here is how a WAPT package looks:

WAPT repositories

Packages are stored in a web repository. They are not stored in a database.

They are served by the **`Nginx`** web server, available with Linux and Windows.

The `Packages` index file is the only thing necessary. It lists the packages available on allowed repositories and some basic information on each package.

That mechanism allows to easily set up a replication process between multiple repositories.

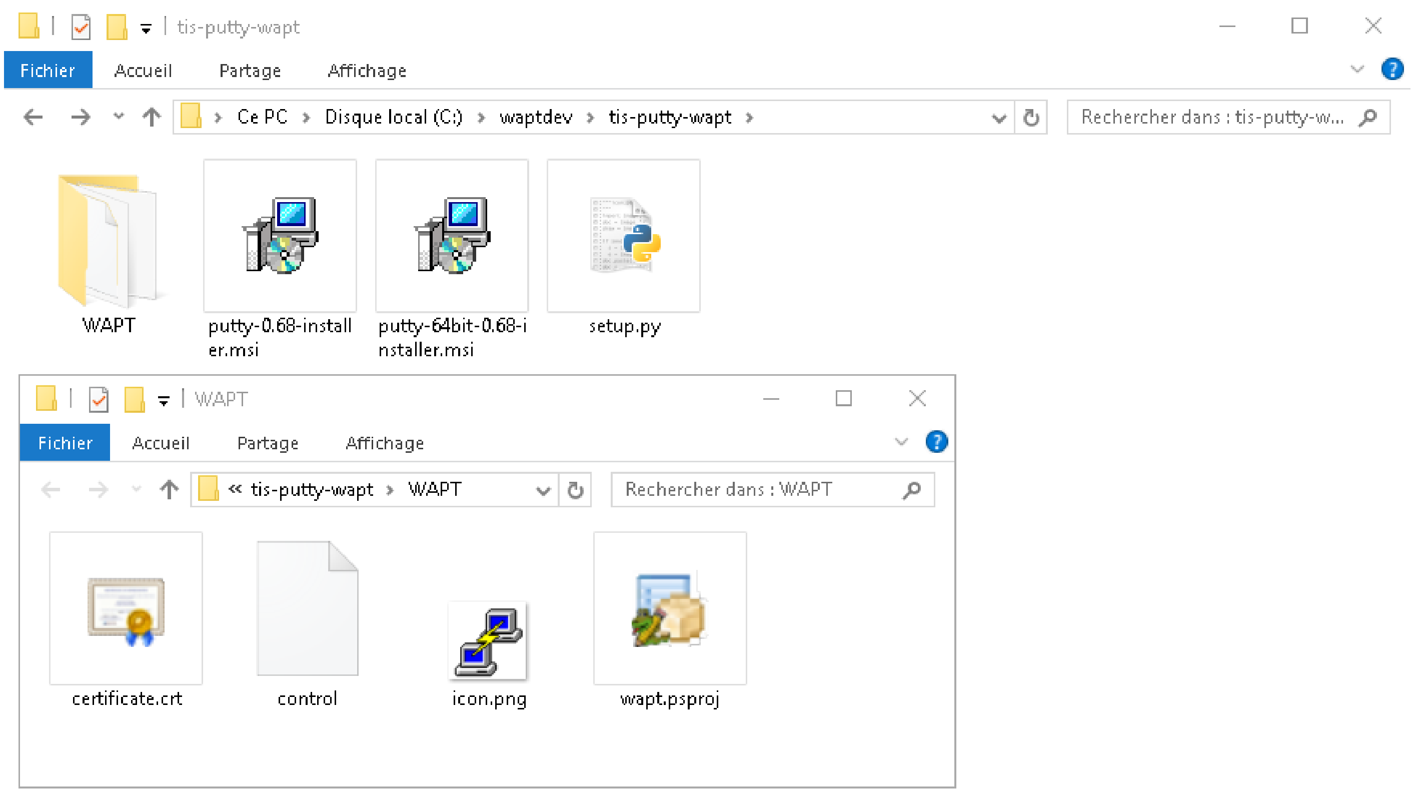


Fig. 1: WAPT package structure

Types of WAPT packages

There are 7 types of WAPT packages:

Fig. 2: Anatomy of a simple WAPT package

Base packages

They are classic software packages.

They are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

Group packages

They are groups/ bundles of packages.

Hint:

- a group / bundle of softwares often corresponds to a *host profile* (ex: **accounting**);
- a group of hosts often corresponds to a room, building, etc;
- a host can be a member of several groups (ex: one or more hosts profiles in the same room in a building);

They are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

Host packages

Host packages are named after the *UUID* of the computer BIOS.

Each host will look for its *host* package to know the packages that it must install (i.e. *dependencies*).

They are stored in the web directory https://srvwapt.mydomain.lan/wapt-host/.

Unit packages

New in version 1.6: Enterprise

Unit packages bear the complete name of OU (Organizational Unit), example: **OU=room1,OU=prod,OU=computers,DC=mydomain,DC=la**

By default, each computer looks for the *unit* packages that the host belongs to:

- OU=room1,OU=prod,OU=computers,DC=mydomain,DC=lan;
- OU=computers,DC=mydomain,DC=lan;
- DC=mydomain,DC=lan;

and then installs the list of associated dependencies.

Unit packages are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

Unit packages are not explicitly assigned to the host (i.e. as dependencies in the *host package*) but are implicitly taken into account by the WAPT agent dependency engine during the WAPT upgrade.

Note: If the computer is removed from an Organizational Unit, obsolete *unit* packages will be removed.

waptwua packages

waptwua packages contain the list of authorized or prohibited Windows Updates.

When this package is installed on the endpoint, the next update scan performed by WAPT will choose Windows updates based on this filtering.

If the host has several *waptwua* packages, then WAPT will merge all package rules.

When this package is installed on the host, the next **`update`** will scan for official Windows updates applicable to the host based on this filtering.

They are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

selfservice packages

New in version 1.7: Enterprise

selfservice packages contain a list of groups or users (Active Directory or local) and their associated lists of authorized packages that Users are allowed to install by themselves.

They are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

profile packages

New in version 1.7: Enterprise

profile packages are similar to *group* packages.

However, *profile* packages work a little differently and are most useful when an Active Directory Server is operating within the *Organization*:

- the WAPT agent will list the Active Directory groups where the host belongs;
- if a *profile* package has the same name as the Active Directory group, then the WAPT agent will install automatically the *profile* package for the Active Directory group of which it is a member;

If the host is no longer a member of its Active Directory group, then the *profile* package will be uninstalled.

Profile packages are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

Profile packages are not explicitly assigned to the host (i.e. as dependencies in the *host package*) but are implicitly taken into account by the WAPT agent dependency engine during the WAPT upgrade.

Note: For performance reasons, this feature is enabled only if the *use_ad_groups* option is enabled in `wapt-get.ini`.

Dependency mechanism

In WAPT everything works on the principle of dependencies.

By default, the WAPT agent will look for its host package. The *host* package lists packages to install on the computer.

The *host* package is correctly installed when all its dependencies are satisfied.

Each sub-dependency must be satisfied to satisfy an upper-level dependency.

When every dependency is satisfied, the host notifies its status to the WAPT Server and its indicator turns **OK** and green in the WAPT console, meaning the host has the host profile that the *Administrator* or *Package Deployer* has defined for it.

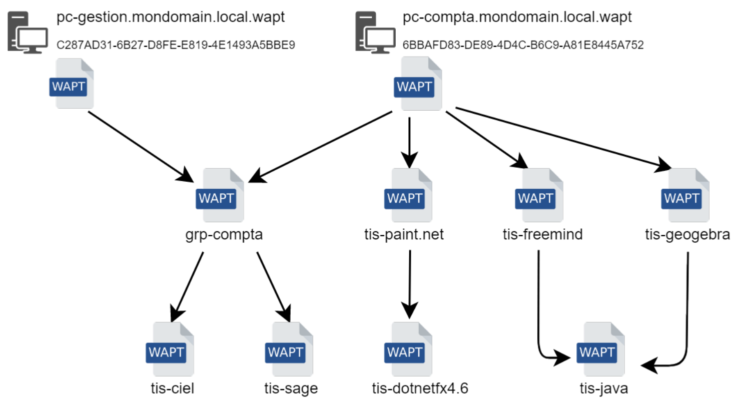

pc-gestion.mondomain.local.wapt
C287AD31-6B27-D8FE-E819-4E1493A5BBE9
pc-compta.mondomain.local.wapt
6BBAFD83-DE89-4D4C-B6C9-A81E8445A752
WAPT
grp-compta
tis-paint.net
tis-freemind
tis-geogebra
tis-ciel
tis-sage
tis-dotnetfx4.6
tis-java

Fig. 3: Conceptual diagram of the dependency mechanism

Hint: When attributing a software package to a host as a dependency, only the software canonical name without its version number is registered as a dependency (ex: I want Freemind to be installed on this machine in its latest version and **`Freemind`** to be configured so that the *User* does not call me because she does not find the icon on her desktop!).

For each dependency, the WAPT agent will take care of automatically installing the latest available package version. So if several versions of **`Freemind`** are available on the repository, the WAPT agent will always get the latest version, unless I have pinned the version for reason of compatibility with other sets of tools.

Afterwards, when the agent contacts the repository to check for new updates, it will compare the package versions on the repository with its own local list of packages already installed on the machine.

If an update of an installed package is available, the client will switch the status of the package to **NEED UPGRADE**. It will then install the software updates during the next **`upgrade`**.

Private key / Public key principle

Introduction

Like Android **APK** packages, WAPT packages are signed; a hash of the control sum of all the files included in the package is calculated.

This signing method guarantees the origin and integrity of the package.

Private key / Public key principle

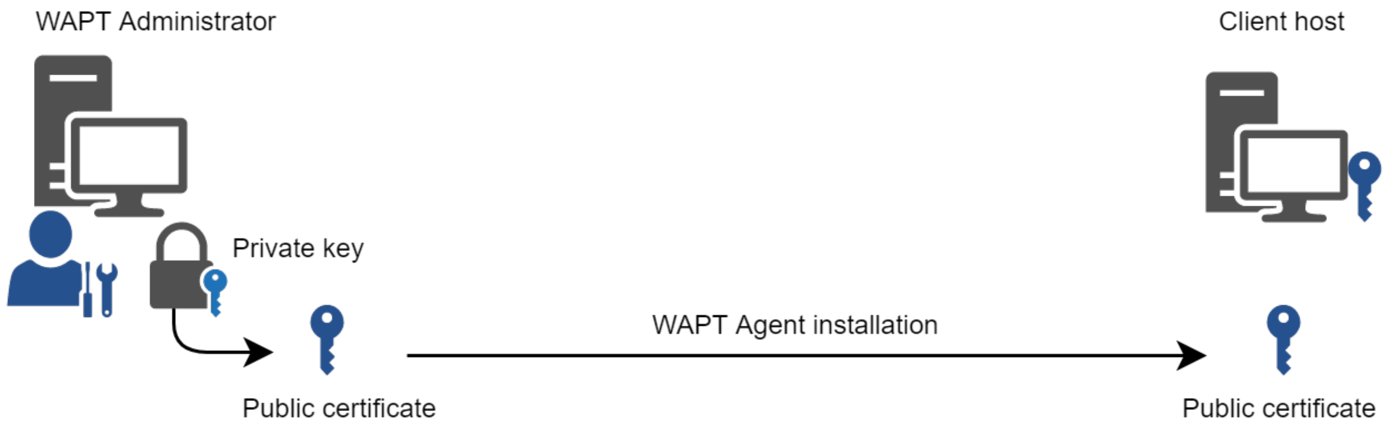


Fig. 4: Private key/ public certificate

To work properly, WAPT requires a private key/ public key pair (self-signed, issued by an internal *Certificate Authority* or commercially issued).

The **private key** will be used to **sign** WAPT packages whereas the **public key** will be distributed with every WAPT client so that WAPT agents may validate the files that were signed with the private key.

The different public keys will be stored in the WAPT subdirectory `ssl`. That folder can contain several public keys.

Package verification

When a WAPT package is downloaded, the WAPT agent (**`waptagent`**) will check the integrity of the package, and then check that the package has been properly **signed**.

If the WAPT package signature does not match any of the public keys located in `C:\Program Files (x86)\wapt\ssl`, the WAPT agent will refuse to install the package.

For more information, please refer to the documentation on *how the installation process integrity of a WAPT package is insured*.

The private certificate is important

Attention: The private key must **NOT** be stored on the WAPT Server, nor on any public or shared storage that could be accessed by non-authorized personnel. Indeed, WAPT security is based on keeping the private key **private**.

The private key must be stored in a safe place, because **she who has your key controls your network**!

Finally, to ensure maximum security, the private key can be secured in a smartcard or a cryptographic token that WAPT *Administrators* or *Package Deployer* will carry physically on them, using the smartcard or the token only when needed to sign a WAPT package.

Note: From WAPT 1.5 onward, the private key is protected with a password by default.

6.1.4 WAPT architecture and operating mode

Inventory/ information feedback

WAPT keeps a hardware and software inventory of each host.

That inventory is stored in a small database integrated in each WAPT agent.

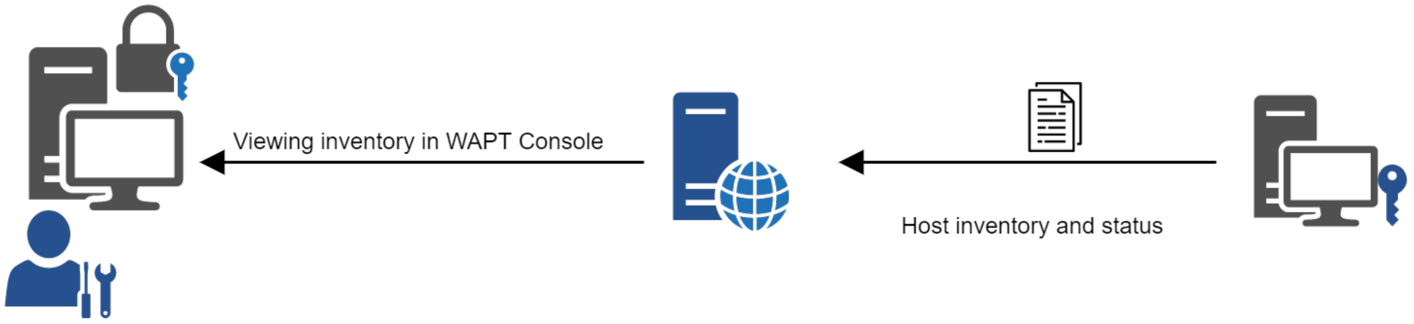
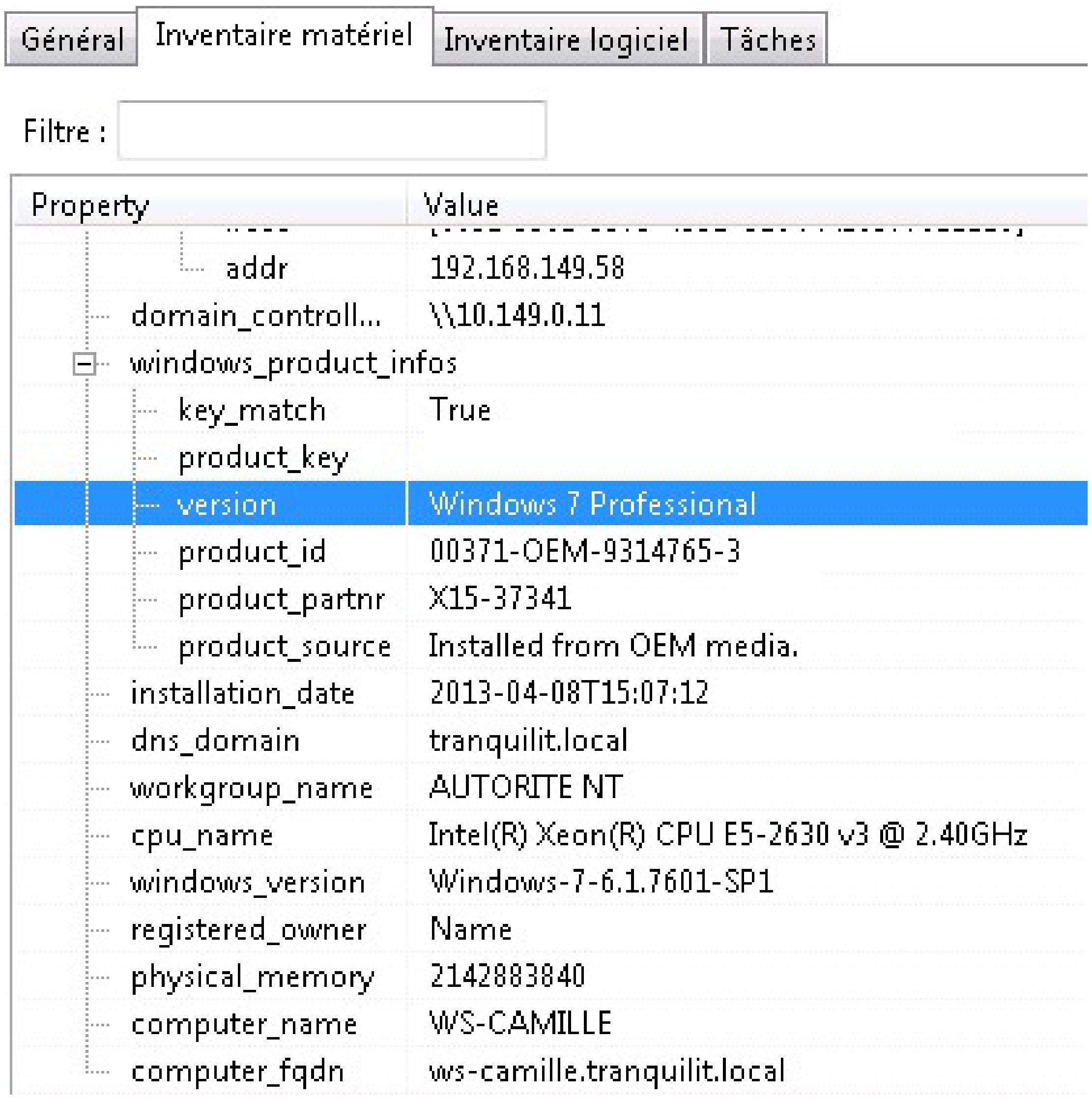


Fig. 5: Inventory feedback mechanism

- when first registering with the WAPT Server, the WAPT agent sends the entire inventory (BIOS, hardware, software) to the server;
- when the WAPT agent updates, the WAPT agent will report its inventory status to the WAPT Server;



Property	Value
addr	192.168.149.58
domain_controll...	\\10.149.0.11
windows_product_infos	
key_match	True
product_key	
version	Windows 7 Professional
product_id	00371-OEM-9314765-3
product_partnr	X15-37341
product_source	Installed from OEM media.
installation_date	2013-04-08T15:07:12
dns_domain	tranquilit.local
workgroup_name	AUTORITE NT
cpu_name	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
windows_version	Windows-7-6.1.7601-SP1
registered_owner	Name
physical_memory	2142883840
computer_name	WS-CAMILLE
computer_fqdn	ws-camille.tranquilit.local

Fig. 6: The inventory in the WAPT console

The central inventory allows you to filter hosts by their components, software or any other searchable argument.

Information feedback

The WAPT agents also report back their WAPT package status.

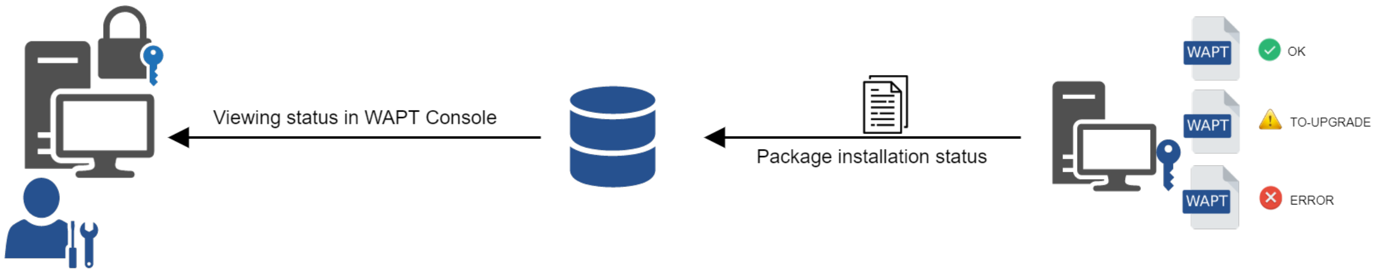
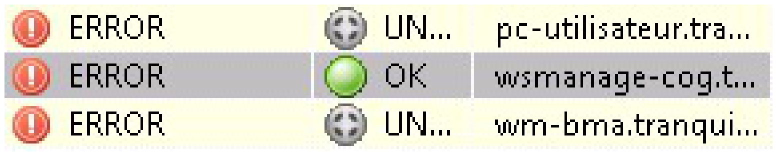


Fig. 7: Inventory feedback returned to the WAPT Server

In case of errors during package installation, the information will be reported to the WAPT Server. The host will then appear in **ERROR** in the console.



ERROR	UN...	pc-utilisateur.tra...
ERROR	OK	wsmanage-cog.t...
ERROR	UN...	wm-bma.tranqui...

Fig. 8: Packages with error status in the WAPT console

The *Administrator* can see the package returned in error in the console and fix the package accordingly.

For each **`upgrade`**, WAPT will try to install a new version of the package until no error status is returned.

Note: From WAPT 1.3.13 onward, WAPT agents sign their inventory before sending it to the WAPT Server.

For more information, please refer to *signing inventory updates*.

Complete diagram of the WAPT operating mechanism

We find here the common WAPT behavior, from duplicating a package from an external repository accessible on the Internet, to deploying it on network hosts.

Read the diagram clockwise:

- import packages from an external repository (or create a new package from scratch);
- test, validate, build and then sign the package;
- upload the package onto the main repository;
- packages are automatically downloaded by WAPT clients;
- packages are executed based on the selected method:

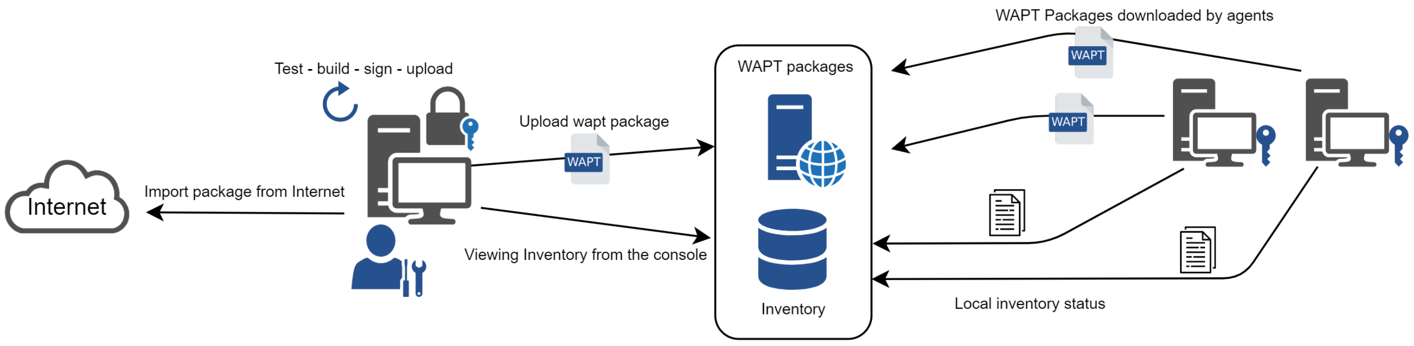

WAPT Packages downloaded by agents
Test - build - sign - upload
WAPT packages
Upload wapt package
WAPT
WAPT
WAPT
Internet
Import package from Internet
Viewing Inventory from the console
Inventory
Local inventory status

Fig. 9: WAPT general operating mode

- The *Administrator* forces the **`upgrade`**;
- the *User* chooses the right time for themself;
- a scheduled task launches the upgrade;
- the upgrade is executed when the machine shuts down;

- inventory information feedback;
- the updated inventory is reported in the console;

WAPT agent behavior with packages install / remove / session_setup / audit

A key concept that can be hard to understand is the behavior of a WAPT agent when installing a package and the considerations around it.

WAPT agent package installation can be split in SSS steps:

- package downloaded in agent cache;
- package unzip to temp folder;
- `setup.py` content is stored in WAPT agent database located in `C:\Program Files (x86)\wapt\db\waptdb.sqlite`;
- software installed from unzipped files:
- in case of success: downloaded package + unzipped files are deleted and status is sent to server;
- in case of failure: downloaded package is kept - unzipped files are deleted - error status sent to server;

That behavior is important as it has an impact on further actions.

For instance when removing a package the following steps are taken:

- `setup.py` content is retrieved from WAPT agent database located in `C:\Program Files (x86)\wapt\db\waptdb.sqlite`;
- software uninstall from registry **`UninstallString`** is executed;
- if defined, **`uninstall()`** function is executed from retrieved package source code;

Similar steps are reproduced when executing **`session_setup`** and **`audit`**.

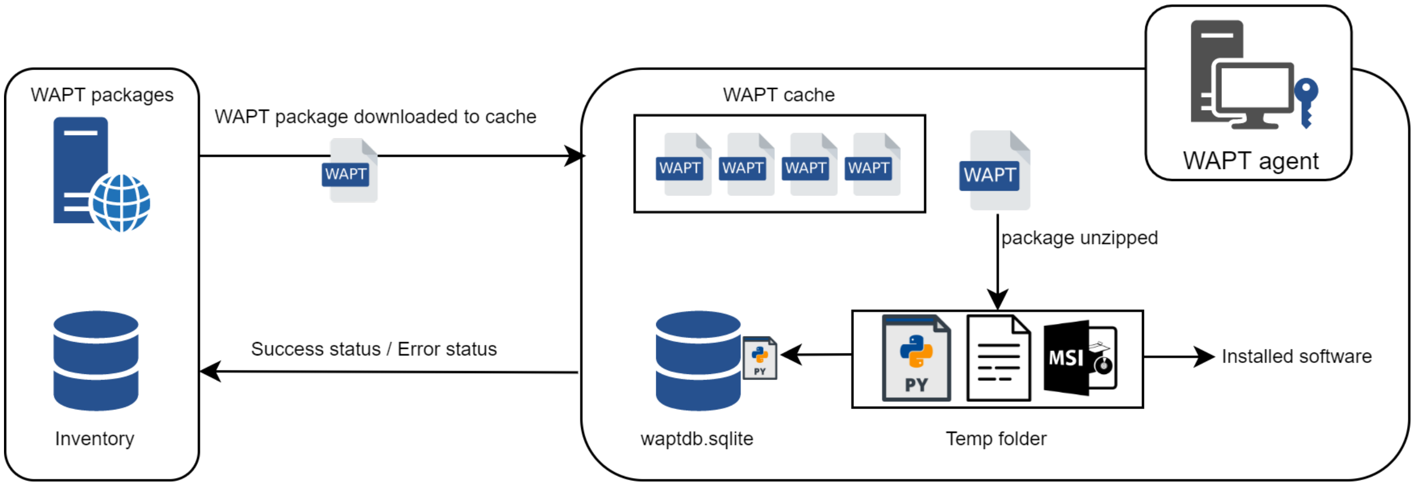


Fig. 10: WAPT install behavior

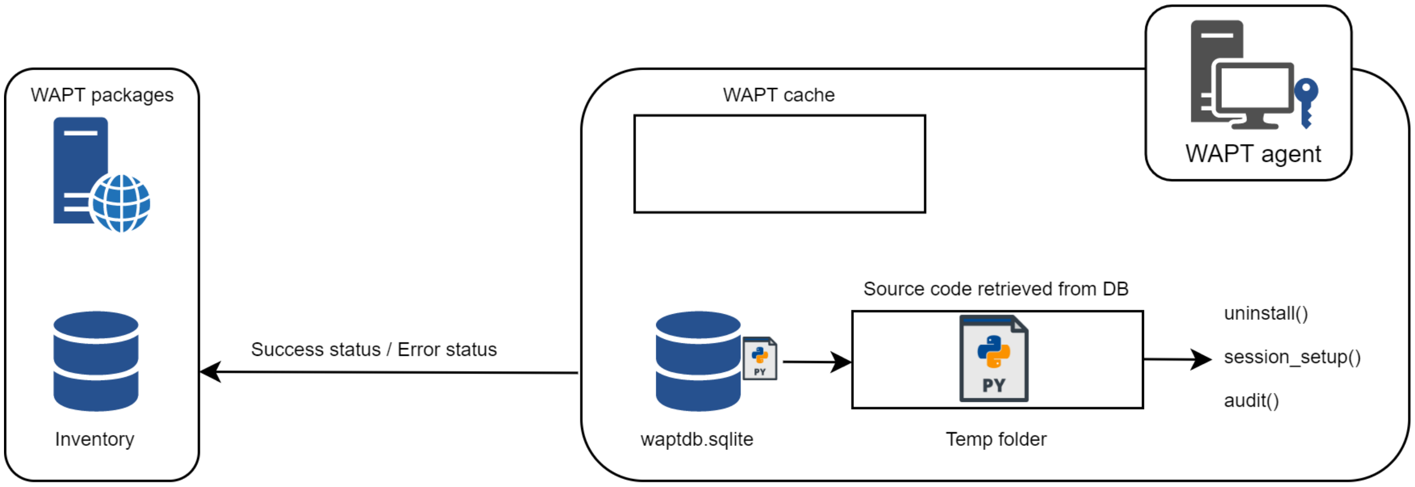


Fig. 11: WAPT behavior with uninstall / session_setup and audit

WAPT Server architecture

The WAPT Server architecture relies on several distinct roles:

- the *repository role* for distributing packages;
- the *inventory* and *central server* role for hardware and software inventory;
- the *proxy* role to relay actions between the WAPT console and the WAPT agents;

Repository role

First, the WAPT Server serves as a web repository.

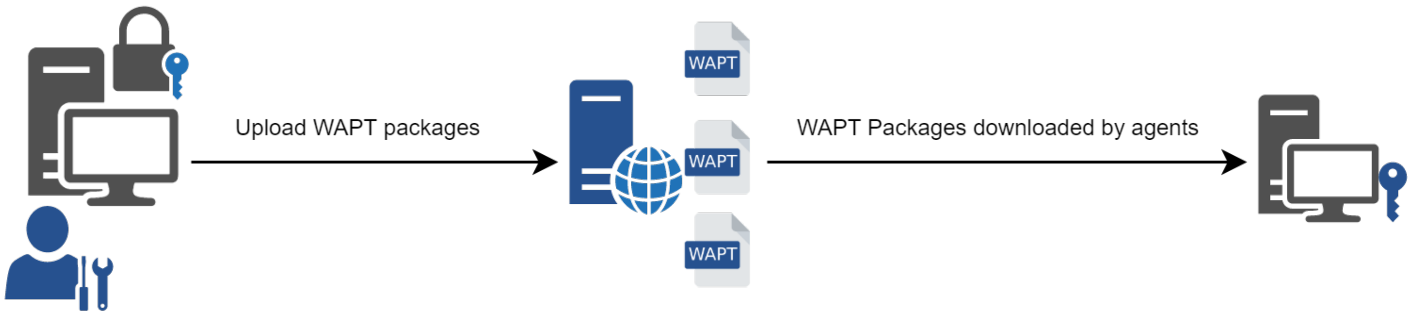


Fig. 12: WAPT repository mechanism

- the repository role is accomplished by a **`Nginx`** web server;
- the repository allows the distribution of WAPT packages, the installers for **`waptagent`** and **`waptsetup`**;
- WAPT packages are available via a web browser by visiting https://srvwapt.mydomain.lan/wapt;
- *host* packages are stored in a directory that is not accessible by default (https://srvwapt.mydomain.lan/wapt/wapt-host/);

Inventory server role

Second, the WAPT Server serves as an inventory server.

The inventory server is a passive service that collects information reported by WAPT agents:

- hardware inventory;
- software inventory;
- WAPT packages status;
- tasks status (*running*, *pending*, *error*);

Note: The WAPT service is not active in the sense that it only receives information from clients. As a consequence, if the inventory server fails, the inventory will recover by itself from inventory status reports received from the deployed WAPT agents.

In the Community version of WAPT, access to inventory data is only possible through the WAPT console.

WAPT **Enterprise** 1.7 will come with a *Business Intelligence* like web based reporting.

Proxy role

Third, the WAPT Server serves as a command relay proxy.

It acts as a relay between the WAPT management console and deployed WAPT agents.

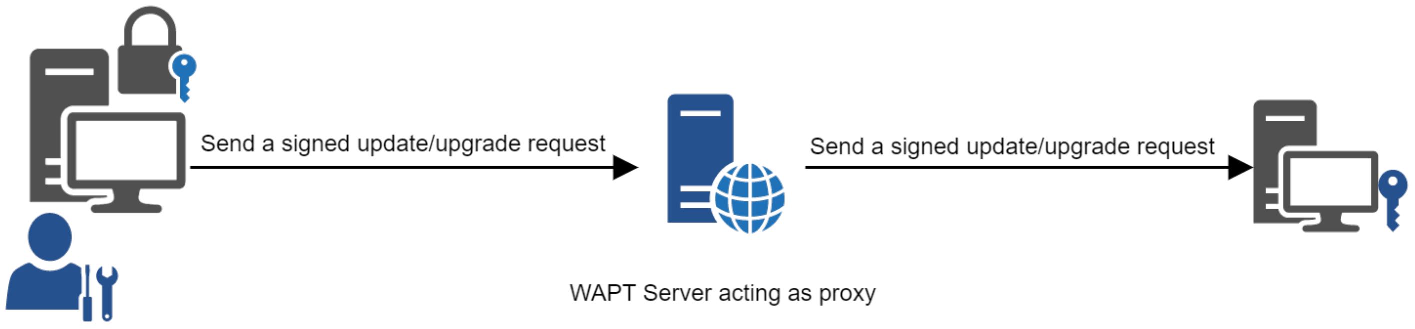

Send a signed update/upgrade request
Send a signed update/upgrade request
WAPT Server acting as proxy

Fig. 13: WAPT proxy mechanism

Note: Every action triggered on a WAPT agent from the server are signed with the *Administrator*'s private key. Without a valid private key, it is not possible to trigger remote actions on remote WAPT equipped devices. For more information on remote actions, please refer to *signing actions relayed to the WAPT agents*.

WAPT common interactions

update

When an **`update`** command is launched on an agent (from the console, via the command-line or via the WAPT tray), it is equivalent to ordering the agent to check the WAPT repository for new packages. By default, the WAPT agent will look for updates every two hours.

If the date of the `Packages` index file has changed since the last **`update`**, then the WAPT agent downloads the new `Packages` file (between 20 and 100k), otherwise, it does nothing.

The WAPT agent then compares the `Packages` file with its own local database.

If the WAPT agent detects that a package must be added or updated, it will switch the status of the host and package to *NEED-UPGRADE*.

It will not launch the installation of the package immediately. The WAPT agent will wait for an "**`upgrade`**" order to launch the upgrade.

upgrade

When we launch a command **upgrade** (from the WAPT console, using the command line, with a Windows scheduled task or manually with the WAPT tray), we ask the WAPT agent to install the packages with a *NEED-UPGRADE* status.

An **update** must come before an **upgrade**, otherwise the agent will not know whether updates are available.

Working principle of the WAPT agent

By default, the WAPT agent will trigger an **update**/ a **download-upgrade** at startup; after starting up, the WAPT agent will check every 2 hours to see whether it has something to do.

Packages to be installed will be downloaded and cached in the folder `C:\Program Files (x86)\wapt\cache`.

waptexit will launch an **upgrade** when the computer shuts down. An *Administrator* will also be able to launch an **upgrade** from the WAPT console.

If the WAPT Server is not reachable when upgrading, the WAPT agent will still be able to install cached packages.

Inventory updates will then be sent to the WAPT Server when network connectivity returns.

The 4 goals of the WAPT agent are therefore:

- to install a *base*, a *group* or a *unit* package if it is available;
- to remove obsolete packages;
- to resolve package dependencies and conflicts;
- to make sure all installed WAPT packages are up to date compared to the ones stored on the repository;
- to regularly update the WAPT server with its hardware status and the status of installed software;

6.1.5 WAPT package creation

WAPT language and development environment

WAPT is built using the Python language.

Any Rapid Application Development environment intended for Python development is suitable.

Tranquil IT has developed some useful WAPT specific plugins for the **PyScripter** IDE (https://sourceforge.net/projects/pyscripter).

Tranquil IT recommends using **PyScripter**, available with the *tis-waptdev* meta-package.

Principles of WAPT package development

The strength of Python

All the power of **Python** can be advantageously put to use.

Many libraries already exist in Python for:

- doing conditional loops (if . . . then . . . else . . .);
- copying, pasting, moving files and directories;

- checking whether files or directories exist;
- checking whether registry keys exist;
- checking access rights, modifying access rights;
- looking up information on external data sources (LDAP, databases, files, etc);
- etc ...

The power of 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:

- **what was complicated is made simple**;
- **what was simple is made trivial**;

Now, I want to *install my WAPT Server*!!

6.2 Security Principles

Here are documented the advanced security principles included in WAPT.

The reading of this portion of the documentation is not essential for your daily usage of WAPT; it is however recommended for you to better understand some architectural choices made by the developers of the software.

6.2.1 Presentation of the security principles in WAPT

Preamble and definitions

Attention: The WAPT service operates as a **privileged** system account.

Attention: From WAPT version 1.5, the default installation directory becomes `C:\Program Files (x86)\wapt`.

Hint: the sub-components **`wapttray`**, **`waptservice`** and **`waptexit`** of the WAPT agent may be optionally deactivated according to usage context.

Perimeter to secure

The elements to secure and that strictly concern WAPT are:

- **the WAPT Server** (*waptserver*);
- **the WAPT agents** (*wapt-get*) and its sub-components (*wapttray*, *waptservice* et *waptexit*);
- **the management console** (*waptconsole*);
- **the network communications** between these different components;

In complement to the elements listed above, an *Organization* that uses WAPT will have to choose and follow a methodology that is adapted to her use case:

- insure the safe provisioning of all other files that are to be incorporated into a WAPT package;
- develop the WAPT package python installation script so as to avoid any exploitable security or confidentiality defect;
- manage in a safe way the private keys for signing the packages;
- manage in a safe way the Autorities of Certification and Revocation for the SSL and HTTPS certificates;

The safe management of these complementary elements is excluded from the perimeter of this documentation.

Description of typical users

The following roles must be understood to evaluate the security principles incorporated in WAPT:

- **User**

  individual/ user of a WAPT equipped end-device (**Enterprise** and **Community**);
- **Package Deployer**

  individual with the ability to sign packages that **DO NOT** contain python code (generally *group*, *host* and *unit* packages) and to upload the package to the main repository (**Enterprise**);

- **Package Developer**

  individual with the ability to sign any package, may it include or not include python code, and to upload the package to the main repository (**Enterprise**);

- **SuperAdmin**

  all priviledged account in WAPT (**Community**);

- **Local Administrator**

  individual with local administration right of the WAPT equipped end-devices (**Enterprise** and **Community**);

Note: Depending on the context within this documentation, an *Administrator* will have the meaning of a *Package Deployer*, *Package Developer* or *SuperAdmin*.

Note: The *Users* that are members of the Active Directory security group **waptselfservice** are considered to be *Local Administrators* from the point of view of WAPT security.

Description of the sensitive assets in WAPT

By definition, a sensitive asset is a data (or a function) that is considered as having value to an attacker.

Its value is estimated according to several security criteria (also called security needs):

- availability;
- integrity;
- confidentiality;
- authenticity;

The sensitive assets to protect are as follows:

Sensitive assets A1: communications

Communications between the central WAPT Server and the WAPT agents, as well as the communications between the WAPT console and the WAPT Server are a sensitive asset and they must be protected.

Note: Need for securing the communications

- integrity;
- confidentiality;
- authenticity;

Sensitive asset A2: inventory data

The informations on the state of deployment of the packages, as well as hardware and software configurations of the end-devices are a sensitive asset and they must be protected.

Note: Security need of the inventory data

- integrity;
- confidentiality;

Sensitive asset A3: log journals

The logs generated by WAPT on the central server and by the agents are a sensitive asset and they must be protected.

Note: Security needs of historical logs

- availability;

Sensitive asset A4: configuration values

The configuration values (HTTPS server keys, database access configuration, server authentication configuration) are sensitive and they must be protected.

Note: Security needs of configuration values

- integrity;
- confidentiality;

Sensitive asset A5: WAPT executables on the end-devices

The WAPT executables installed on managed clients are a sensitive asset and they must be protected (i.e. the content of the `<WAPT>` directory that includes the binaries, the configuration files and the local database).

Note: Security needs of configuration values

- integrity;

Sensitive asset A6: authentication

Authentication to the administration console as well as the authentication of the clients on the WAPT Server are a sensitive asset and they must be protected (public key of each WAPT agent).

Note: Security need of the authentication

- integrity;
- confidentiality;

Description of hypotheses on WAPT's working environment

By definition, the hypotheses are statements on WAPT's usage context or its working environment.

The following hypotheses on WAPT's working environment must be considered:

Hypothesis H1: the Administrators and the Package Deployers are trained

The *Administrators* and the *Package Deployers* are trained on WAPT usage. In particular, they must insure that their logins, passwords and private keys are kept secret.

Hypothesis H2: the operating systems underlying WAPT are sane

WAPT's underlying operating systems implement adequate protection mechanisms that are configured according to good practice (confinement, access control, etc).

The underlying operating system are patched and up to date at the time of the installation of WAPT, they are free of viruses, trojan horses, etc.

Hypothesis H3: the binaries necessary for WAPT to operate are sane

All libraries and tools necessary to install WAPT are considered to be sane. Upon the WAPT agent receiving a request, the agent verifies that the request has been properly signed.

Hypothesis H4: the WAPT packages are built in a safe manner

The *Administrator* is responsible for insuring that the files to be incorporated into a WAPT package come from safe sources and are free of viruses, trojan horses, etc.

Hypothesis H5: the Users of the end-devices are not Local Administrators

A *User* must not have local administration rights on the WAPT equipped device. Otherwise, the *User* must be considered a *Local Administrator*.

In particular, a *User* must not have write access to WAPT's installation directory.

Hypothesis H6: the Local Administrators are trained

The *Local Administrator* of a device must be trained to use WAPT, or at minimum he must not make changes to files located in WAPT's installation folder.

Description of threats on WAPT's sensitive assets

By definition, a threat is an action or an event susceptible to bring prejudice to the security of the WAPT equipped device.

The threat agents to be considered for evaluating security in WAPT are as follows:

- **Unauthorized entities**: it is a human attacker or en entity that interacts with WAPT without legitimately having access to it.

Note: The *Administrators* and the *Local Administrators* are not considered to be attackers.

The threats bearing on WAPT's sensitive assets defined above are as follow:

Threat T1: installation of an unsafe software by an unauthorized entity

This threat corresponds to an attacker that would be able to use a component of the WAPT agent to permanently install a malicious application, or to remove or deactivate a security component on the WAPT equipped device.

Threat T2: modification of configuration values by an unauthorized entity

The threat corresponds to an attacker that would be able to modify a configuration element of WAPT that had been previously defined by a legitimate WAPT *Administrator*.

Threat T3: illegitimate access by an unauthorized entity

This threat corresponds to an attacker that would be able to recover the login credentials of an *Administrator*, or bypass the authentication mechanism in such a way to access or alter a sensitive asset stored on the server. It also corresponds to an attacker being able to impersonate a WAPT agent.

Threat T4: network listening by an unauthorized entity

This threat corresponds to an attacker being able to intercept and gain knowledge of network traffic between the WAPT agents and the server hosting WAPT.

Threat T5: modification of network traffic by an unauthorized entity (type *Man In The Middle*)

This threat corresponds to an attacker being able to alter network traffic between the agents and the server hosting WAPT, or between the console and the WAPT Server.

Description of WAPT's security functions

By definition, security functions are the set of technical measures and mechanisms implemented to protect in a proportionate way the sensitive assets against identified threats.

Security function F1: access authentication

Security function F1A: authentication of a device on initial registration in the WAPT database

New in version 1.5.

Note: risks avoided

- the registering of an illegitimate device in the database;
- a denial-of-service attack by overloading the database;
- the insertion of a fraudulent inventory in the database;

Solution implemented

To exist in the database and thus to appear in the management console, a device must register with the WAPT Server using the **`register`** command.

The **`register`** command may be executed automatically when installing or updating the WAPT agent if the device has a Kerberos machine account that is correctly registered in the *Organization*'s Active Directory domain.

If the device does not present to the WAPT Server a valid Kerberos ticket, then the **`register`** fails;

Note: The Kerberos registration method assumes that the Active Directory server is responsive at the time of launch of the **`register`** command.

Security function F1B: verification of server HTTPS certificates by the WAPT agents

New in version 1.5.

Note: risks avoided (notably MITM)

- the sending of sensitive informations to an illegitimate and unauthorized WAPT Server;
- the recovery of sensitive informations by an unauthorized entity;
- the display of fake information in the management console of the *Administrator*;
- an incorrect date to be sent upon a HEAD request, thus preventing future upgrades (request for a modified file date);
- sending the WAPT console password to an illegitimate and unauthorized WAPT Server;

Solution implemented

For the secured version of WAPT to work correctly:

- an option for verifying the server HTTPS certificate is introduced in `C:\Program Files (x86)\waptwapt-get.ini` on the WAPT agents that will **force the verification of the server certificate by the WAPT agents**;
- an option for verifying the server HTTPS certificate is introduced in `C:\Program Files (x86)\waptwapt-get.ini` on the WAPT agents that will force the verification of the server certificate by the **WAPT console**;

Technically, it may be implemented in two ways:

- by using a certificate verification tool implemented in the configuration file of WAPT's **`Nginx`** web server; this method is typically provided by a *Certificate Authority* that is trusted by your network;
- by using the *certificate pinning* method, which consists of providing the WAPT agent a short list of trusted certificates that will be stored in `C:\Program Files (x86)\wapt\ssl\server`;

Security function F1C: no listening port on the WAPT agents

New in version 1.5.

Note: risks avoided

- an unauthorized entity using an open port fraudulently;

Solution implemented

The connections to the WAPT Server are initiated exclusively by the agents, and the forced immediate actions are relayed through a permanent websocket initiated by the WAPT agent (**`update`**/ **`upgrade`**/ **`install`** . . .).

Note: if HTTPS is activated, then the WAPT agent checks that the websocket is connecting to the rightful server.

Security function F1D: signature of inventory return states

New in version 1.3.12.13.

Note: risks avoided

- an unauthorized entity sending a fake inventory for a device that rightfully exists in the database;

Solution implemented

- On the first **register**, each device generates a key/ certificate pair that is stored in `C:\Program Files (x86)\wapt\private`, only accessible in read-only mode to *Local Administrators*. Once the device has successfully registered, the public key is sent to the WAPT Server;
- When the inventory is updated, the new inventory status is sent along with the private key of the device. The new inventory is then decrypted with the public key stored in the database;
- The server will refuse any inventory that is signed with a wrong key;

Security function F1E: verification of authorizations before launching of WAPT commands

Note: risks avoided

- avoid the execution of sensitive tasks on WAPT clients by unauthorized entities;

Solution implemented

The *Users* interact with WAPT through WAPT user interfaces (**wapt-get** in command line interface, **wapttray**, **waptexit**, **waptselfservice**).

The user interfaces then delegate the execution of the desired tasks to the local WAPT service running as system account.

The following actions do not require to be authenticated with the WAPT service:

- `wapt-get update` (update the available list of packages);
- `wapt-get upgrade` (launch waiting upgrades);
- `wapt-get download-upgrade` (download waiting upgrades);
- `wapt-get clean` (remove packages left in cache after installation);
- stop any running WAPT task;
- stop/ reload the WAPT service;

The other actions require the *User* be authenticated and the *User* either be a member of the **waptselfservice** Active Directory security group, or be a *Local Administrator*, they are:

- `wapt-get install`: requests the WAPT agent to install a WAPT package flagged as **MISSING**;
- `wapt-get remove`: requests the WAPT agent to remove a package;

- `wapt-get forget`: requests the WAPT agent to forget the existence of a previously installed WAPT package without removing the software or the configuration;

Security function F2: protecting the integrity of the installation process of WAPT packages

Security function F2A: signature of WAPT packages

Note: risks avoided

- to avoid an unauthorized entity modifying the content or the behavior of a WAPT package;

Solution implemented

- when an *Administrator* or a *Package Deployer* builds a WAPT package, the file `manifest.sha256` is created that lists the control sums of all files in the package;
- 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`;
- the whole is then compressed and suffixed with a *.wapt* extension;
- 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`;
- the WAPT agent then checks that the certificate or the chain of certificates in `certificate.crt` has been signed with a key matching one of the certificates present in the folder `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;
- if one of these steps does not pass, this means that a file has been modified/ added/ removed. The execution of the `setup.py` is then canceled.
- the altered package is then deleted from the local cache and the event is journalized in the logs of the agent;

Note: Since the version 1.5 of WAPT, the format of the `manifest` file has changed from *sha1* to *sha256*.

Security function F2B: signature of the attributes in the control files

New in version 1.4.

Note: risks avoided

- an unauthorized entity modifying WAPT dependencies on WAPT equipped devices by falsifying `https://waptserver/wapt/Packages`;

Solution implemented

When a WAPT package is signed, the sensitive attributes of the package are listed inside the **signed_attributes** attribute of the control file.

Note: Example of a *signed_attributes* list:

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*,

The attributes listed in *signed_attributes* are signed with the private key of the *Administrator* and stored in the attribute *signature* of the `control` file.

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

On the WAPT Server, the index `Packages` is regenerated when the **`wapt-scanpackages`** command is triggered by adding or removing a package.

The WAPT Server extracts from each package the certificate of the signer and adds it in the `Packages` zip file in the directory `ssl`. Each certificate is named after its hexadecimal encoded fingerprint.

When the WAPT agent launches an **`update`**, it downloads the `Packages` index file that contains the signed attributes of all available packages and the certificates of the signers.

If the signer's certificate is approved, which means that the certificate has been signed by a Trusted *Certificate Authority* or that the certificate itself is trusted, AND if the signer's certificate can verify the attributes' signature, the package is added to the index of available packages. Otherwise it is ignored.

Security function F2C: access restriction to the installation folder of the WAPT agent

Note: risks avoided

- an unauthorized entity modifying the behavior of a WAPT agent;

The installation folder `C:\Program Files (x86)\wapt` is accessible in read-write mode:

- to the *Local Administrators* by direct access to the installation folder of the WAPT agent on the device;
- to the *Administrators* through the deployment of WAPT agent upgrades;

Neither the *Package Deployers*, nor the *Users* have write-access to the WAPT agent's installation folder.

Security function F2D: total access restriction to the folder storing the key / certificate for inventory signing

Note: risks avoided

- an unauthorized entity falsifying an inventory status update;
- an unauthorized entity impersonating the identity of a WAPT equipped device;

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: This method for storing the key and the certificate results from a technical design choice that says that the WAPT equipped device would embed any and all information related to itself.

Security function F3: securing the communications between the different components of WAPT

Security function F3A: signature of immediate action calls sent to the WAPT agents

New in version 1.5.

Note: risks avoided

- an unauthorized entity sending falsified requests to the WAPT agents;

Solution implemented

The following commands are signed by the WAPT Server before being relayed to the targeted WAPT agents via their Websockets:

- `wapt-get install`: requests the WAPT agent to install a WAPT package flagged as **MISSING**;
- `wapt-get remove`: requests the WAPT agent to remove a package;
- `wapt-get forget`: requests the WAPT agent to forget the existence of a previously installed WAPT package without removing the software or the configuration;
- `wapt-get update-status`: requests the WAPT agent to send its current inventory status to the WAPT Server;
- `wapt-get upgrade`: requests the WAPT agent to execute a package flagged as **NEED UPGRADE**
- `wapt-get update`: requests the WAPT agent to update the list of available packages;

All the attributes in the requests for immediate action are signed:

- the device's *UUID*;
- the action (ex: **`install`**);
- the arguments (ex: tis-firefox);
- the timestamp of the requests;

The certificate matching the signature is passed along:

- upon receiving a request, the WAPT agent verifies that the request has been properly signed;
- the agent will the verify that the timestamp is within a one minute delay window;
- ultimately, the agent will verify that the certificate is authorized to launch actions;

6.2.2 Presentation of cryptographic processes

Date	Jan 10, 2023
Written by	Hubert TOUVET
Applicable for WAPT	>= 1.5.0.17
Version of the Document	1.5.0.17-0

Cryptographic processes are used in the following activities:

- signature and verification of the **files contained in a package**;
- signature and verification of the **attributes of a package**;
- signature and verification of **instantaneous actions** on the WAPT Clients;
- signature of inventories and **status of WAPT clients**;
- authentication of the WAPT client Websocket connections on the server;
- HTTPS communication between the WAPT clients and the WAPT Server;
- HTTPS communication between the WAPT console and the WAPT Server;
- HTTPS communication between the WAPT clients and the WAPT repositories;

Folders and files referenced in this document

- `<WAPT>`: WAPT installation folder. By default `%Program Files (x86)%WAPT`;
- `<WAPT>wapt-get.ini`: WAPT client configuration file (**`wapt-get`** and **`waptservice`**);
- `<WAPT>ssl`: default directory for signed actions and trusted certificates;
- `<WAPT>sslserver`: default directory for storing HTTPS server certificates (pinning);
- `<WAPT>private`: default certificate directory for signing the inventory and the Websocket connections;
- `%LOCALAPPDATA%waptconsolewaptconsole.ini`: configuration file for the console and package development actions for the **`wapt-get`** tool;
- `%appdata%waptconsolessl`: default trusted certificate directory for importing packages from an external repository (i.e. *package templates*);

Actors

- **Organization**

  it is the realm of responsibility within which WAPT is used;

- **Certificate Authority**

  it is the entity that keeps the keys that have been used to sign certificates for the *Package Developers* and the HTTPS servers;

- **Administrators**

  they have a personal RSA key and a certificate that has been signed by the *Certificate Authority* of the *Organization*; they also have a login and a password for accessing the WAPT console;

- **WAPT clients**

  realm of Windows PCs that the *Organization* has allowed the *Administrators* to manage with WAPT. The Windows clients are joined to the *Organization*'s Active Directory domain;

- **Internal WAPT repositories**

  it is one or several Linux/ Nginx servers that deliver signed WAPT packages to WAPT clients using the HTTPS protocol;

- **WAPT Server**

  it is the Linux / Nginx/ PostgreSQL /WAPT server that the *Organization* uses to keep the inventory of WAPT equipped devices.

  By default, the WAPT Server also plays the role of an internal WAPT Repository. The WAPT Server has a machine account in the *Organization*'s Active Directory.

- **external WAPT repositories**

  it is a public WAPT repository that the *Package Developers* may use to import packages designed by other *Organizations*, under the condition that they check the adequacy of the WAPT package in regards the internal policies on security and safety;

- **Active Directory Server**

  server that manages the *Organization*'s domain;

Summary of crypto modules present in WAPT

On the client side of WAPT (WAPT 1.5.0.12):

- **Python 2.7.13** standard *ssl* module linked on `OpenSSL 1.0.2j 26 Sep 2016`: used for the HTTPS connections between the WAPT clients and the WAPT server;
- **cryptography==1.9** linked on `openssl 1.1.0f`: used for all RSA crypto operations such as key generations, X509 certificate generations and signature verifications;
- **kerberos-sspi==0.2** and **requests-kerberos==0.11.0**: used for authenticating the WAPT client on its first registering on the WAPT Server;
- **pyOpenSSL==17.0.0**: used to recover the WAPT Server certificate chain;
- **certifi==2017.4.17**: used as base for the Root Authorities certificates;
- **Openssl 1.0.2l** dll: used in waptcommon.pas written with the FPC Indy library and the TIdSSLIOHandlerSocketOpenSSL class;

On the server side of WAPT:

- **nginx/1.10.2**: configured for TLS1.2, cipher 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
- **python 2.7.5** standard *ssl* module linked on `OpenSSL 1.0.1e-fips 11 Feb 2013`;
- **cryptography==1.9** linked on `OpenSSL 1.0.1e-fips 11 Feb 2013`: used for all RSA crypto operations such as key generations, X509 certificate generations and signature verifications;

Key and certificate management for the Administrator

Context

Packages and actions done by an *Administrator* are signed so that only Trusted Administrators are authorized to manage the devices.

The WAPT *Administrator* holds:

- a private 2048 bit *RSA* key that has been encrypted by the aes-256-cbc algorithm;
- a *X509* certificate signed by an *Certificate Authority* trusted by the *Organization*;

Note: The process for creating the keys and signing, distributing and revocating the certificates are of the responsibility of the *Organization* using WAPT; that process is beyond the functional perimeter of WAPT.

However, to make the testing of WAPT easy, WAPT offers a function to generate a RSA key and its corresponding X509 certificate:

- the generated RSA key is 2048bit long, encrypted with aes-256-cbc, encoded in PEM format and saved with a *.pem* extension;
- the certificate is either self-signed, or signed by a Trusted Authority from whom we have received a key and a PEM formated certificate;
- if the certificate is self-signed, then its *KeyUsage* attribute contains the *keyCertSign* flag;
- if the *Administrator* is authorized by the *Organization* to sign packages that contain python code (the presence of a `setup.py` file is detected in the package), the *extendedKeyUsage* attribute of the certificate contains the **CodeSigning** flag;
- the *X509* certificate is encoded and handed over to the *Administrator* in PEM format with a *.crt* extension;

Validity of the Administrator's certificate

For WAPT version up to 1.5.0.12, WAPT agent does not verify the revocation state of the *Administrator*'s certificate during the process of verifying the package, the attributes or the actions of the *Administrator*.

It only checks the dates of validity (*notValidBefore*/ *notValidAfter* attributes). The certificate is valid if (**Now** >= *notValidBefore* and **Now** <= *notValidAfter*).

Authorizing the Administrator's certificate to sign a package

The certificate used by the WAPT console to sign packages and actions is defined with the *personal_certificate_path* parameter in the section `[global]` of the file `%LOCALAPPDATA%waptconsolewaptconsole.ini`.

WAPT asks the *Administrator* for his password and then searches a private key (encoded in PEM format) that matches a certificate amongst the `.pem` files in the directory containing the certificates.

When signing a package, WAPT will refuse the certificate if the package contains a `setup.py` file and the certificate is not a *CodeSigning* type.

Managing the WAPT agent's key and certificate

Context

The WAPT client (`waptservice`) uses RSA keys and X509 certificates to interact with the WAPT Server.

The WAPT client certificate is used in the following situations:

- when updating the WAPT client status on the server (`update_server_status`) **signing informations**;
- when the WAPT agent establishes a Websocket with the server (`waptservice`) **signing the WAPT client UUID**;

First emission and later update of the WAPT agent's certificate

- on finishing the installation process of the WAPT agent on the device, the WAPT agent automatically registers itself on the WAPT Server by sending a Kerberos authenticated HTTPS request that uses the TGT of the machine account.

  The WAPT agent uses Windows kerberos APIs implemented with **`kerberos-sspi`** and **`requests-kerberos`** python modules.

Note: this process works if and only if the device is joined to the Windows domain for which the WAPT Server is configured.

- if the key and the certificates have not yet been generated, or if they do not match the current *FQDN* of the device, the WAPT agent generates a self-signed RSA key and X509 certificate with the following parameters:
  - the key is 2048 bit RSA encoded in PEM format and stored in the file `<WAPT>private<device FQDN>.pem`;
  - the generated certificate has the following attributes:
    * *Subject.COMMON_NAME* = <device FQDN>;
    * *Subject.ORGANIZATIONAL_UNIT_NAME* = name of the *Organization* registered in the WAPT client's Windows registry;
    * *SubjectAlternativeName.DNSName* = <device FQDN>;
    * *BasicConstraint.CA* = True;
    * *validity* = 10 years;
    * *serialnumber* = random;
  - the certificate is saved in the `<WAPT>private<device FQDN>.crt`;

Note: Only machine accounts and *Local Administrators* have access to the `<WAPT>private` directory because specific ACLs have been applied upon first installation of the WAPT agent on the device.

- the inventory and the WAPT agent status updates are sent to the WAPT Server over POST HTTPS requests;
- the POST HTTPS requests are authenticated by adding two specific headers:
  - *X-Signature*:
    * JSON encoded BLOB (Binary Large Object) of inventory or status informations;
    * signature of the JSON file with the private key of the WAPT Client: *sha256* hashing and *PKCS#1 v1.5* padding;

    * *base64* encoding of the signature;

  - *X-Signer*: *Subject.COMMON_NAME* or UUID of the WAPT Client.

- after having initially authenticated the WAPT client with Kerberos, the WAPT Server receives the certificate sent by the Client and stores it in the table *hosts* of its inventory in PEM format (column *host_certificate*).

Note: If the device is renamed, the key/ certificate pair is regenerated.

When the WAPT agent will update its status with the WAPT Server, the POST request will be refused because the remote device is registered in the database with another certificate.

The device will then retry to **`register`** with the WAPT Server using Kerberos; then the new certificate will be saved in the database.

Deploying certificates of Authorities of Certification to verify packages and validate actions on Clients

PEM formated certificates are stored in files with *.crt* or *.pem* extensions in the directory defined with the *public_certs_dir* parameter in the file `<WAPT>wapt-get.ini`. They are reputed to be **trusted certificates**.

The *public_certs_dir* parameter is initialized by default to be `<WAPT>ssl`.

Authority certificates are deployed when the WAPT agents are first deployed.

From the console, the *Administrator* compiles a personalized installer to be deployed by *GPO* or other means on target devices.

The WAPT console includes in its personalized installer the certificates present in the `<WAPT>ssl` directory of the PC on which the installer is being compiled.

The *Administrator* must insure to save in `<WAPT>ssl` only the certificates of Authorities that are strictly necessary before launching the compilation of the installer.

New or updated certificates of *Certificate Authority* for the verification of packages and the validation of actions may also be deployed a posteriori with an Active Directory GPO or a WAPT package.

Deploying certificates of Authorities of Certification for the HTTPS communication between the WAPT clients and the WAPT Server

The WAPT service (**`waptservice`**) and the command line tool **`wapt-get`** exchange with the WAPT Server to send its inventory (**`register`**) and the package deployment status (**`update-status`**).

These two types of connections verify the HTTPS certificate of the WAPT Server.

verify_cert parameter in section `[global]` in `<WAPT>wapt-get.ini`:

- *verify_cert* = 1

  this method will only work well if the HTTPS server is configured to send its certificate and the intermediary certificates upon initialization of the TLS connexion.

- *verify_cert* = <path to .pem>

  check the HTTPS certificate using the indicated bundle of certificates. All the certificates of the intermediary Certificate Authorities must be bundled in a *.pem* formated file;

- *verify_cert* = 0

  do not verify the HTTPS certificate of the server;

Conventionally, the approved bundle of certificates from the *Certificate Authority* is stored in the `<WAPT>sslserver` directory.

The WAPT console includes a function to facilitate the initial recovery of the server certificate chain. The function stores it in *.pem* format in `<WAPT>sslserver<server FQDN>`.

The *Administrator* is responsible for insuring that the recovered certificate chain is **authentic**.

During the compilation of the WAPT agent installer, the certificates or the bundle of certificates is incorporated into the installer.

When the installer is deployed on the WAPT clients, the bundle is copied in `<WAPT>sslserver` and the *verify_cert* parameter in section``[global]`` in `<WAPT>wapt-get.ini` is filled out to indicate the path to the bundle.

HTTPS communication between the WAPT clients and the WAPT repositories

Deploying certificates of Authorities of Certification

The HTTPS exchanges between the WAPT agent and the main repository and between the WAPT agent and the WAPT Server use the same methods.

The WAPT agent uses the same bundle of certificates to communicate in HTTPS with the main repository, with the WAPT Server, and with the secondary repositories.

The WAPT client

The HTTPS connection is implemented with **`requests`**, **`urllib3`** et **`ssl`** python modules.

The certificate emitted by the repository HTTPS server is verified with the **`urllib3.contrib.pysopenssl.PyOpenSSLContext`** and **`urllib3.util.ssl_wrap_socket`** python modules.

Websocket communications between the WAPT clients and the WAPT Server

To allow immediate actions on the WAPT clients, the WAPT service deployed on the clients establishes and maintains a permanent Websocket with the WAPT server.

This connection is TLS encrypted and uses on the client side the same bundle of certificates as the HTTPS connexion from the WAPT client to the WAPT Server.

Communications between the WAPT console and the WAPT Server

Deploying certificates of Authorities of Certification

verify_cert parameter in section `[global]` in file `%LOCALAPPDATA%waptconsolewaptconsole.ini`:

- *verify_cert* = 1

  this method will only work well if the HTTPS server is configured to send its certificate and the intermediary certificates upon initialization of the TLS connexion.

- *verify_cert* = <path to .pem>

  check the HTTPS certificate using the indicated bundle of certificates. All the certificates of the intermediary Certificate Authorities must be bundled in a *.pem* formated file;

- *verify_cert* = 0

  do not verify the HTTPS certificate of the server;

Conventionally, the approved bundle of certificates from the *Certificate Authority* is stored in the `<WAPT>sslserver` directory.

The WAPT console includes a function that facilitates the initial recovery of the server certificate chain and that stores it in *.pem* format in the `<WAPT>sslserver<server FQDN>`.

The *Administrator* is responsible for insuring that the recovered certificate chain is **authentic**.

It is also possible to recover the server certificate chain and fill out the *verify_cert* parameter with the command line `wapt-get enable-check-certificate`.

Deploying the certificates of Authorities of Certification to verify packages imported in the main repository

Context

In the WAPT console, tab *Private Repository*, a button *Import from Internet* allows to download a package from an external repository whose address is provided with the *repo-url* parameter in the section `[wapt_templates]` of `%LOCALAPPDATA%waptconsolewaptconsole.ini`.

A checkbox *Verify Package Signature* insures that the imported package has been signed with a trusted certificate.

Certificates from Trusted Authorities

The certificates from Trusted Authorities present in the directory specified with the *public_certs_dir* parameter in section `[wapt_templates]` in file `%LOCALAPPDATA%waptconsolewaptconsole.ini` are considered to be trusted.

If the parameter is not explicitly mentioned, it is initialized at `%appdata%waptconsolessl`.

This directory is not automatically populated by WAPT. It is the responsibility of the *Administrator* to copy/ paste into it the `PEM` files of other trusted *Administrators* or the certificates from trusted Certificate Authorities.

The Certificates from Trusted Authorities are encoded in *.pem* format and stored in files with *.pem* or *.crt* extensions. It is possible to store several certificates in each `.crt` or `.pem` file.

It is not necessary to have the complete chain of certificates, WAPT will accept the signature of a package as long as:

- the certificate of the package is also included in the *public_certs_dir* directory. The matching is done using the fingerprint of the certificate;
- the certificate of the Authority that has signed the certificate of the package is included in the *public_certs_dir* directory. The matching is done using the *issuer_subject_hash* attribute of the certificate. The signature of the certificate is done using the **`x509.verification.CertificateVerificationContext`** class;

Process for signing a package

The process for signing a package is launched with the following actions:

- action `wapt-get.exe build-upload <directory>`;
- action `wapt-get.exe sign-package <path-to-package-file.wapt>`;
- shell command `wapt-signpackage.py <WAPT-package-list>`;
- saving a *host* package in the WAPT console;
- saving a *group* package in the WAPT console;
- importing a package from an external repository;
- creating a package with the MSI setup wizard;

Initial parameters

- ZIP file of the package;
- *.pem* formated RSA private key of the certificate holder (encrypted with OpenSSL's *aes-256-cbc* algorithm if the key has been created in the WAPT console);
- *X509* certificate of the certificate holder matching the private key;
- if the package to be signed contains a `setup.py` file, then the *X509* certificate must have the *advanced Key Usage* extension *codeSigning (1.3.6.1.5.5.7.3.3)*;

Signature of the attributes in the control file

The `control` file defines the metadata of a package and in particular its name, its version, its dependencies and its conflicts. It is the identity card of the WAPT package.

These metadata are primarily used by the WAPT agent to determine whether a package must be upgraded, and what packages must be first installed or removed.

The package attributes are therefore signed to insure the integrity and the authenticity of the WAPT package.

Process steps:

- the attributes *signed_attributes*, *signer*, *signature_date*, *signer_fingerprint* are added to the structure of the `control` file:
  - *signed_attributes*: comma separated list of the names of the attributes;
  - *signer*: CommonName of the certificate holder;
  - *signature_date*: current date and time (UTC) in '%Y-%m-%dT%H:%M:%S format;
  - *signer_fingerprint*: hexadecimal encoded `sha256` fingerprint of the fingerprint obtained with the **`fingerprint`** function included in the **`cryptography.x509.Certificate`** class;
- the attributes of the control structure are JSON encoded;
- the resulting JSON BLOB is signed with *sha256* hashing and *PKCS#1 v1.5* padding;
- the signature is base64 encoded and stored as a *signature* attribute in the `control` file;

Signing the files of the package

- the `control` file attributes are signed and serialized in JSON format. The result is stored in the `<WAPT>control` file of the WAPT package;
- the PEM serialized X509 certificate of the certificate holder is stored in the `<WAPT>certificate.crt` file of the WAPT package;
- the *sha256* fingerprints of the all files contained in the WAPT package are hexadecimal encoded and stored as a JSON list [(filename,hash),] in the `<WAPT>manifest.sha256` file in the WAPT package;
- the content of the file `<WAPT>manifest.sha256` is signed with the private key of the *Administrator* (2048 bit RAS key), *sha256* hashed and *PKCS#1 v1.5* padded:
  - the signature process relies on the signing function of the **`cryptography.rsa.RSAPrivateKey.signer`** class;
  - **`cryptography.rsa.RSAPrivateKey.signer`** relies on the OpenSSL functions of **`EVP_DigestSignInit`**;
- the signature is base64 serialized and stored in the file `<WAPT>signature.sha256` of the WAPT package;

Verifying the signature of a package attributes

The verification takes place:

- when the index file of available packages on the WAPT client is updated from the `Packages` index file on the repository;
- when a package signature is verified (installation, download) when not in *development* mode, i.e. if the installation takes place from a ZIP file and not from a development directory;

The verification consists of:

- reading the control attributes from the WAPT package's `control` file;
- recovering the X509 certificate from the certificate holder from the WAPT package's `certificate.crt` file;
- decoding the base64 formated signature attribute;
- constructing a JSON structure with the attributes to be signed (such as defined in the **`PackageEntry`** class);
- verifying if the public key of the holder's certificate can verify the hash of the JSON structured list of attributes and the signature of the `control` file, using *sha256* hashing and *PKCS#1 v1.5* padding;
- verifying whether the certificate is trusted (either it is present in the list of trusted certificates, or signed by a Trusted *Certificate Authority*);

In case we must verify the attributes without having the WAPT package on hand, we recover the list of certificates of potential certificate holders from the `Packages` index file on the WAPT repository. The certificates are named `ssl/<hexadecimal formated certificate fingerprint>.crt`.

An attribute in the WAPT package's `control` file specifies the fingerprint of the `control` file's certificate holder.

Verifying the signature of a WAPT package

The verification takes place:

- when installing a package on a WAPT Client;
- when editing an existing package;
- when importing a package from an external repository (if the checkbox is checked in the WAPT console);

The verification consists of:

- recovering the X509 certificate from the certificate holder from the WAPT package's `certificate.crt` file;
- verifying that the certificate has been signed by a Trusted Authority whose certificate is present in the file `ssl` on the WAPT client;
- verifying the signature of the file `manifest.sha256` with the public key;

Signing immediate actions

Context

From the WAPT console, the *Administrators* may launch actions directly on the WAPT clients connected with the WAPT Server using Websockets.

The WAPT console signs these actions with the key and certificate of the *Administrator* before sending them to the WAPT Server using an HTTPS POST request; the request is then forwarded to the targeted WAPT clients.

Possible immediate actions are:

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

Signing process for immediate actions

- the action is defined by its name and the actions attributes. The attributes are *uuid*, *action*, *force*, *notify_server*, and *packages* (for actions implicating a list of packages)
- the attributes *signed_attributes*, *signer*, *signature_date*, *signer_certificate* are added to the structure of the action:
  - *signed_attributes* list of the attributes that are signed;
  - *signer* Subject.COMMON_NAME of certificate holder;
  - *signature_date*: current date and time (UTC) in '%Y-%m-%dT%H:%M:%S' format;
  - *signer_certificate* certificate holder's base64 encoded *X509* certificate;
- the structure is JSON encoded;

- the signature of the JSON file is calculated from the RSA private key of the *signer* using a *sha256* hash algorithm and a *PKCS1 v1.5* padding;
- the signature is base64 encoded and stored on the *signature* attribute inside the JSON file;

Verifying the signature of an immediate action

Context

From the WAPT console, the *Administrators* may launch actions directly on the WAPT clients connected with the WAPT Server using Websockets.

The actions are JSON encoded, signed with the key and certificate of the *Administrator*, and relayed to the targeted WAPT clients by the WAPT Server.

Possible immediate actions are:

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

The action **`get_tasks_status`** does not require SSL authentication.

Verification process

Upon receiving an event on the Websocket connexion of the WAPT client:

- the X509 certificate of the certificate holder is extracted from the JSON file (format PEM);
- the WAPT client tests whether the certificate is to be trusted, i.e. that it is present in `<WAPT>ssl` or that it has been signed by a Trusted Authority (certificate of the Authority present in `<WAPT>ssl`);
- the WAPT client checks whether the certificate can verify the signature that is present in the JSON structure of the action, which consists of:
  - extracting the base64 encoded signature from the *signature* attribute in the JSON file;
  - extracting the signature date formated in '%Y-%m-%dT%H:%M:%S' from the *signature_date* attribute;
  - checking that the signature date is neither too old in the past, nor too late into the future by over 10 minutes;
  - reconstructing a JSON representation of the attributes of the action;
  - checking that the certificate's public key can verify the JSON file with the signature by using a *sha256* hash algorithm and a *PKCS1 v1.5* padding;

Checking the complete download of a package

For each package, a *md5* sum of the package is calculated and stored in the `Packages` index file of the repository.

When installing a package, the WAPT client checks whether a local version of the package is already available in the cache directory `<WAPT>cache`.

If the package file is cached, its *md5* sum is calculated and compared with the *md5* sum in the index file. If they are different, the cached package is deleted.

This *md5* sum is only used to insure that a package has been fully downloaded.

The checking of the signature of the package will fully insure its integrity and its authenticity.

6.3 Comparing features between the WAPT Enterprise and Community versions

6.3.1 Current feature list as of 2023-01-10

Table 1: Comparison of features between the Enterprise and the Community version of WAPT as of 2023-01-10

Feature	Enterprise	Community
Deploy, update and remove software on hosts running	✓	✓
The repository is hosted on infrastructure under your control and not under the control of a large Mega-Corp, Inc. that will prioritize its own interests over yours	✓	✓
Deploy and update **configurations in SYSTEM context**	✓	✓
Deploy and update **configurations in USER context**	✓	✓
Get a **comprehensive inventory** of hardware, software and applied WAPT packages	✓	✓
Benefit from the **differenciated self-service** (authorized users may install authorized software from authorized WAPT package stores)	✓	✗
Benefit from **simplified Windows Updates** that work much better than a standard WSUS (only the required KBs are dowloaded from Microsoft)	✓	✗
Simplify and structure your administrative workload by applying WAPT packages to an OU	✓	✗
Configure and manage easily WAPT **store relays to preserve bandwidth** in multi-site environments	✓	✗
Get access to **ready-to-deploy WAPT packages** for common free-to-use software	✓	✓
Work with **easily verifiable python recipes** for installing, updating and removing software and configuration, recipes may embedd Powershell code or scripts made with other languages (ex: for personalizing a software using a LDAP directory)	✓	✓
Benefit from **hundreds of Helpers** for simplifying your software packaging	✓[1]	✓
Encrypt your sensitive data for transport (software license keys, login, password, server FQDN, API informations for registering software with the vendor, etc)	✓	✗
Automate the auditing of your configurations for an **easy, automated and always up-to-date compliance**	✓	✗
Benefit from the power of SQL integrated with the WAPT console to make **the reports that you need**	✓	✗

6.3.2 Features coming soon

Below is a list of features that we have identified as being really useful to WAPT and WAPT's user community and that we have already started to work on. No timeline is promised, stay tuned, we are only promising you that we are working very hard to achieve these objectives.

Feature	Enterprise	Community
Multi-tenant, multi-client mode with ACL (Access Control Lists) for MSPs (Managed Service Providers) and large multi-departmental or international organisations using an internal PKI (Public Key Infrastructure) based mecanism	✓	✗
Simple to use screensharing for user support, built with the same level of security and privacy as WAPT	✓	✗
History of actions done via WAPT for a complete reporting of a host`s software maintenance lifecycle	✓	✗
Authentication of WAPT Administrators using cryptographic tokens (ex: smartcards)	✓	✗
Access to ready-to-deploy WAPT packages or recipes for licensed business software (common business software for industry, medical, office, public collectivities, cybersecurity, etc)	✓	✗
Access to ready-to-deploy WAPT package extensions for simplifying desktop armoring using Applocker or equivalent	✓	✗
Continued support for Windows XP in WAPT for factory machine tools, Hospital medical equipment, expensive research instruments, etc	✓[4]	✗
Operating system image deployment tool integrated within WAPT	✓	✗
Integration of useful subset of WAPT inventory with popular ITSM (IT Service Management) tools and triggering of actions from the users ITSM console	✓	✗

6.3.3 Summary of operating principles in WAPT

- WAPT is agent based to allow no inbound open port in hosts` firewalls that initiate a secured bi-directional websocket with the server for allowing real-time reporting and actions;
- Can work with Trusted Data Gateways using simple task scheduling;
- Works on the principle of smoothly pulling updates and then applying upgrades at convenient time (works with low / intermittent bandwidth, high latency, high jitter);
- Does not require an AD (works with Windows Home edition too), but will show the host in its Active Directory tree if the host is joined to an AD;

[1] The Enterprise version embeds more SetupHelper functions than the Community version.
[2] In the Community version, the WAPT SuperAdmin password is shared between individuals that manage the WAPT server.
[3] A minimal volume of licences must be subscribed in order to benefit from Tranquil IT's telephone support for the daily operation of the software. Additional paid support is available to help you with your WAPT packaging needs.
[4] Windows XP does not work with Python > 2.7. So a special branch of WAPT will be frozen with the last build of the WAPT agent running with 2.7. This version of the agent will of course be excluded from the target of evaluation in future security certifications.

- Methods for deploying WAPT agent:
  1. using a GPO (Group Policy Object) or an Ansible script;
  2. manually after having downloaded the agent from the WAPT server or using SSH (Secured Shell);
- Methods for registering hosts with the WAPT server:
  1. automatically using the host`s kerberos account;
  2. manually with the WAPT Superadmin login and password;
- Upgrades may be triggered:
  1. upon shutdown of the host, the standard mode;
  2. by an authorized WAPT Administrator in an emergency (ex: critical vulnerabilities running in the wild);
  3. by the user at a time she chooses (ex: 24/7 nursing cart unused during lunch break with a simple click);
  4. via a scheduled task running at a predetermined time (best for servers);
- Security is insured with:
  1. signing of WAPT packages using asymetric cryptography;
  2. authentication of hosts against the WAPT server using symetric cryptography on registering;
  3. confidentiality of the WAPT server using WAPT deployed client certificates;

6.4 WAPT Community and WAPT Enterprise

6.4.1 Main functional benefits of the Enterprise version of WAPT

WAPT is designed to help IT system administrators manage the lifecycle of their installed base of Windows applications, drivers, OSes and configurations in system and user context.

With WAPT Enterprise, you benefit automatically from the base functions included in WAPT Community to help you deploy, upgrade and remove software and configurations on your Windows devices, from a central console.

WAPT is an *opencore* model and the **Enterprise** version is based on the **Community** version with the following features designed to answer the needs of larger Organisations:

- **Active Directory authentication** of WAPT package developers, package deployers, self-service users and for the initial registering of the WAPT agents with the WAPT Server. In addition, the display of WAPT equipped devices in the WAPT console follow the same structure as the hierarchical structure of the Organization's Active Directory OU;

- **role separation between package developers and package deployers**. This way, central IT teams may build the software packages because they know the Organization's security guidelines, and local IT teams may deploy the WAPT packages because they know the needs of their user base. Such a separation is implemented using differentiated sets of keys (i.e. **Code Signing** SSL certificates for package developers and **Simple** SSL certificates for package deployers);
- **differentiated self service** WAPT Enterprise allows you to apply lists of allowed packages to groups in Active Directory. Allowed users are free to install qualified packages from their list of approved packages without having to submit a ticket to their IT teams.
- **WAPT WUA** WAPT allows to manage the Windows Updates on your endpoints.
- **advanced reporting for corporate teams**. This reporting completes the operational reporting already available in the WAPT console; reports help WAPT operators demonstrate their efficacy with WAPT for insuring a greater level of security and conformity for their networks, systems, software and applications.
- **dynamic repository configuration**. Starting with WAPT 1.8, repository replication can be enabled using a WAPT agent installed on an existing machine, a dedicated appliance or Virtual Machine.

  The replication role is deployed through a WAPT package that enables the **`Nginx web server`** and configures scheduling, packages types, packages sync, and much more.

  This feature allows WAPT agents to find dynamically their closest available WAPT repository from a list of rules stored on the WAPT server.

The Enterprise version of WAPT is particularly advisable for Organizations:

- that manage large installed bases of devices (generally above 300 units);
- that are spread geographically with many subsidiaries or production sites;
- that require a strong traceability of actions performed on the installed base of devices for reasons of audit or security;

6.4.2 Description of services available with the WAPT Enterprise contract

Access to future improvements in WAPT Enterprise

By subscribing to a WAPT Enterprise contract and by maintaining your subscription valid, you benefit from the future improvements brought into the core of WAPT and you benefit automatically from all future improvements to the WAPT Enterprise version.

A lapsing of your subscription will automatically switch your WAPT instance back to its corresponding Community version; advanced functions only available in the Enterprise version will no longer be accessible.

Direct telephone support for your daily usage of WAPT

When your subscription reaches above a certain volume, Tranquil IT, the creator of WAPT, allows you a privileged access to its core team of WAPT experts and developers.

We give you access to a dedicated telephone hot-line with a direct answer to satisfy your needs for support.

We are committed to providing you with reliable and pertinent answers on the subscribed perimeter, quickly.

By subscribing or renewing your WAPT Enterprise contract, you will receive a notification indicating the practicalities to access our support.

Attention: The support concerns only the use in your Organization of the WAPT Enterprise software; additional support for adapting, personalizing, debugging or creating WAPT packages may be obtained with prepaid support tickets.

Up to two individuals in your Organisation may communicate with our direct support.

Price and preferential access to WAPT training

You may choose to train your IT team on any particularity of WAPT.

WAPT Enterprise subscribers benefit from a privileged access to Tranquil IT's training advisers and a 50% discount on standard training prices.

6.4.3 Why does Tranquil IT display a 300 device limit for WAPT Community on her commercial documentation ?

Tranquil IT displays a 300 device limit on her commercial documentation for the WAPT Community version.

Note: There is **no technical restriction to using the Community version**, it is available as free software.

Tranquil IT's philosophy regarding free software is to give back a little of what we took, so it is important to us to keep our original promise to give systems administrators a mean to deploy, maintain up-to-date and remove software and configurations on their installed base of Windows devices.

What you will find in the WAPT Enterprise version are features that are designed to help administer large installed bases and we meet people in that situation daily.

As a general rule, we have observed that advanced management needs start to be felt when the number of devices goes over 300-400 units.

6.5 Installing the WAPT Server

The WAPT Server can be installed on Debian Linux, CentOS or Microsoft Windows **64 bits only**.

6.5.1 Installation tips and requirements

You have to take into consideration a few security points in order to get all the benefits of WAPT:

- we advise you to install WAPT Server on a Linux server (Debian or CentOS) following the security recommendations of French *ANSSI* or the recommendations of your state cyberdefense agency.
- the WAPT Server must be installed on a **dedicated machine** (physical or virtual);

Attention: In all steps of the documentation, you will not use any accent or special character for:

- user logins;
- path to the private key and the certificate bundle;
- the CN (Common Name);
- the installation path for WAPT;
- group names;

- the name of hosts or the the name of the server;
- the path to the folder `C:\waptdev`;

6.5.2 Hardware recommendations

The WAPT Server can be installed either on a virtual server or a physical server.

RAM and CPU recommendations are:

Size of the network	CPU	RAM
From 0 to 200 desktops	1 CPU	1024MB
From 200 desktops onward	4 CPU	4096Mo

A minimum of 10GB of free space is necessary for the system, the database and log files. **For better performance, Tranquil IT recommends the database to be stored on fast storage, such as SSD drives or PCIe-based solid-state drives**.

The overall disk requirement will depend on the number and size of your WAPT packages (software) that you will store on your main repository; 30GB is a good start. It is not strictly required to store WAPT packages on fast drives.

6.5.3 Configuring the DNS

Configuring the Organization's DNS for WAPT

Note: DNS configuration is not strictly required, but it is very strongly recommended.

In order to make make your WAPT setup easier to manage, it is strongly recommended to configure the *DNS* server to include *A field* or *CNAME field* as below:

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

Replace *mydomain.lan* with your network's *DNS* suffix.

These DNS fields will be used by WAPT agents to locate the WAPT Server and their WAPT repositories closest to them.

Configuring DNS entries in Microsoft RSAT.

- The *A* field must point to the WAPT Server IP address;

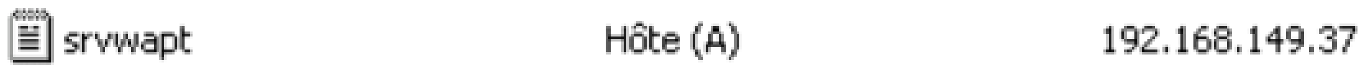

srvwapt Hôte (A) 192.168.149.37

Fig. 14: Configuration of the A field in Windows RSAT

You can now install the WAPT Server on your favorite operating system:

- *install the WAPT Server on GNU / Linux Debian*;

- *install the WAPT Server on CentOS / RedHat*;
- *install the WAPT Server on Windows*;

6.5.4 Installing WAPT Server on Debian Linux

Setting up the GNU/Linux Debian server

In order to install a fresh Debian Linux 10 *Buster* (physical or virtual) without graphical interface, please refer to the Debian GNU/Linux Installation Guide.

Configuring network parameters

The different parameters presented below are not specific to WAPT; you may adapt them as required for your environment.

Modify the following files in order to get a proper naming (*FQDN*) and network addressing strategy.

In the following example:

- the *FQDN* name is *srvwapt.mydomain.lan*;
- the short-name of the WAPT Server is *srvwapt*;
- the *DNS* suffix is *mydomain.lan*;
- the IP address is *10.0.0.10/24*;

Configuring the name of WAPT Server

Hint: The short name of the WAPT Server must not be longer than 15 characters (the limit is due to sAMAccountName restriction in Active Directory).

The name of the WAPT Server must be a FQDN (Fully Qualified Domain Name), that is to say it has both the server name and the DNS suffix.

- modify the `/etc/hostname` file and write the *FQDN* of the server;

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

- configure the `/etc/hosts` file, be sure to put both the *FQDN* and the short name of the server;

```
# /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
```

Hint:

- on the line defining the DNS server IP address, be sure to have the IP of the server (not 127.0.0.1), then the *FQDN*, then the short name;

- do not change the line with `localhost`;

Configuring the IP address of the WAPT Server

- configure the IP address of the WAPT Server in the `/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
```

- apply the network configuration by rebooting the machine with a `reboot`;
- if it has not already been done, create the DNS entry for the WAPT Server in the *Organization*'s Active Directory;
- *Configuring the Organization's DNS for WAPT*
- after reboot, configure the system language in English in order to have non-localized logs for easier searching of common errors;

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

- check that the machine clock is on time (with NTP installed);

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

Hint: If the NTP package is not installed.

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

- update and upgrade your Debian;

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

- install systemd;

  ```
  apt install systemd
  ```

- install certificates;

  ```
  apt install ca-certificates
  ```

- restart server;

```
reboot
```

You may no go on to the next step and *install WAPT on your Debian*.

Installing the WAPT Server on Debian Linux

Attention: The upgrade procedure is different from installation. For upgrade, please refer to *Upgrading the WAPT Server*.

Installing the WAPT Server requires a few steps:

- configuring the repositories;
- installing additional Linux packages;
- installing and provisioning the PostgreSQL database;
- post-configuring the WAPT Server;

Note: The WAPT Server packages and repository are signed by Tranquil IT and it is necessary to get the gpg public key below in order to avoid warning messages during installation.

Configuring DEB repositories and installing WAPT and PostgreSQL packages

The configuration of repositories for **WAPT Enterprise** and **WAPT Community** Edition differs. **Make sure to choose the right one!**

During installation, you may be asked for the Kerberos realm. Just press `Enter` to skip this step.

Important: Follow this procedure for getting the right packages for the WAPT Enterprise Edition. For WAPT Community Edition please refer to the next block.

To access WAPT Enterprise ressources, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **deb** parameter to access WAPT Enterprise repository.

```
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: Follow this procedure for getting the right packages for the WAPT Community Edition. For WAPT Enterprise Edition please refer to the previous block.

```
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
```

(continues on next page)

(continued from previous page)

```

```

Installing the WAPT Server packages

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

Post-configuring

Attention: For post-configuration to work properly, you must first have properly configured the *hostname* of the WAPT server. To check *hostname* configuration use the command `echo $(hostname)` which must return the *hostname* that will be used by WAPT agents on client computers.

Hint: This post-configuration script must be run as **root**.

- run the script:

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

- click on *Yes* to run the postconf script:

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

         < yes >          < no >
  ```

- enter the password for the *SuperAdmin* account of the WAPT Server (minimum 10 caracters);

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

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

  < OK >          < Cancel >
  ```

- confirm the password;

  ```
  Please enter the server password again:

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

  < OK >          < Cancel >
  ```

- choose the authentication mode for the initial registering of the WAPT agents;
  - choice #1 allows to register computers without authentication (same method as WAPT 1.3). The WAPT server registers all computers that ask;
  - choice #2 activates the initial registration based on Kerberos. (you can activate it later);

  - choice #3 does not activate the Kerberos authentication mechanism for the initial registering of machines equipped with WAPT. The WAPT server will require a login and password for each machine registering with it;

```
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 >
```

- select *OK* to start WAPT Server;

```
Press OK to start waptserver

       < OK >
```

- select *Yes* to configure Nginx;

```
Do you want to configure nginx?

   < Yes >          < No >
```

- enter the *FQDN* of the WAPT Server;

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

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

      < OK >             < Cancel >
```

- select *OK* and a self-signed certificate will be generated, this step may take a long time …

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

Nginx is now configured, select *OK* to restart **Nginx**:

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

       < OK >
```

The post-configuration is now finished.

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

                 < OK >
```

Listing of post-configuration script options:

Flag	Description
`--force-https`	Configures **Nginx** so that *port 80 is permanently redirected to 443*

The WAPT Server is now ready.

You may go to the documentation on *installing the WAPT console*!!

6.5.5 Installing WAPT Server on CentOS7

Configuring the CentOS/ RedHat server

In order to install a fresh CentOS7 machine (virtual or physical) without graphical interface (choose **minimal** installation), please refer to official CentOS documentation. This documentation is also valid for Redhat7 initial installation.

Configuring network parameters

The different parameters presented below are not specific to WAPT; you may adapt them as required for your environment.

Modify the following files in order to get a proper naming (*FQDN*) and network addressing strategy.

In the following example:

- the *FQDN* name is *srvwapt.mydomain.lan*;
- the short-name of the WAPT Server is *srvwapt*;
- the *DNS* suffix is *mydomain.lan*;
- the IP address is *10.0.0.10/24*;

Configuring the name of WAPT Server

Hint: The short name of the WAPT Server must not be longer than 15 characters (the limit is due to sAMAccountName restriction in Active Directory).

The name of the WAPT Server must be a FQDN, that is to say it has both the server name and the DNS suffix.

- modify the `/etc/hostname` file and write the *FQDN* of the server;

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

- configure the `/etc/hosts` file, be sure to put both the *FQDN* and the short name of the server;

```
# /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
```

Hint:

- on the line defining the DNS server IP address, be sure to have the IP of the server (not 127.0.0.1), then the *FQDN*, then the short name;
- do not change the line with `localhost`;

Configuring the IP address of the WAPT Server

- modify the `/etc/sysconfig/network-scripts/ifcfg-eth0` file and define a static IP address. The name of the file can be different, like `ifcfg-ens0` for example;

```
# /etc/sysconfig/network-scripts/ifcfg-eth0 du serveur wapt
TYPE="Ethernet"
BOOTPROTO="static"
NAME="eth0"
ONBOOT="yes"
IPADDR=10.0.0.10
NETMASK=255.255.255.0
GATEWAY=10.0.0.254
DNS1=10.0.0.1
DNS2=10.0.0.2
```

- apply the network configuration by rebooting the machine with a `reboot`;

  ```
  reboot
  ```

- if it has not already been done, *create the DNS entries for the WAPT Server* in the *Organization* Active Directory;
- after reboot, configure the system language in English in order to have non-localized logs for easier searching of common errors;

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

- check that the machine clock is on time (with NTP installed), and that SELinux and the firewall are enabled;

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

Hint: If the NTP package is not installed.

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

- update CentOS7 and set up the EPEL (Extra Packages for Enterprise Linux) repository;

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

You may now go on to the next step and *install WAPT on your CentOS/ RedHat*.

Installing the WAPT Server on CentOS / RedHat

Attention: The upgrade procedure is different from installation. For upgrade, please refer to *Upgrading the WAPT Server*.

Installing the WAPT Server runs a few steps:

- configuring the repositories;
- installing additional Linux packages;
- installing and provisioning the PostgreSQL database;
- post-configuring the WAPT Server;

Configuring RPM repositories and installing WAPT and PostgreSQL packages

The configuration of repositories for **WAPT Enterprise** and **WAPT Community** Edition differs. **Make sure to choose the right one!**

During installation, you may be asked for the Kerberos realm. Just press `Enter` to skip this step.

Important: Follow this procedure for getting the right packages for the WAPT Enterprise Edition. For WAPT Community Edition please refer to the next block.

To access WAPT Enterprise ressources, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **baseurl** parameter to access WAPT Enterprise repository.

```
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: Follow this procedure for getting the right packages for the WAPT Community Edition. For WAPT Enterprise Edition please refer to the previous block.

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

(continues on next page)

(continued from previous page)

```
gpgcheck=1
EOF
```

Installing the WAPT Server packages

```
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-configuring

- initialize the PostgreSQL database and activate the 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: For post-configuration to work properly, you must first have properly configured the *hostname* of the WAPT server. To check, use the command `echo $(hostname)` which must return the DNS address that will be used by WAPT agents on client computers.

Hint: This post-configuration script must be run as **root**.

- run the script:

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

- click on *Yes* to run the postconf script:

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

       < yes >          < no >
```

- choose a password for the *SuperAdmin* account of the WAPT server (minimum length is 10 characters);

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

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

        < OK >          < Cancel >
```

- confirm the password;

```
Please enter the server password again:

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

        < OK >            < Cancel >
```

- choose the authentication mode for the initial registering of the WAPT agents;
  - choice #1 allows to register computers without authentication (same method as WAPT 1.3). The WAPT server registers all computers that ask;
  - choice #2 activates the initial registration based on Kerberos. (you can activate it later);
  - choice #3 does not activate the kerberos authentication mechanism for the initial registering of machines equipped with WAPT. The WAPT server will require a login and password for each machine registering with it;

```
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 >
```

- select *OK* to start WAPT Server;

```
Press OK to start waptserver

       < OK >
```

- select *Yes* to configure Nginx;

```
Do you want to configure nginx?

   < Yes >        < No >
```

- fill in the *FQDN* of the WAPT server;

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

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

      < OK >            < Cancel >
```

- select *OK* and a self-signed certificate will be generated, this step may take a long time . . .

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

Nginx is now configured, select *OK* to restart **`Nginx`**:

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

      < OK >
```

The post-configuration is now finished.

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

                 < OK >
```

Listing of post-configuration script options:

Flag	Description
`--force-https`	Configures **`Nginx`** so that *port 80 is permanently redirected to 443*

The WAPT Server is now ready.

You may go to the documentation on *installing the WAPT console*!!

6.5.6 Installing WAPT Server on a Windows host

Installing WAPT Server on Windows

Attention:

- The WAPT Server can not be installed on a computer with services already listening on ports 80 and 443 (example WSUS with IIS).
- ports 80, 443 and 8080 are used by the WAPT Server and must be available;
- If ports 80 and 443 are already occupied by another web service, you should take a look at the documentation for *changing the default ports on Windows*.
- port 8088 is used on client machines by the WAPT agent and must be available;
- The WAPT server **will not run** on a x86 version of Windows;
- The installation of the WAPT server must be done using a **Local Administrator** account on the host;

Attention: From WAPT Server 1.5 onward, **`Nginx`** is the **ONLY** supported web server. **Apache or IIS (with or without WSUS) are no longer supported in WAPT**.

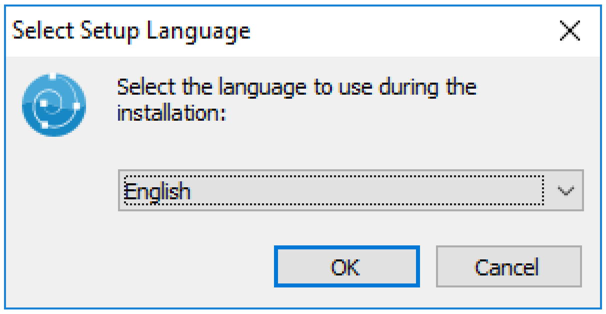
In case of problems when installing WAPT, visit the *Frequently Asked Questions*.

Note:

- installing WAPT on a Linux server is the recommended method;

- the WAPT Server may be installed on **64bit only** Windows 7, 8.1, 10 and Windows Server 2008/R2, 2012/R2, 2016 and 2019;

- Community: download and execute waptserversetup.exe;
- Enterprise: download and execute waptserversetup.exe;
- choose the installation language;

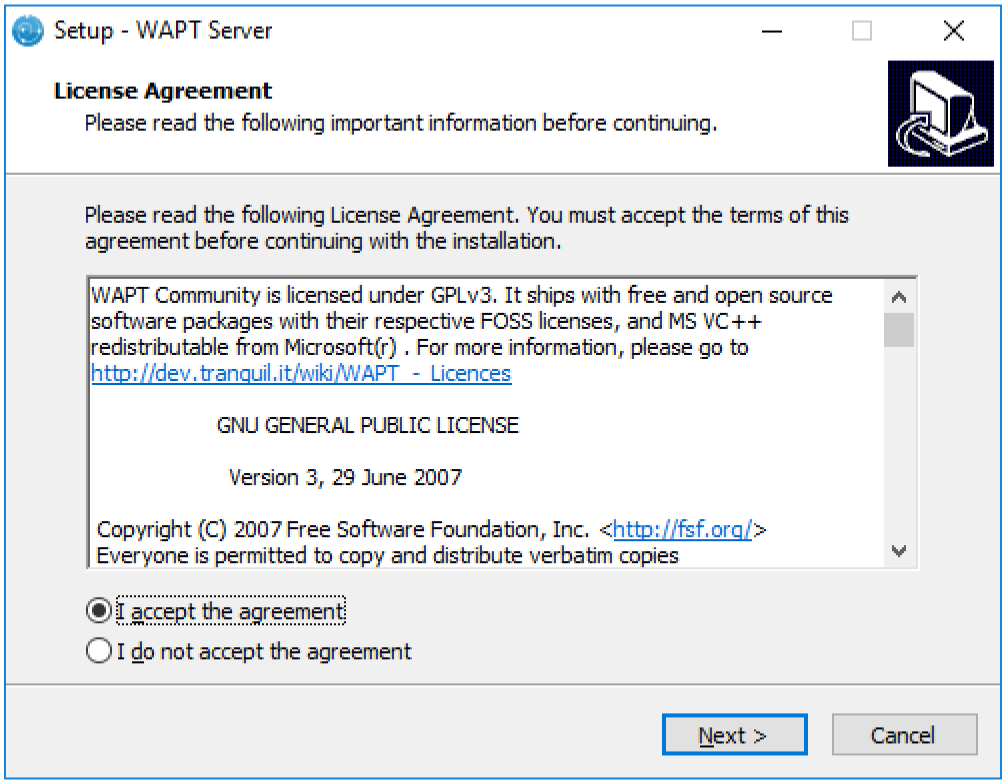

Select Setup Language

Select the language to use during the installation:

English

Fig. 15: Choose the language for WAPT

- accept the GNU Public License and click on *Next* to go on to the next step;


Setup - WAPT Server

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

WAPT Community is licensed under GPLv3. It ships with free and open source software packages with their respective FOSS licenses, and MS VC++ redistributable from Microsoft(r) . For more information, please go to http://dev.tranquil.it/wiki/WAPT_-_Licences

GNU GENERAL PUBLIC LICENSE

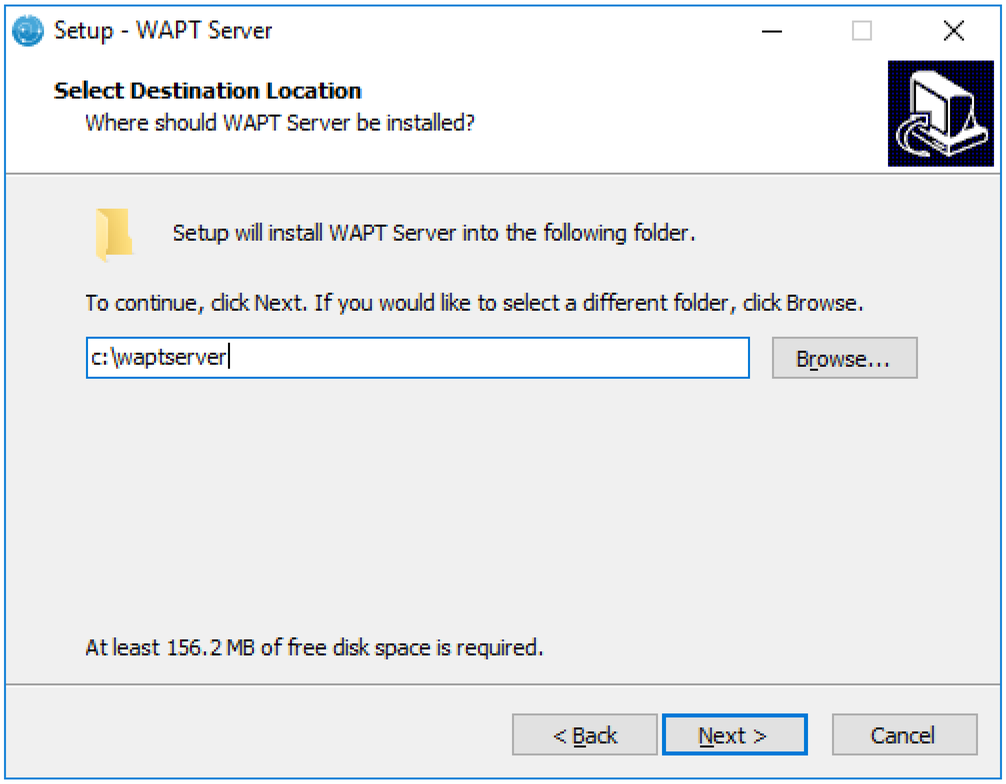
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies

[✓ I accept the agreement]
I do not accept the agreement

Fig. 16: Accept the WAPT license terms

- choose the installation directory (leave the default) and click on *Next* to go on to the next step;
- choose additional task (leave as is for the first installation);



Setup - WAPT Server

Select Destination Location

Where should WAPT Server be installed?

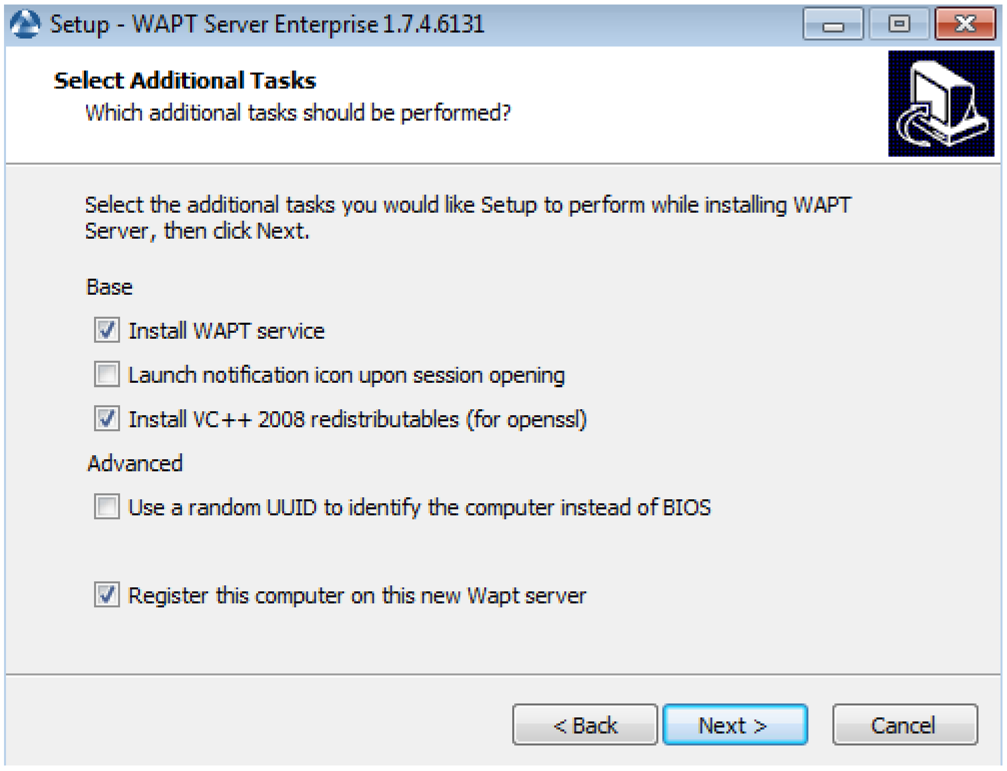
Setup will install WAPT Server into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

c:\waptserver

At least 156.2 MB of free disk space is required.

Fig. 17: Choose the WAPT destination folder



Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing WAPT Server, then click Next.

Base

- [x] Install WAPT service
- [ ] Launch notification icon upon session opening
- [x] Install VC++ 2008 redistributables (for openssl)

Advanced

- [ ] Use a random UUID to identify the computer instead of BIOS

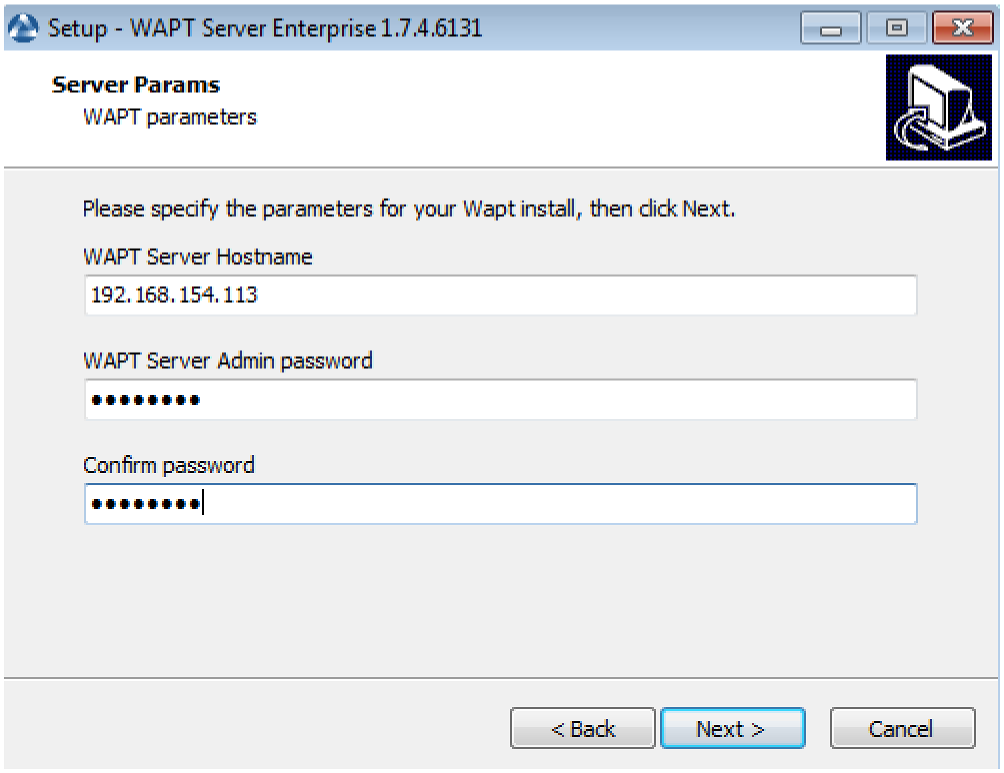
- [x] Register this computer on this new Wapt server

Fig. 18: Choose additional task

- choose the password for the WAPT server;
- create a key if this is your first installation, otherwise select the existing key;
- choose the password for the private key;
- build the WAPT agent;
- choose the prefix used for naming your WAPT packages;
- click on the *Install* to launch the installation, wait for the installation to complete, then click on *Finish* (leave default options);
- click on *Finished* to close the installer;

The WAPT Server on your Windows is ready.

You may now go to the documentation on *installing the waptagent*!!



Server Params
WAPT parameters

Please specify the parameters for your Wapt install, then click Next.

WAPT Server Hostname
192.168.154.113

WAPT Server Admin password
••••••••

Confirm password
••••••••

Fig. 19: Choose Password

Changing the listening port of the WAPT Server

Note: In some cases, it is not possible to install the WAPT Server on a Windows machine because a service already occupies ports 80 and 443.

It is the case if for example, a IIS web service is active on the host (example: anti-virus server, WSUS, web server . . .).

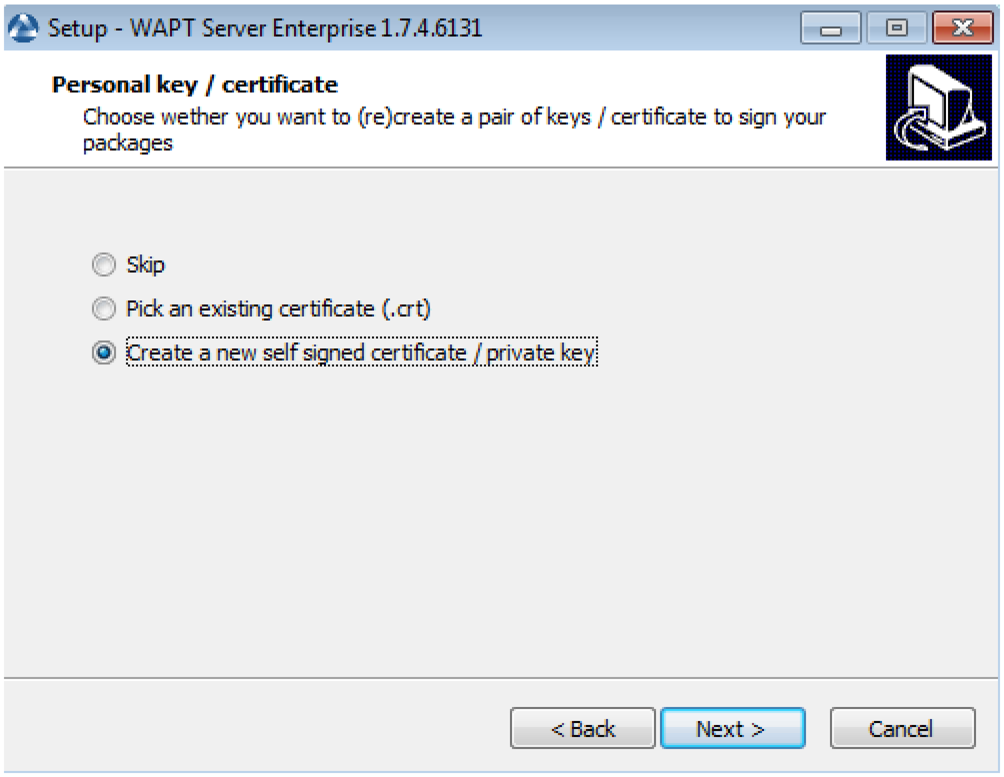
In that case, we will change the listening port on the **Nginx** web server integrated to the WAPT Server.

Installing the WAPT Server

- The installation of WAPT still needs ports 80 and 443 be available when installing the WAPT Server, so the first step consists of stopping the service that listens on ports 80 and/ or 443 (IIS/ Anti-virus).
- launch now the installation of the WAPT Server and follow the post-configuration procedure, but do not launch the WAPT console. If you need guidance, you may follow the documentation to *install the WAPT Server on Windows*.
- now stop the **Nginx** service and the WAPT service:

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

- finally, restart the service that listens on ports 80 and/ or 443 (IIS/ Anti-virus / Web server . . .);

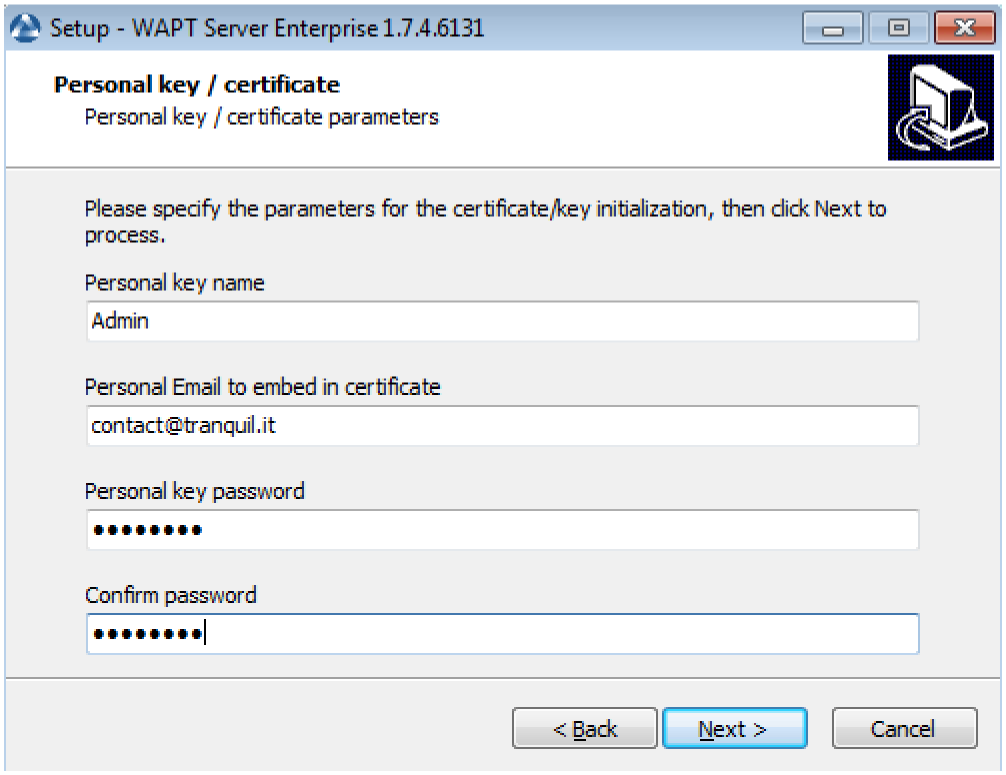

Setup - WAPT Server Enterprise 1.7.4.6131

Personal key / certificate

Choose wether you want to (re)create a pair of keys / certificate to sign your packages

- Skip
- Pick an existing certificate (.crt)
- [✓ Create a new self signed certificate / private key]

Fig. 20: Create the signature key


Setup - WAPT Server Enterprise 1.7.4.6131

Personal key / certificate

Personal key / certificate parameters

Please specify the parameters for the certificate/key initialization, then click Next to process.

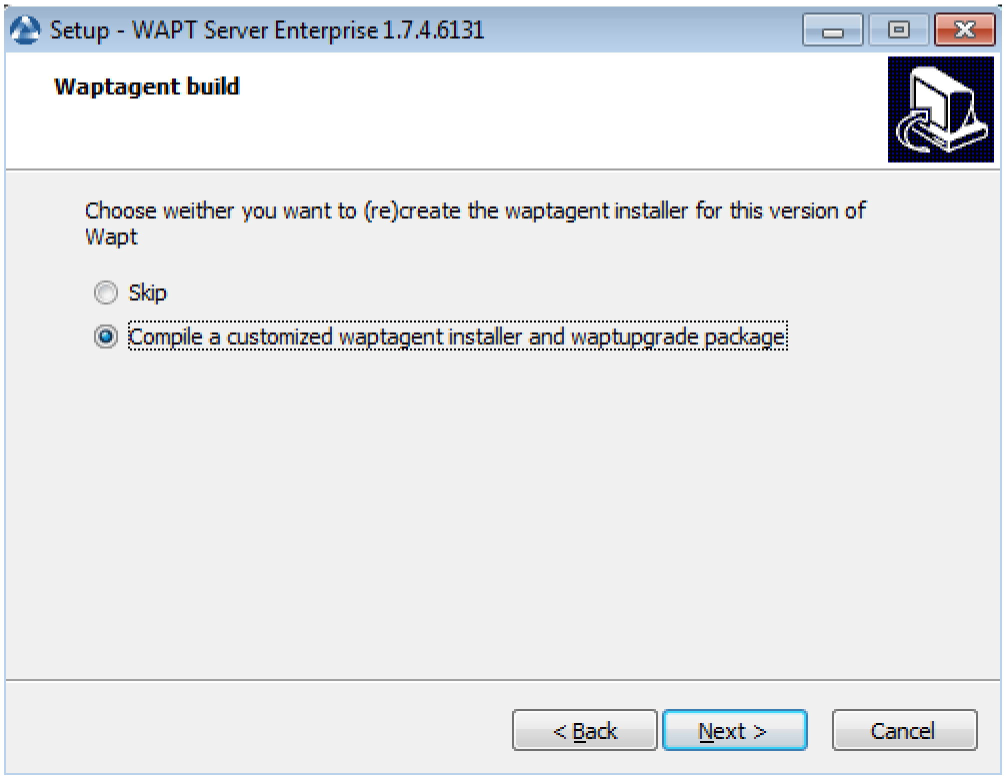
Personal key name: Admin

Personal Email to embed in certificate: contact@tranquil.it

Personal key password: ●●●●●●●●

Confirm password: ●●●●●●●●

Fig. 21: Choose the password of the private key

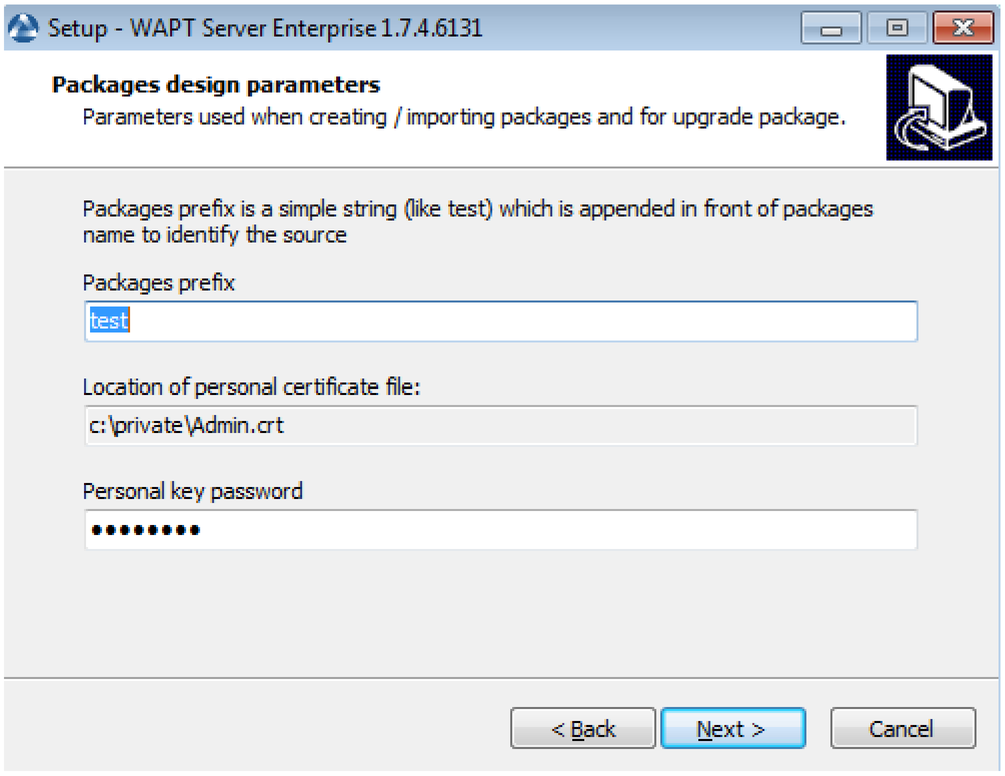

Setup - WAPT Server Enterprise 1.7.4.6131

Waptagent build

Choose weither you want to (re)create the waptagent installer for this version of Wapt

- Skip
- [✓ Compile a customized waptagent installer and waptupgrade package]

Fig. 22: Build the WAPT agent



Setup - WAPT Server Enterprise 1.7.4.6131

Packages design parameters

Parameters used when creating / importing packages and for upgrade package.

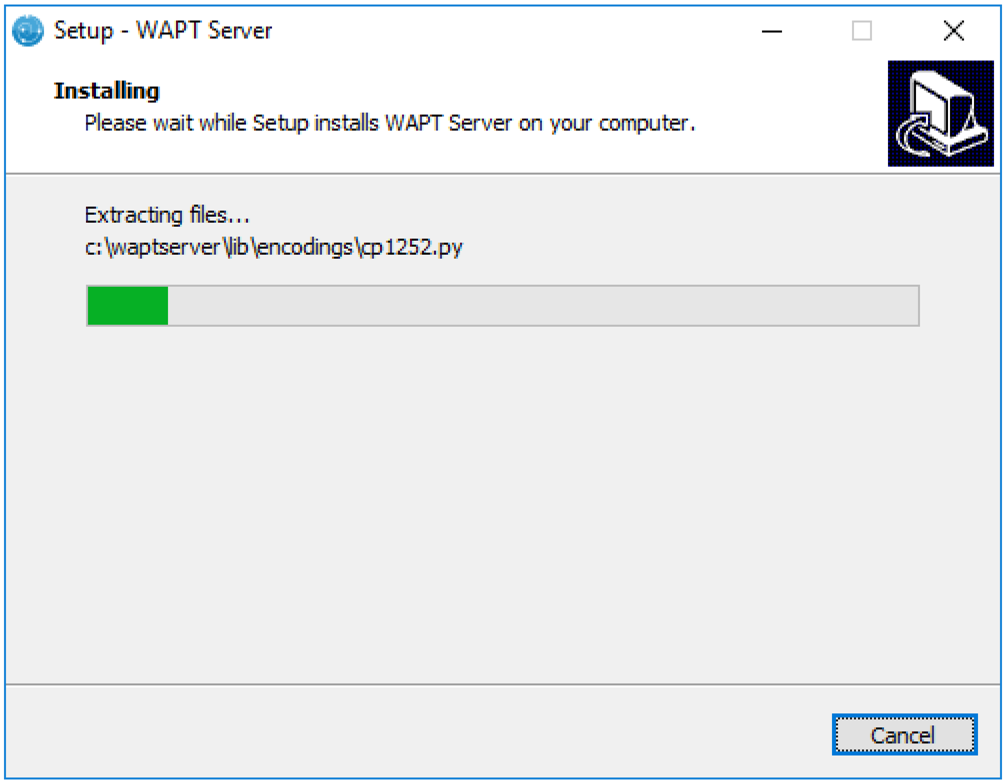
Packages prefix is a simple string (like test) which is appended in front of packages name to identify the source

Packages prefix: test

Location of personal certificate file: c:\private\Admin.crt

Personal key password: •••••••••

Fig. 23: Choose the prefix used for naming your WAPT packages

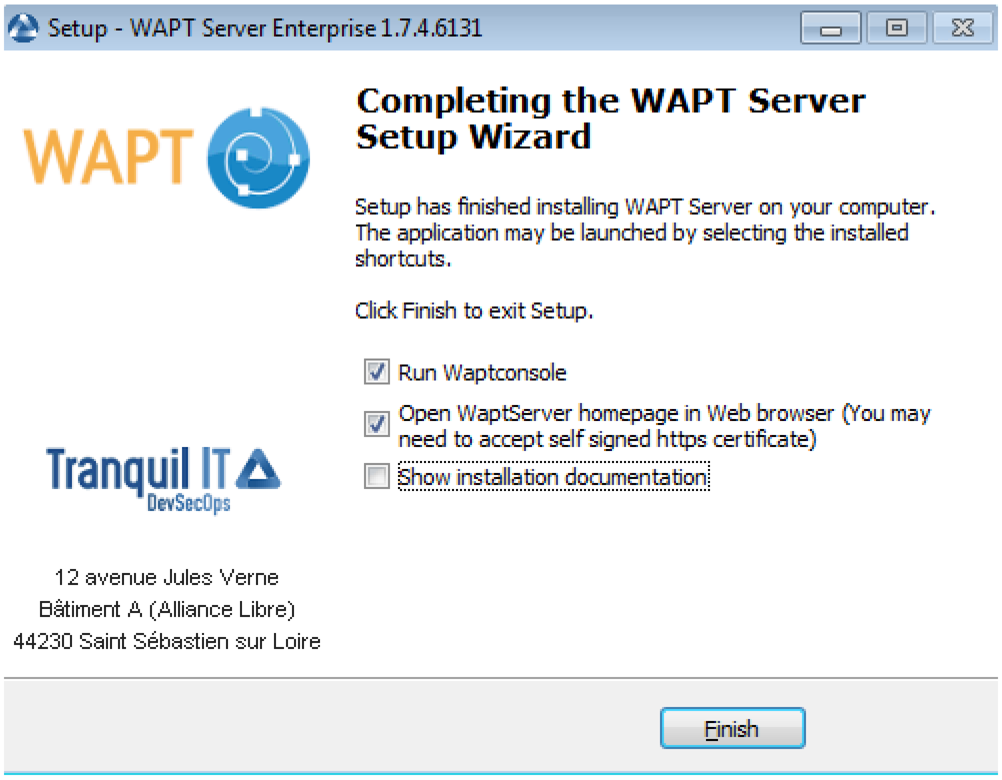

Setup - WAPT Server

Installing

Please wait while Setup installs WAPT Server on your computer.

Extracting files...
c:\waptserver\lib\encodings\cp1252.py

Fig. 24: Progress of installation of the WAPT Server


Setup - WAPT Server Enterprise 1.7.4.6131

Completing the WAPT Server Setup Wizard

Setup has finished installing WAPT Server on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

- [✓] Run Waptconsole
- [✓] Open WaptServer homepage in Web browser (You may need to accept self signed https certificate)
- [ ] Show installation documentation

12 avenue Jules Verne
Bâtiment A (Alliance Libre)
44230 Saint Sébastien sur Loire

Fig. 25: Installation has finished

Configuring the new listening ports in the Nginx

- open the file `C:\Program Files (x86)\wapt\waptserver\nginx\conf\nginx.conf`
- replace the lines:

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

  with:

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

- restart **Nginx** with `net start WAPTNginx`;
- open `C:\Program Files (x86)\wapt\wapt-get.ini`;
- add the port to the specified URL, example:

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

- restart the WAPT service with `net start waptservice`;

Making changes to firewall rules on the Windows devices

You must now make changes to the two waptserver rules that were created during initial installation: waptserver *waptserver 80* and *waptserver 443*.

- change the listening ports:

Go on to the next step to *launch the WAPT console*.

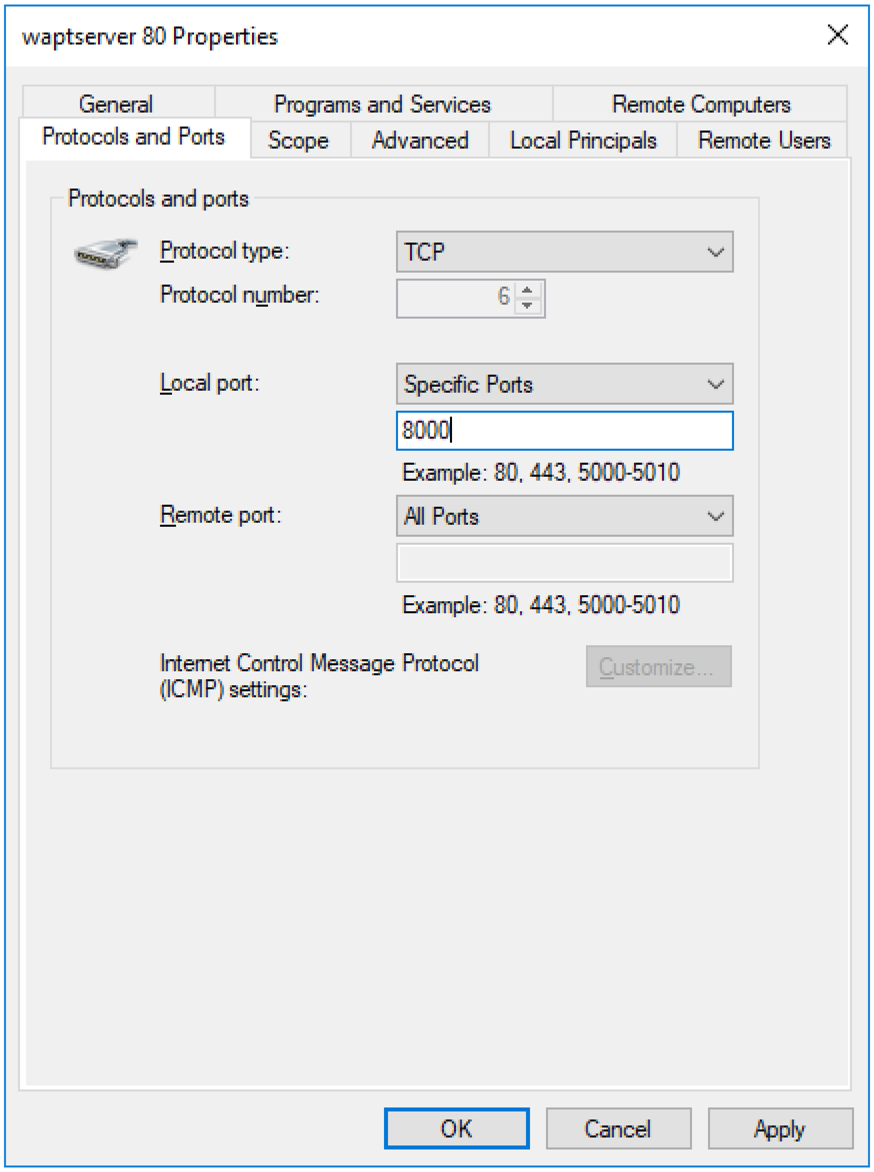
Hint: If you had already launched the WAPT console, do not forget to change the port values in the WAPT console configuration file by clicking the *Wrench* on the console login screen.

6.5.7 Installing WAPT Server with Ansible

Installing WAPT Server with Ansible

To avoid mistakes and automate your WAPT Server deployment, we provide Ansible roles for WAPT Server installation.

You can explore the role source code by visiting Tranquil IT repository on Github.



Protocols and ports

Protocol type: TCP

Protocol number: 6

Local port: Specific Ports

8000

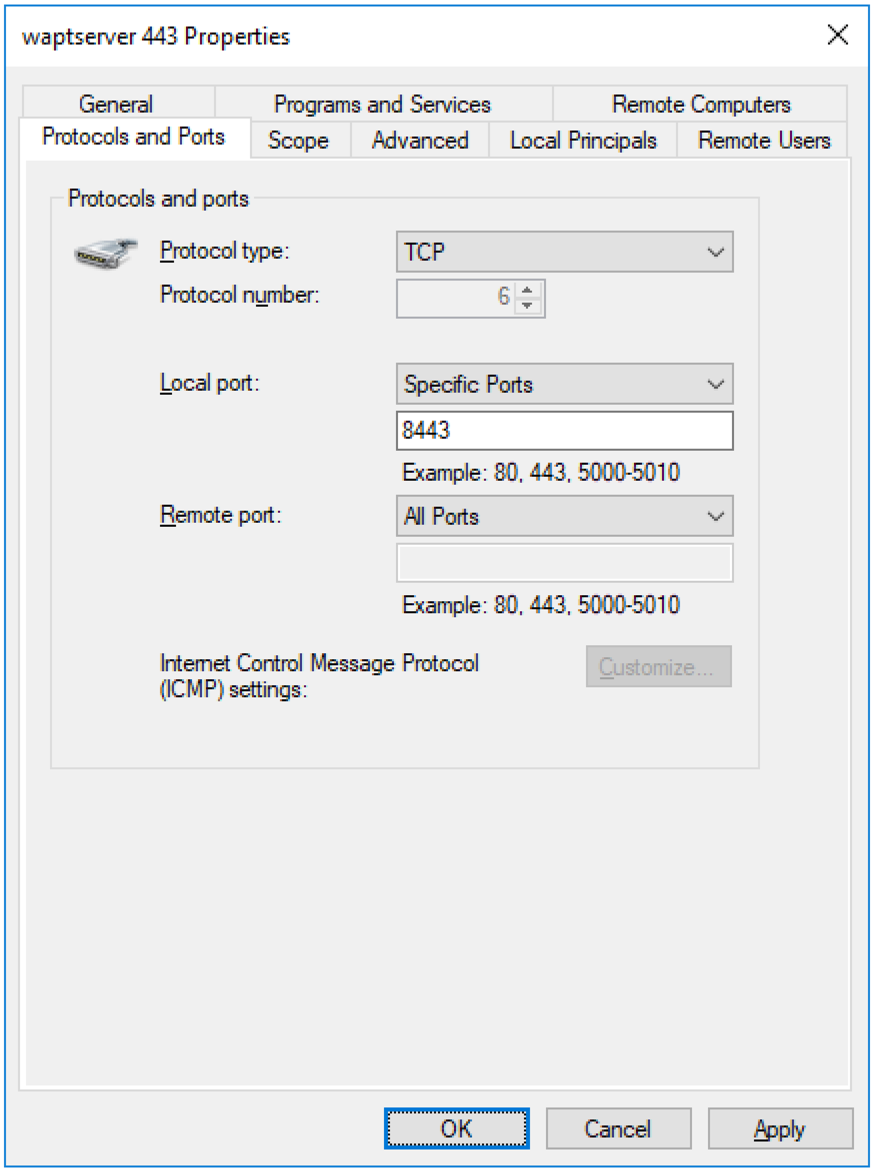
Example: 80, 443, 5000-5010

Remote port: All Ports

Example: 80, 443, 5000-5010

Internet Control Message Protocol (ICMP) settings:

Fig. 26: Changing from port 80 to 8000



Protocols and ports

Protocol type: TCP

Protocol number: 6

Local port: Specific Ports

8443

Example: 80, 443, 5000-5010

Remote port: All Ports

Example: 80, 443, 5000-5010

Internet Control Message Protocol (ICMP) settings:

Fig. 27: Changing from port 443 to 8443

Requirements

- Debian Linux or CentOS hosts;
- a sudoers user on these hosts;
- Ansible 2.8;

Installing the Ansible role

- install `tranquilit.waptserver` Ansible role;

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

- to install the role elsewhere, use the *-p* subcommand like this;

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

Using the Ansible role

- ensure you have a working ssh key deployed on your hosts, if not you can generate and copy one like below;

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

- edit Ansible hosts inventory (`./hosts`) and add the Linux hosts;

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

- create a playbook with the following content in `./playbooks/wapt.yml`;

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

- run your playbook with the following command;

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

Congratulations, you have installed your WAPT server on your Linux server!

Role variables

Available variables are listed below, along with default values (see `defaults/main.yml`):

- version of WAPT that will be installed from WAPT Deb/RPM repository;

  ```
  wapt_version: "1.8"
  ```

- version of PostgreSQL that will be installed from WAPT Deb/RPM repository;

  ```
  pgsql_version: "9.6"
  ```

- version of CentOS used for RPM repository address;

  ```
  centos_version: "centos7"
  ```

- `launch_postconf` defaults to True, it launches WAPT Server postconfiguration script silently;

  ```
  launch_postconf: True
  ```

Example Ansible playbook

Here is an example of an Ansible playbook.

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

6.6 Configuring WAPT

The WAPT Server having been successfully installed, the next steps are as follow:

6.6.1 Installing the WAPT management console

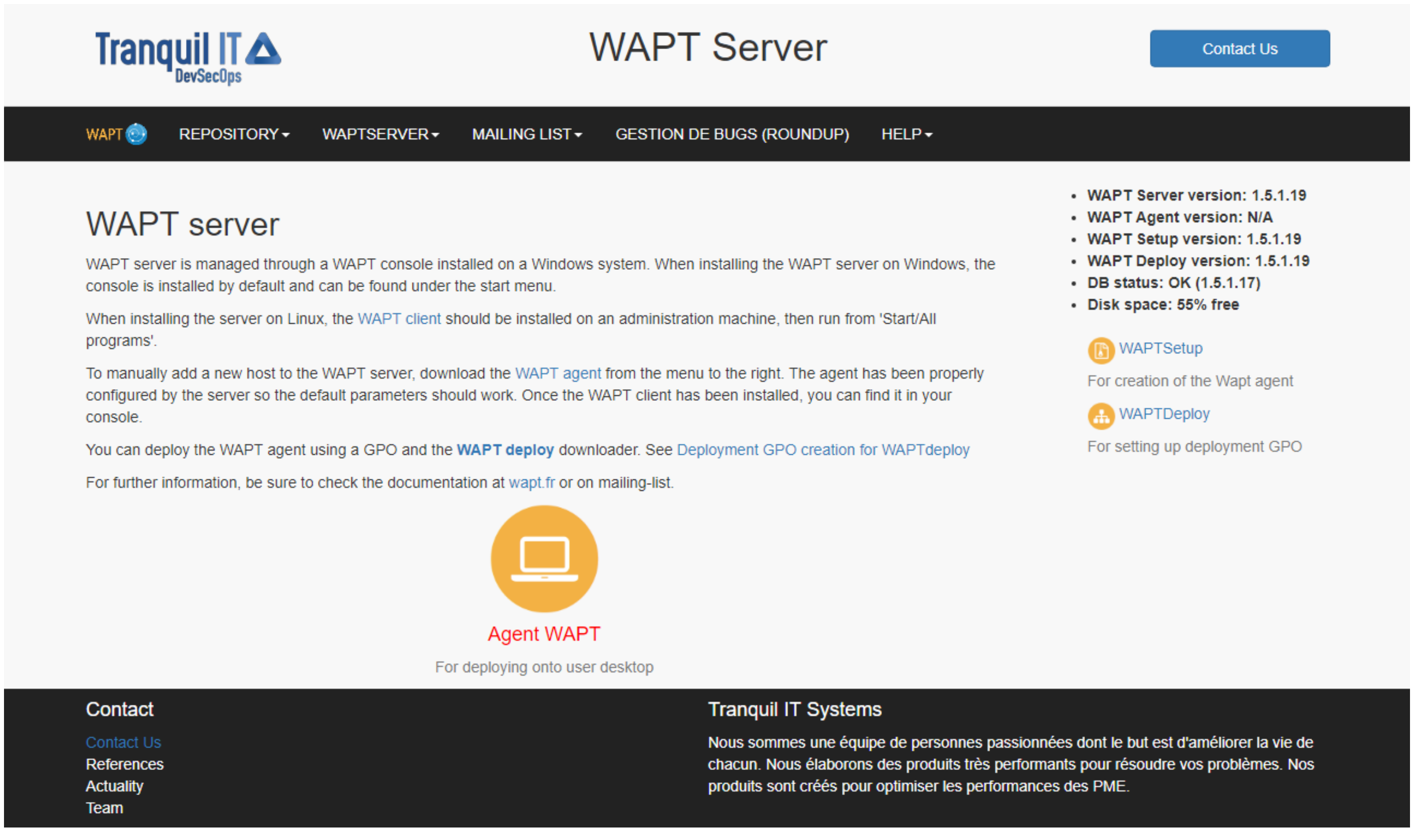
Note:

- managing WAPT is done mainly via the WAPT console installed on the *Administrator*'s workstation;
- the Administrator's computer must be joined to the *Organization*'s Active Directory;
- the host name of the Administrator's workstation must not be longer than 15 characters. This is a limit of sAMAccountName attribute in Active Directory;
- the Administrator's computer will become critical for WAPT administration and WAPT package testing;
- if DNS records are properly configured, you should be able to access the WAPT web interface by visiting: https://srvwapt.mydomain.lan;

Downloading and launching the installation of the WAPT console on the Administrator's computer

Attention: Warning, the WAPT console **MUST NOT** be installed on your Windows based WAPT Server.

The WAPT console must be installed on the workstation from which you manage your network.


WAPT Server

WAPT server

WAPT server is managed through a WAPT console installed on a Windows system. When installing the WAPT server on Windows, the console is installed by default and can be found under the start menu.

When installing the server on Linux, the WAPT client should be installed on an administration machine, then run from 'Start/All programs'.

To manually add a new host to the WAPT server, download the WAPT agent from the menu to the right. The agent has been properly configured by the server so the default parameters should work. Once the WAPT client has been installed, you can find it in your console.

You can deploy the WAPT agent using a GPO and the **WAPT deploy** downloader. See Deployment GPO creation for WAPTdeploy

For further information, be sure to check the documentation at wapt.fr or on mailing-list.

- **WAPT Server version: 1.5.1.19**
- **WAPT Agent version: N/A**
- **WAPT Setup version: 1.5.1.19**
- **WAPT Deploy version: 1.5.1.19**
- **DB status: OK (1.5.1.17)**
- **Disk space: 55% free**

WAPTSetup

For creation of the Wapt agent

WAPTDeploy

For setting up deployment GPO

Agent WAPT

For deploying onto user desktop

Contact

Contact Us
References
Actuality
Team

Tranquil IT Systems

Nous sommes une équipe de personnes passionnées dont le but est d'améliorer la vie de chacun. Nous élaborons des produits très performants pour résoudre vos problèmes. Nos produits sont créés pour optimiser les performances des PME.

Fig. 28: WAPT Server web interface

- if DNS records are properly configured, you should be able to access the WAPT web interface by visiting: https://srvwapt.mydomain.lan;
- click on *WAPTSetup* link on the right-hand side of the WAPT Server web page;
- start the executable installer as *Local Administrator* on the *Administrator*'s workstation;
- choose the language and click on *OK* to install the WAPT console;
- click on *OK* to go on to the next step;
- accept the licence terms and click on *Next* to go to next step;
- choose the WAPT destination folder (`C:\Program Files (x86)\wapt` by default);
- click on *Next* and choose your installation options (default value should be right for most installations);

Note:

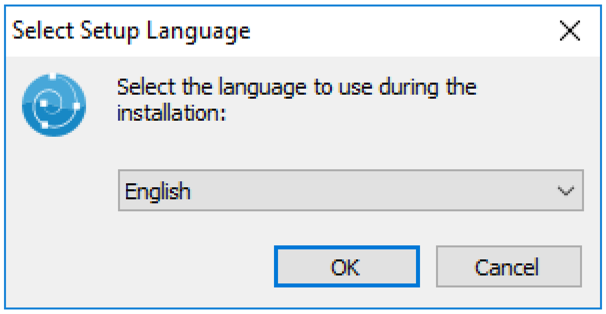
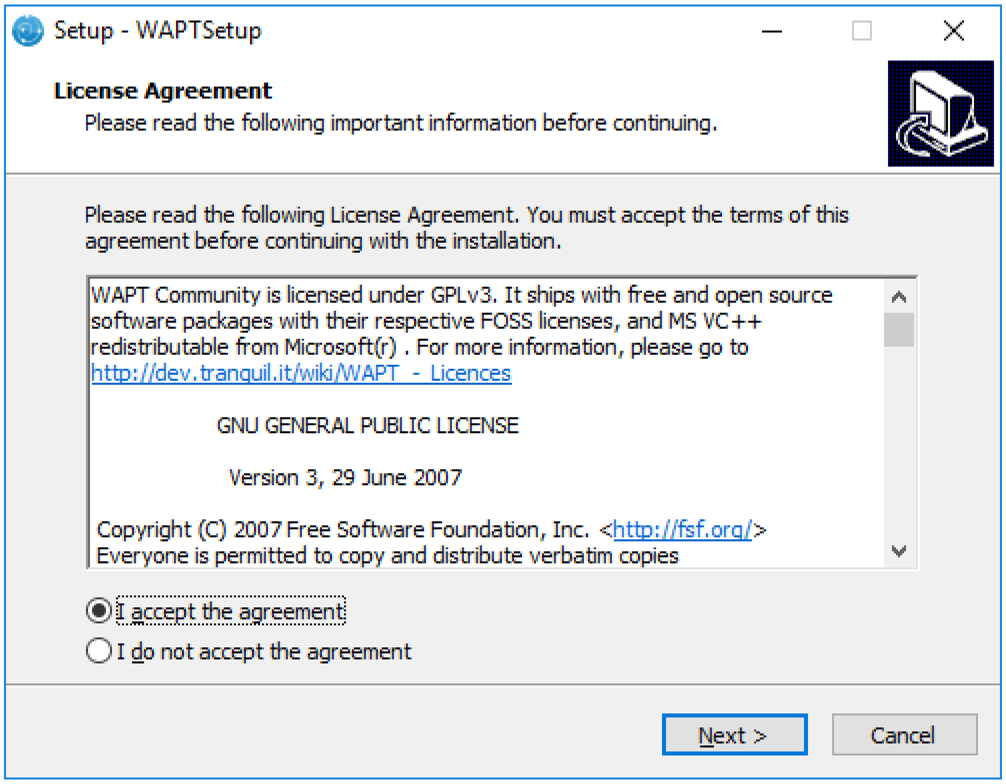


Fig. 29: Choose the language for WAPT


Setup - WAPTSetup

License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

WAPT Community is licensed under GPLv3. It ships with free and open source software packages with their respective FOSS licenses, and MS VC++ redistributable from Microsoft(r) . For more information, please go to http://dev.tranquil.it/wiki/WAPT_-_Licences

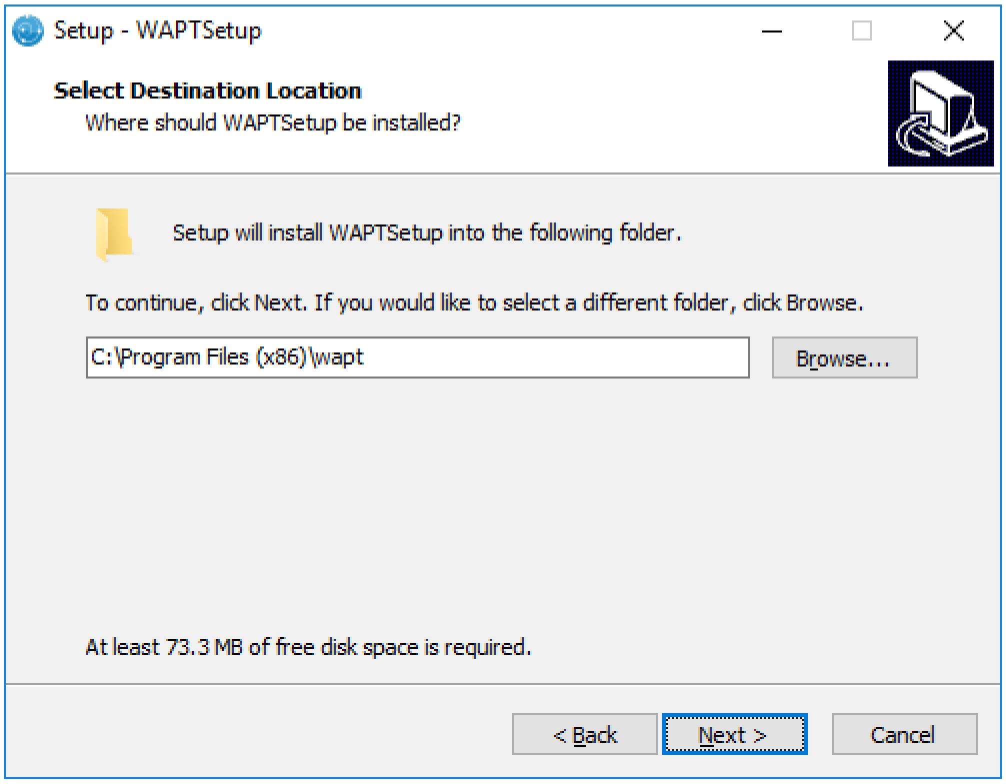
GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies

● I accept the agreement
○ I do not accept the agreement

Fig. 30: Accept the WAPT license terms



Select Destination Location

Where should WAPTSetup be installed?

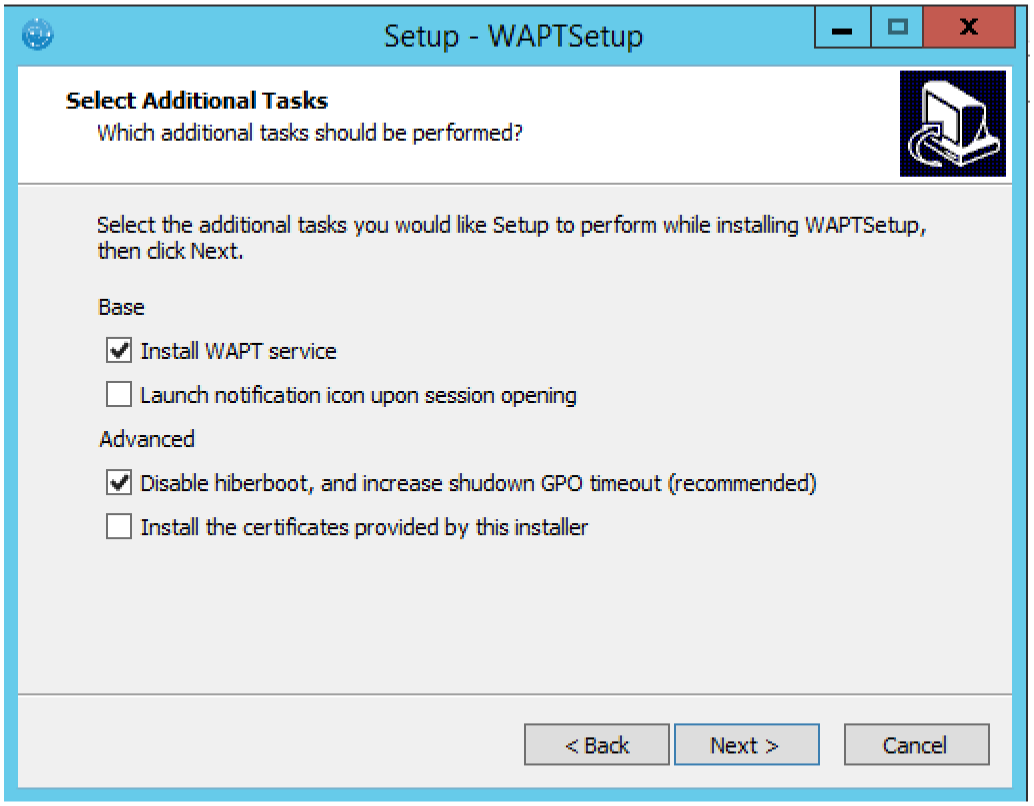
Setup will install WAPTSetup into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\wapt

At least 73.3 MB of free disk space is required.

Fig. 31: Choose the WAPT destination folder



Setup - WAPTSetup

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing WAPTSetup, then click Next.

Base

- [✓] Install WAPT service
- [ ] Launch notification icon upon session opening

Advanced

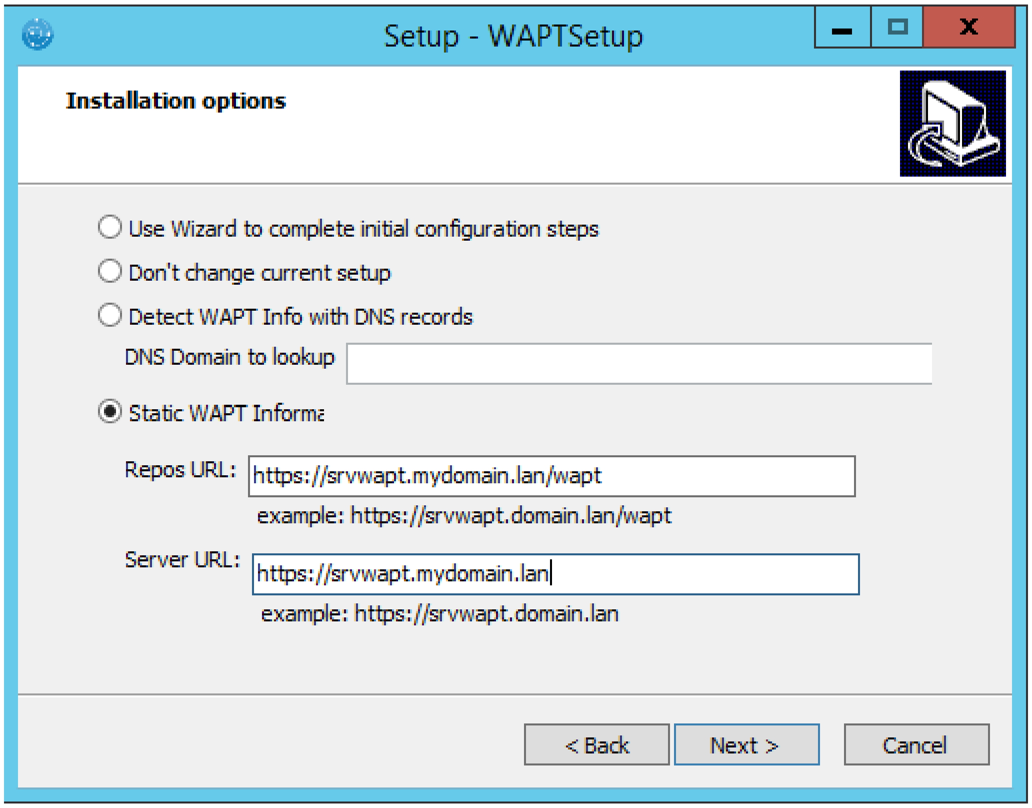
- [✓] Disable hiberboot, and increase shudown GPO timeout (recommended)
- [ ] Install the certificates provided by this installer

Fig. 32: Choose the installer's options

- check *Install WAPT service* if you want to have the WAPT service running on your *Administrator* workstation;
- check *Launch notification tray upon session opening* if you want to have the WAPT icon running in the tray by default;

Setting up the WAPT Server URL

Hint: Here, two choices become available to you, you may choose to be guided by the configuration wizard and go directly to *Using the WAPT console*, or you may choose to continue manually with the configuration.



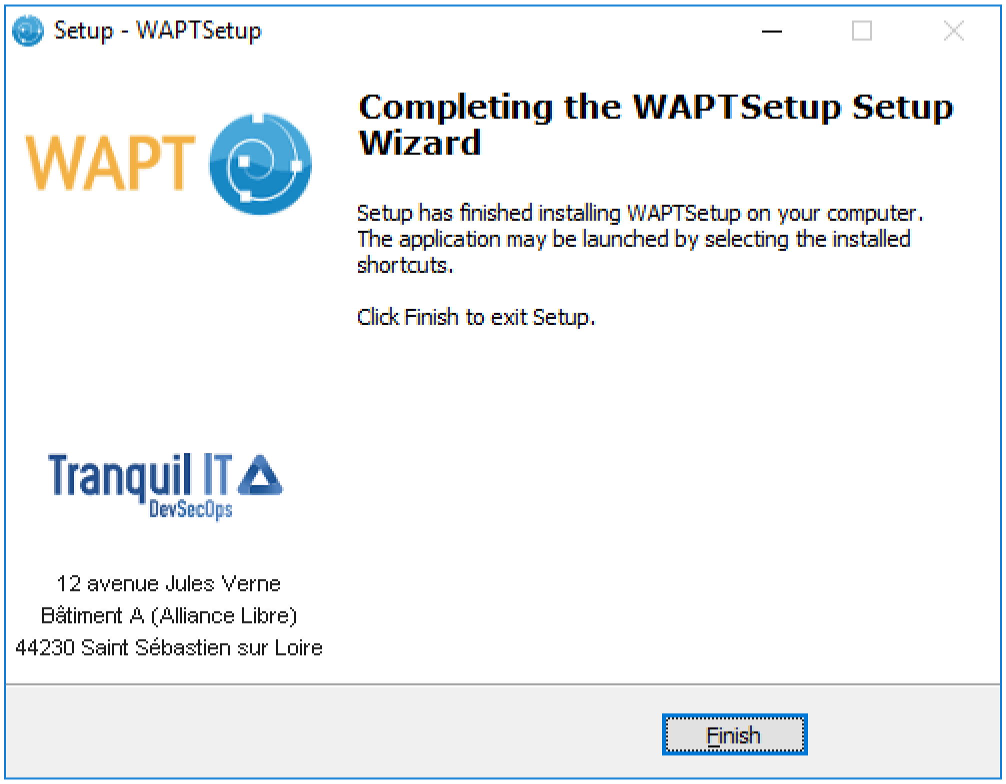
( ) Use Wizard to complete initial configuration steps
( ) Don't change current setup
( ) Detect WAPT Info with DNS records
DNS Domain to lookup
(●) Static WAPT Informa
Repos URL: https://srvwapt.mydomain.lan/wapt
example: https://srvwapt.domain.lan/wapt
Server URL: https://srvwapt.mydomain.lan
example: https://srvwapt.domain.lan

Fig. 33: Choose the WAPT repository and server

Note: Example:

- WAPT repository URL: http://srvwapt.mydomain.lan/wapt
- WAPT Server URL: https://srvwapt.mydomain.lan

- choose the language and click on *OK* to install the WAPT console;
- click *Next* and then *Install* to launch the installation, wait for the installation to complete, then click on *Finish* (leave default options);


Setup - WAPTSetup

Completing the WAPTSetup Setup Wizard

Setup has finished installing WAPTSetup on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

12 avenue Jules Verne
Bâtiment A (Alliance Libre)
44230 Saint Sébastien sur Loire

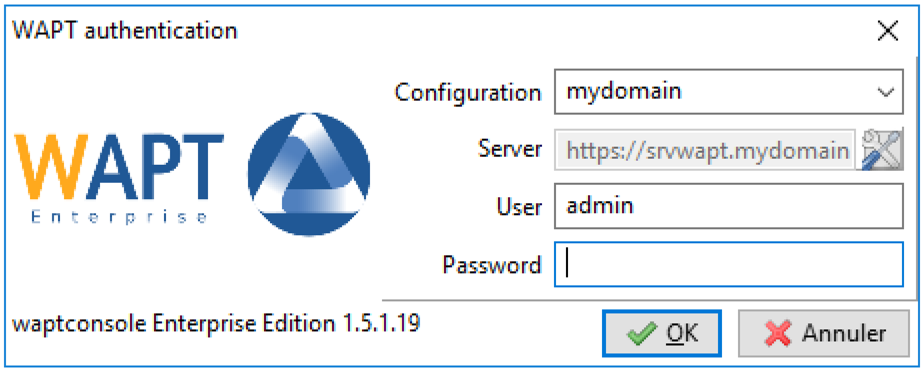
Fig. 34: Installation Wizard has finished

Note:

- check *Register this computer onto WAPT server* to register the computer with the WAPT Server;
- check *Update the list of available package on the main repository* to download the list of available packages from the WAPT repository;

Starting the WAPT console

- launch the WAPT console by looking for the binary `C:\Program Files (x86)\wapt\waptconsole.exe`
- on first start, you must start the WAPT console with elevated privileges. *Right-click on the WAPT console binary* → *Start as Local Administrator*;
- log into the WAPT console with the *SuperAdmin* login and password;


WAPT authentication

Configuration: mydomain

Server: https://srvwapt.mydomain

User: admin

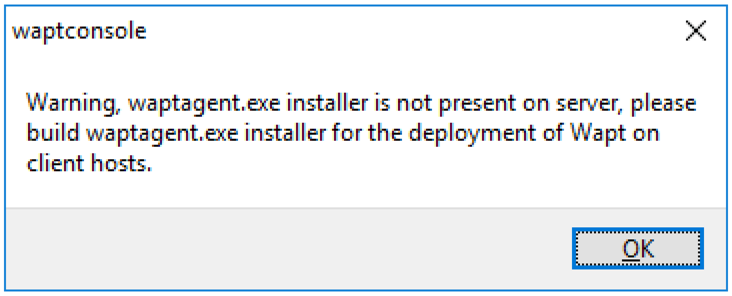
Password:

waptconsole Enterprise Edition 1.5.1.19

Fig. 35: WAPT Server connexion form

if you have any issue logging into the WAPT console, please refer to the FAQ: *Error message when opening the WAPT console*;

Note: A message may appear indicating that your WAPT agent version is obsolete or not yet present.


waptconsole

Warning, waptagent.exe installer is not present on server, please build waptagent.exe installer for the deployment of Wapt on client hosts.

Fig. 36: Mismatch error between the versions of the console and that of the agent

Go to the next step to *create your certificate*!!

6.6.2 Generating the Administrator's certificate for signing WAPT packages

Introduction

Naming conventions

- name of the private key: `wapt-private.pem`;
- public certificate signed with private key: `wapt-private.crt`;

Private key *wapt-private.pem*

Attention: The `wapt-private.pem` file is fundamental for security. It must be stored in a safe place and correctly protected.

The `wapt-private.pem` file is the private key, it is located by default in the `C:\private` folder of the *Administrator* workstation.

For better security this private key may be transfered on an external storage. A smartcard support is in the roadmap.

This private key will be used along with the certificate to sign packages before uploading them onto the WAPT repository.

public certificate signed with private key: `wapt-private.crt`

The `wapt-private.crt` file is the public certificate that is used along with the private key. It is by default created in the `C:\private` folder, copied in `C:\Program Files (x86)\wapt\ssl` of the Administrator and deployed on the desktops managed by the Administrator via a WAPT package or a GPO.

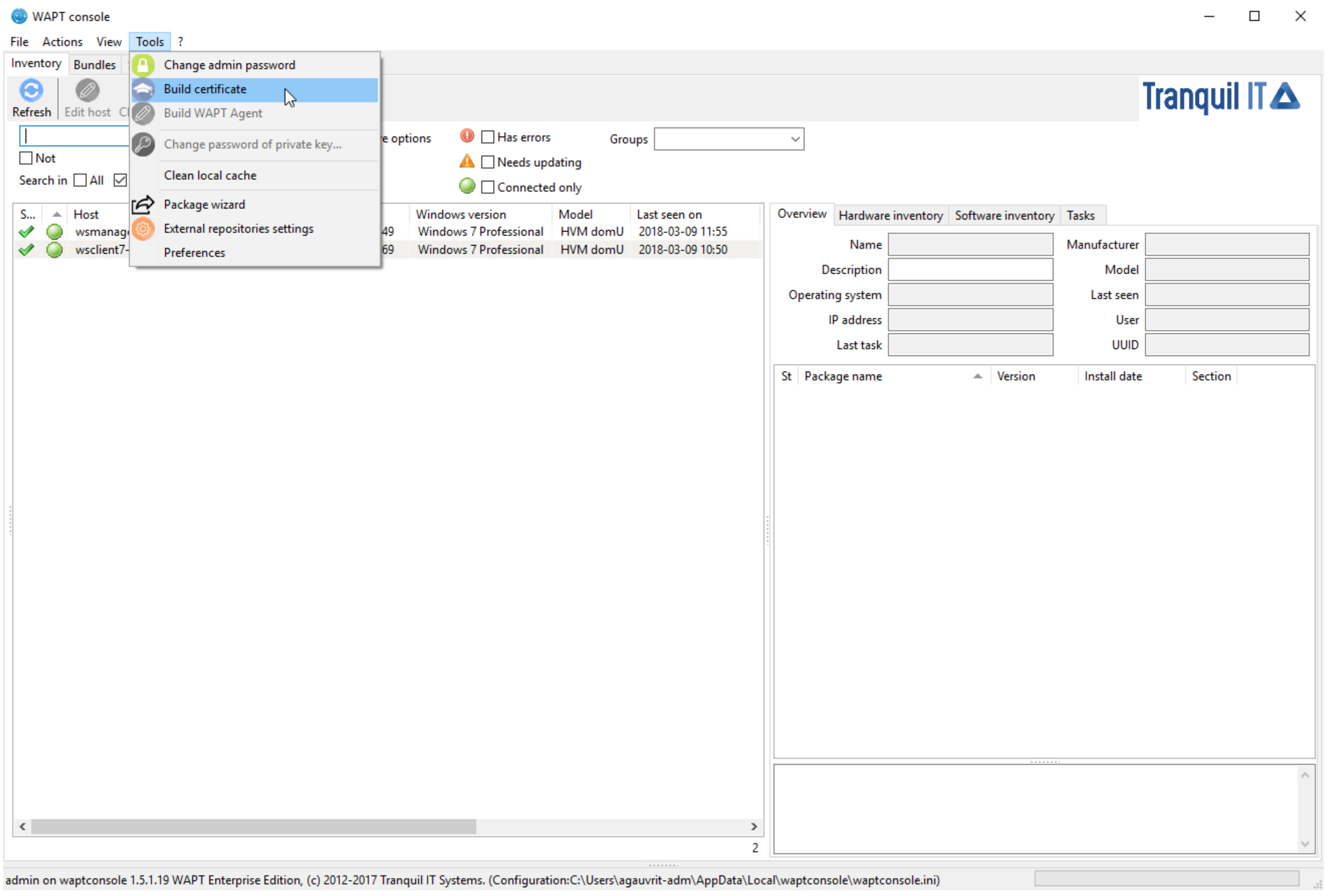
This certificate is used to validate the signature of packages before installation.

Creating a certificate

In the WAPT console go to *Tools* → *Create a certificate*;

Important: We have two different options:

- *create a certificate for the Community version*;
- *create a certificate for the Enterprise version*;

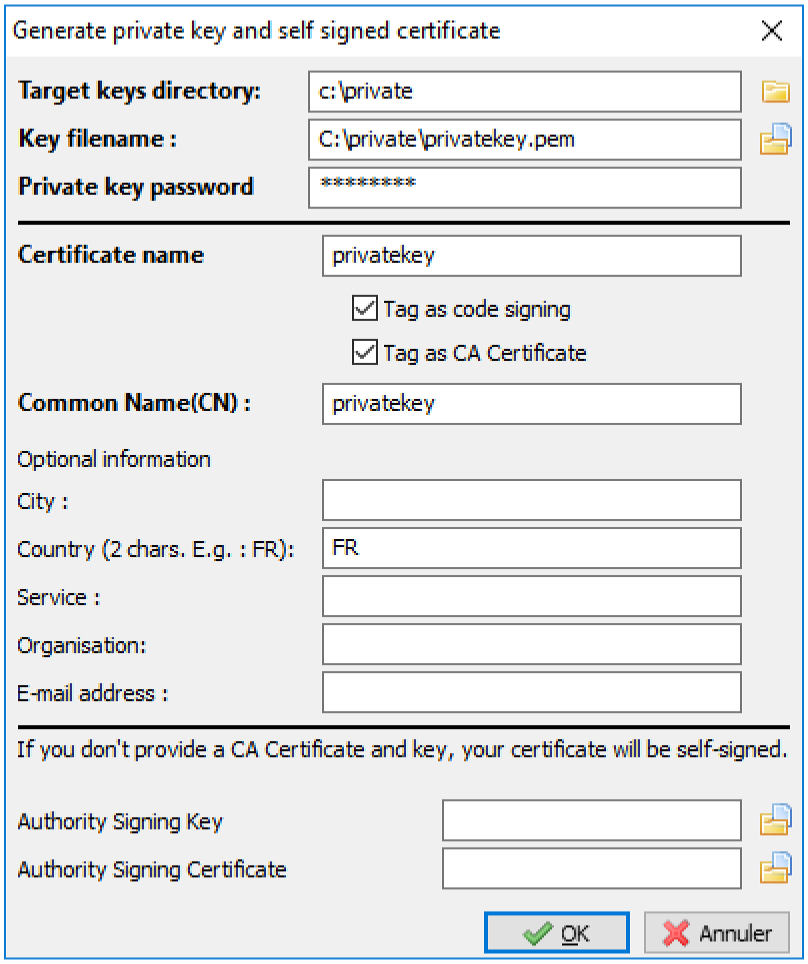


admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\waptconsole.ini)

Fig. 37: Creating a self-signed certificate

Creating a certificate - WAPT Community

- fill in the following fields:


Generate private key and self signed certificate

Target keys directory: c:\private

Key filename : C:\private\privatekey.pem

Private key password *********

Certificate name privatekey

☑ Tag as code signing

☑ Tag as CA Certificate

Common Name(CN) : privatekey

Optional information

City :

Country (2 chars. E.g. : FR): FR

Service :

Organisation:

E-mail address :

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key

Authority Signing Certificate

Fig. 38: Creating a self-signed certificate

- click on *OK* to go on to the next step;

Required informations are:

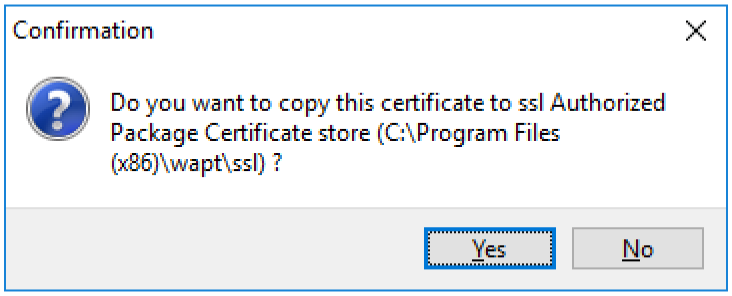
- *Destination folder*: folder where the private key and the public certificate will be stored: **required**;
- *Name of the private key*: name of the `.pem` and *Name of the private key*;
- *Private key password*: password for locking and unlocking the key: **required**;
- *Private key password*: password for locking and unlocking the key: **required**;
- *Common Name (CN)*: name of the Administrator: **required**;
- *Certificate name*: name of the `.crt` certificate: **required**;
- *Additional information*: additional details stored in the private key. This information will help with identifying the origin of the WAPT package: **optional**;

For a fresh install, you can follow the screenshot below.

Hint: The password complexity must comply with your *Organization*'s security requirements (eg. *ANSSI* password recommendations).

Danger:

- the path to your private key must not be in the installation path of WAPT (`C:\Program Files (x86)\wapt`);
- if your key is stored in `C:\Program Files (x86)\wapt`, your *Administrator* private key will be deployed on your clients, **absolutely a no go!**

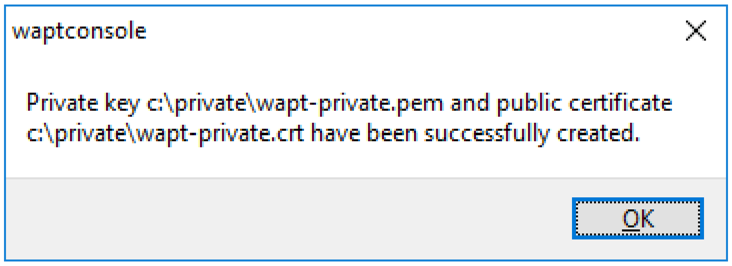

Confirmation

Do you want to copy this certificate to ssl Authorized Package Certificate store (C:\Program Files (x86)\wapt\ssl) ?

Fig. 39: Confirmation of the copy of the certificate in the ssl folder

- click on *Yes* to copy the newly generated certificate in the `C:\Program Files (x86)\wapt\ssl` folder. This certificate will be picked up during the compilation of the WAPT agent and deployed on the client computers;

If everything has gone well the following message will appear:


waptconsole

Private key c:\private\wapt-private.pem and public certificate c:\private\wapt-private.crt have been successfully created.

Fig. 40: Certificate generated successfully

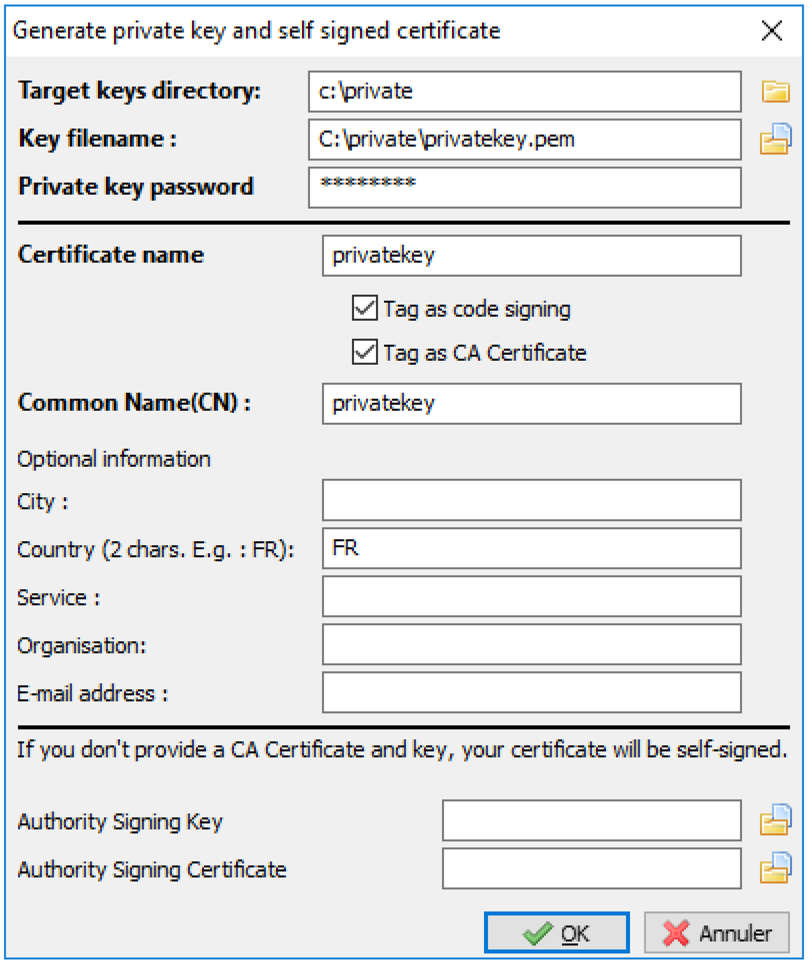
- click on *OK* to go on to the next step;

You may go on to the next step and *configure your WAPT console*!!

Creating a certificate - WAPT Enterprise

With WAPT Enterprise, you can create a Master key with a Certificate Authority flag that can both sign packages and sign new certificates.

Hint: In order to create new signed certificates for delegated, please refer to *Differentiating the role level in WAPT*.



Generate private key and self signed certificate

Target keys directory:	c:\private
Key filename :	C:\private\privatekey.pem
Private key password	*********
Certificate name	privatekey
	[✓] Tag as code signing
	[✓] Tag as CA Certificate
Common Name(CN) :	privatekey
Optional information	
City :	
Country (2 chars. E.g. : FR):	FR
Service :	
Organisation:	
E-mail address :	

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key	
Authority Signing Certificate	

Fig. 41: Creating a self-signed certificate

Required informations are:

- *Destination folder*: folder where the private key and the public certificate will be stored: **required**;
- *Name of the private key*: name of the `.pem` and *Name of the private key*: name of the `.pem` and `.crt` files: **required**;
- *Private key password*: password for locking and unlocking the key: **required**;
- *Common Name (CN)*: name of the Administrator: **required**;
- *Certificate name*: name of the `.crt` certificate: **required**;
- *Code signing*: check this box if the certificate/ key pair will be allowed to sign software packages: **required**;

- *CA certificate*: check this box if this certificate can be used to sign other certificates (main or intermediate Certificate Authority): **required**;
- *Additional information*: additional details stored in the private key. This information will help with identifying the origin of the WAPT package: **optional**;

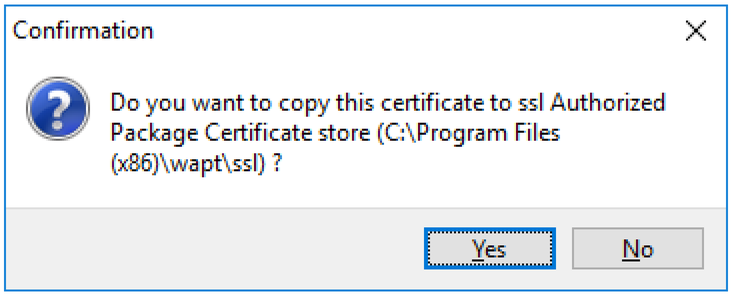
Hint: The password complexity must comply with your *Organization*'s security requirements (eg. *ANSSI* password recommendations).

Note: If your Organization is already equipped with an *Certificate Authority* (CA), you will have to fill the certificate and the key in the fields *CA Certificate* and *CA Key*.

With this procedure you can generate new certificates/ key pairs with or without **Code Signing** capability.

Danger:

- the path to your private key must not be in the installation path of WAPT (`C:\Program Files (x86)\wapt`);
- if your key is stored in `C:\Program Files (x86)\wapt`, your Administrator private key will be deployed on your clients, **absolutely a no go!**

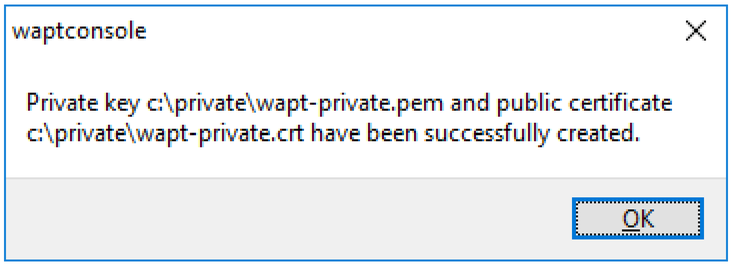

Confirmation

Do you want to copy this certificate to ssl Authorized Package Certificate store (C:\Program Files (x86)\wapt\ssl) ?

Fig. 42: Confirmation of the copy of the certificate in the ssl folder

- click on *Yes* to copy the newly generated certificate in the `C:\Program Files (x86)\wapt\ssl` folder. This certificate will be picked up during the compilation of the WAPT agent and deployed on the clients computers;

If everything has gone well the following message will appear:


waptconsole

Private key c:\private\wapt-private.pem and public certificate c:\private\wapt-private.crt have been successfully created.

Fig. 43: Certificate generated successfully

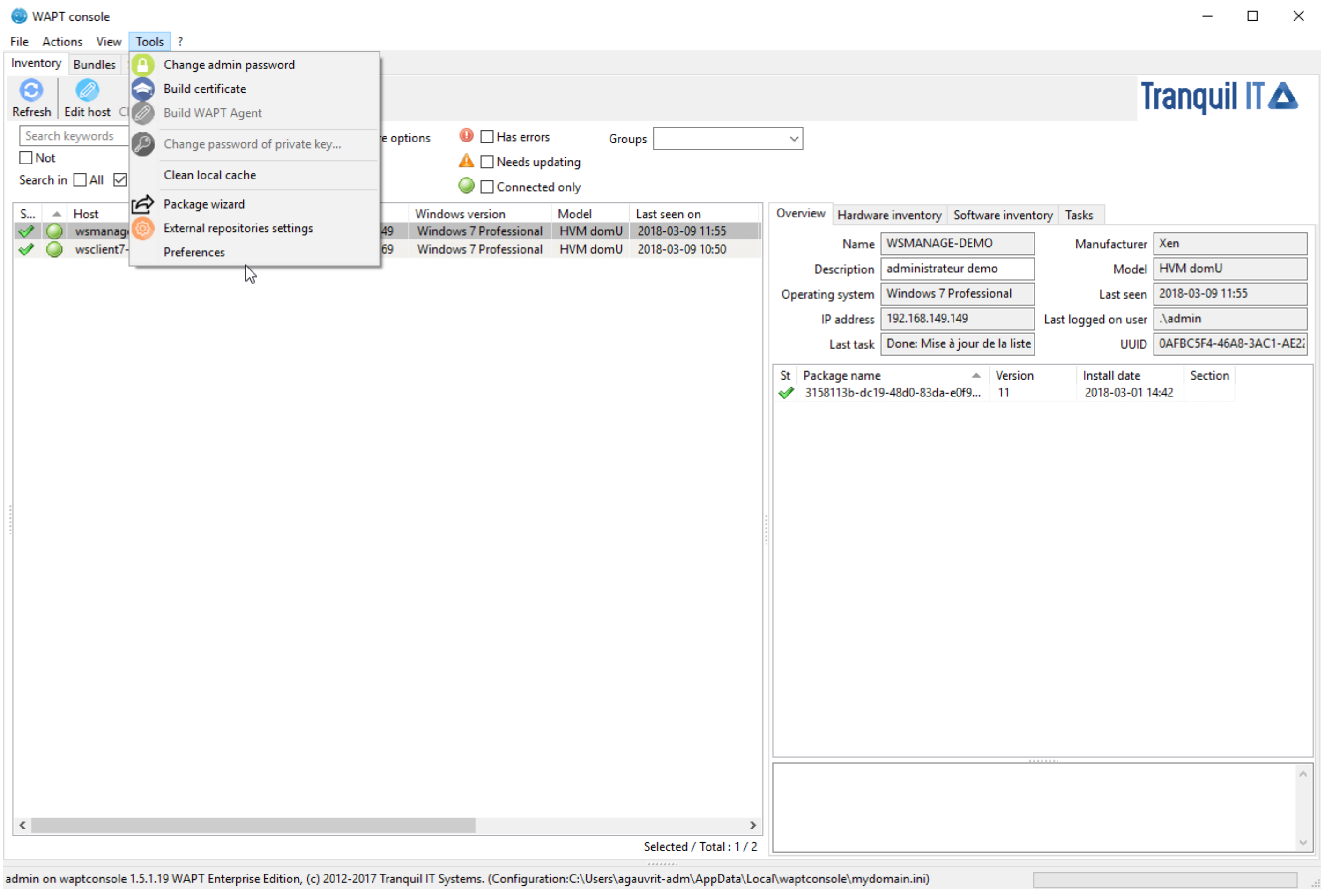
- click on *OK* to go on to the next step;

You may go on to the next step and *configure your WAPT console*.

6.6.3 Configuring the WAPT console

We will now properly configure the WAPT console.

Go to console configuration settings: *Tools* → *Preferences*.



S...	Host		Windows version	Model	Last seen on
✓	wsmanag	49	Windows 7 Professional	HVM domU	2018-03-09 11:55
✓	wsclient7-	69	Windows 7 Professional	HVM domU	2018-03-09 10:50

Selected / Total : 1 / 2

Name	WSMANAGE-DEMO	Manufacturer	Xen
Description	administrateur demo	Model	HVM domU
Operating system	Windows 7 Professional	Last seen	2018-03-09 11:55
IP address	192.168.149.149	Last logged on user	.\admin
Last task	Done: Mise à jour de la liste	UUID	0AFBC5F4-46A8-3AC1-AE2

St	Package name	Version	Install date	Section
✓	3158113b-dc19-48d0-83da-e0f9...	11	2018-03-01 14:42	

admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\mydomain.ini)

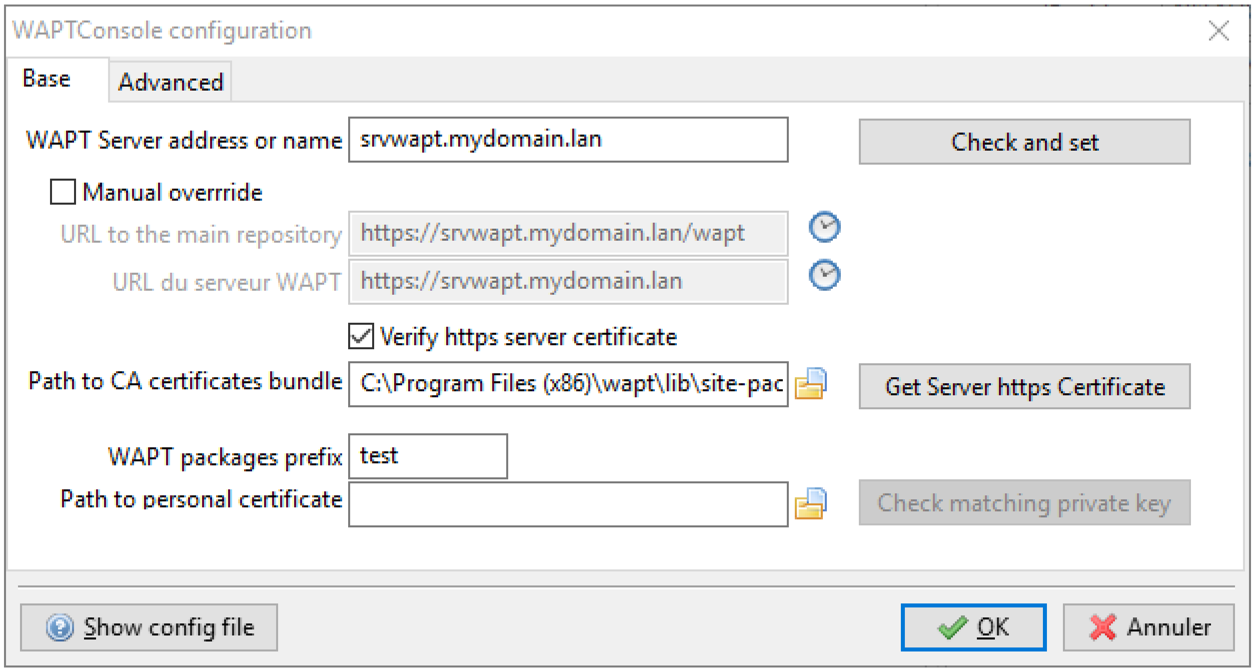
Fig. 44: WAPT console preferences menu

WAPT console preferences

Here we will set the prefix.

Attention: It is not trivial to change the prefix later as it would require re-signing all packages in all repositories, so choose your prefix well the first time!

The path to the private key has been automatically entered when we generated the private key in the previous step.


WAPTConsole configuration

Base | Advanced

WAPT Server address or name: srvwapt.mydomain.lan

[ ] Manual override

URL to the main repository: https://srvwapt.mydomain.lan/wapt

URL du serveur WAPT: https://srvwapt.mydomain.lan

[✓] Verify https server certificate

Path to CA certificates bundle: C:\Program Files (x86)\wapt\lib\site-pac

WAPT packages prefix: test

Path to personal certificate:

Fig. 45: Configuring the WAPT console

If you already have a key (ex: from a previous installation), you'll have to fill your old key here.

If you want to learn more on setting up the WAPT console, you may visit the documentation on *configuring the WAPT console*.

You may now go on to *create the WAPT agent*!!

6.6.4 Building the WAPT agent installer

Choosing the mode to uniquely identify the WAPT agents

In WAPT you can choose the unique identification mode of the agents.

When a WAPT agent registers the server must know if it is a new machine or if it is a machine already registered.

For this, the WAPT server looks at the unique number "UUID" in the inventory.

WAPT offers 3 modes of operation to help you distinguish between hosts, it is up to you to choose the mode that best suits you.

Attention: After choosing a mode of operation it is difficult to change it, think carefully!

Identifying the WAPT agents by their BIOS UUID (serial number)

This mode of operation makes it possible to identify the machines in the console in a physical manner.

If you replace a computer and give the new computer the same name as the previous one, you will have two computers that will appear in the WAPT console since you will have physically two different computers.

Note: Some vendors do inadequate work and assign the same BIOS UUIDs to entire batches of computers. In this case, WAPT will only see one computer . . .

Identifying the WAPT agent by host name

This mode of operation is similar to that in Active Directory. The machines are identified by their hostname.

Note: This mode does not work if several machines in your fleet share the same name. We all know it should not happen!!

Identifying the WAPT agents with a randomly generated UUID

This mode of operation allows PCs to be identified by their WAPT installation. Each installation of WAPT generates a unique random number. If you uninstall WAPT and then reinstall it, you will see a new pc appear in your console.

Building the WAPT agent installer

The **`waptagent`** binary is an InnoSetup installer.

Once the WAPT console has been installed on the *Administrator* computer, we have all files required to build the WAPT agent installer.

- files that will be used during building of the WAPT agent are located in `C:\Program Files (x86)\wapt`;
- installer source files (`.iss` files) are located in `C:\Program Files (x86)\wapt\waptsetup`;

Hint: Before building the WAPT agent, please verify the public certificate(s) in `C:\Program Files (x86)\wapt\ssl`.

If you wish to deploy other public certificates on your *Organization*'s computers that are equipped with WAPT, you will have to copy them in that folder.

Danger: **DO NOT COPY the private key** of any *Administrator* in `C:\Program Files (x86)\wapt`.

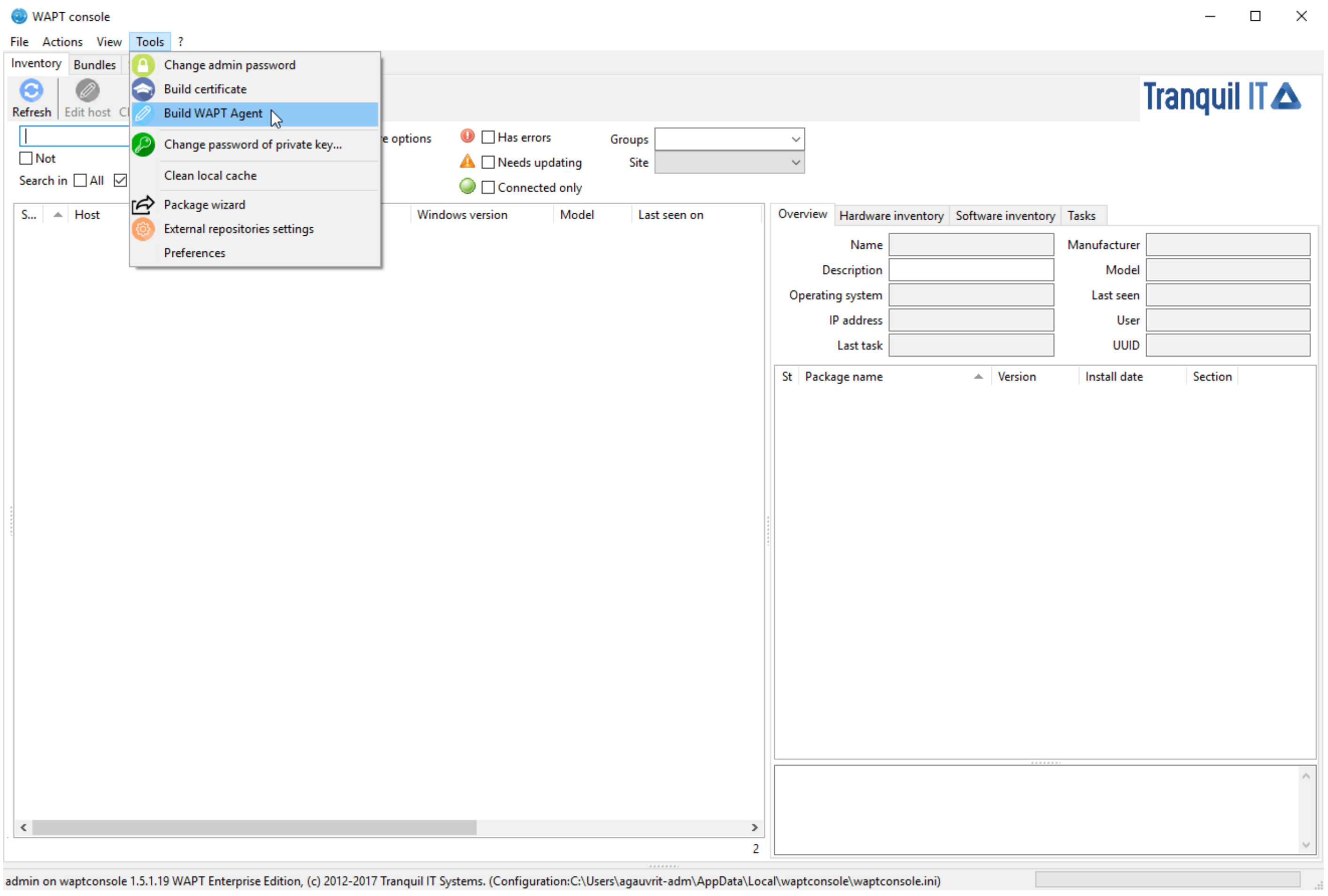
This folder is used when building the WAPT agent and the private keys would then be deployed on all the computers.

- In the WAPT console, go to *Tools → Build the WAPT agent*.
- fill in the informations that are necessary for the installer:
  - the field *Public certificate*: **required**;

    example : `C:\private\mydomain.crt`
  - the field *Address of the WAPT repository*: **required**;

    example : https://srvwapt.mydomain.lan/wapt
  - the field *Address of the WAPT Server*: **required**;

    example : https://srvwapt.mydomain.lan
  - the checkbox *Verify the WAPT Server HTTPS certificate*;
  - the field *Path to the bundle of certificates* to verify the HTTPS certificate of the WAPT Server;
  - the checkbox *Use Kerberos for registering WAPT agents*;
  - the field *Organization* to identify the origin of WAPT packages;
  - the field *Sign waptupgrade with both sha256 and sha1* can be ignored because it is only useful when upgrading from version 1.3;
  - the field *Use computer FQDN for UUID* and *Use random host UUID (for buggy BIOS)* (see explanation in the previous paragraph of this documentation);
  - the field *Enable AD Groups* enables the installation of profile packages based on the Active Directory groups of which the machine is a member. **This feature can degrade the performance of WAPT**;



admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\waptconsole.ini)

Fig. 46: Generate the WAPT agent from the console

- the field *Append host's profiles* allows you to define a list of WAPT packages to install obligatorily;
- the field *Automatic periodic packages audit scheduling* defines the frequency at which the WAPT agent checks whether it has audits to perform;
- Windows update section, refer to *this article on configuring WAPTWUA on the WAPT agent*;

Danger:

- The checkbox **Use Kerberos for the initial registration** must be checked **ONLY IF** you have followed the documentation on **Configuring the Kerberos authentication**.
- The checkbox **Verify the WAPT Server HTTPS certificate**must be checked **ONLY IF** you have followed the documentation on **Activating the verification of the SSL / TLS certificate**.

- provide the password for unlocking the private key:

Once the WAPT agent installer has finished building, a confirmation dialog pops up indicating that the **`waptagent`** binary has been successfully uploaded to https://srvwapt.mydomain.lan/wapt/.

Note: A warning shows up indicating that the GPO hash value should be changed. GPOs may be used to deploy the WAPT agent on your Organization's computer.

Updating the WAPT agents

The `test-waptupgrade` package has also been uploaded on the repository.

The `test-waptupgrade` package contains the WAPT agent with arguments specified during the installation of WAPT on your Administrator's computer.

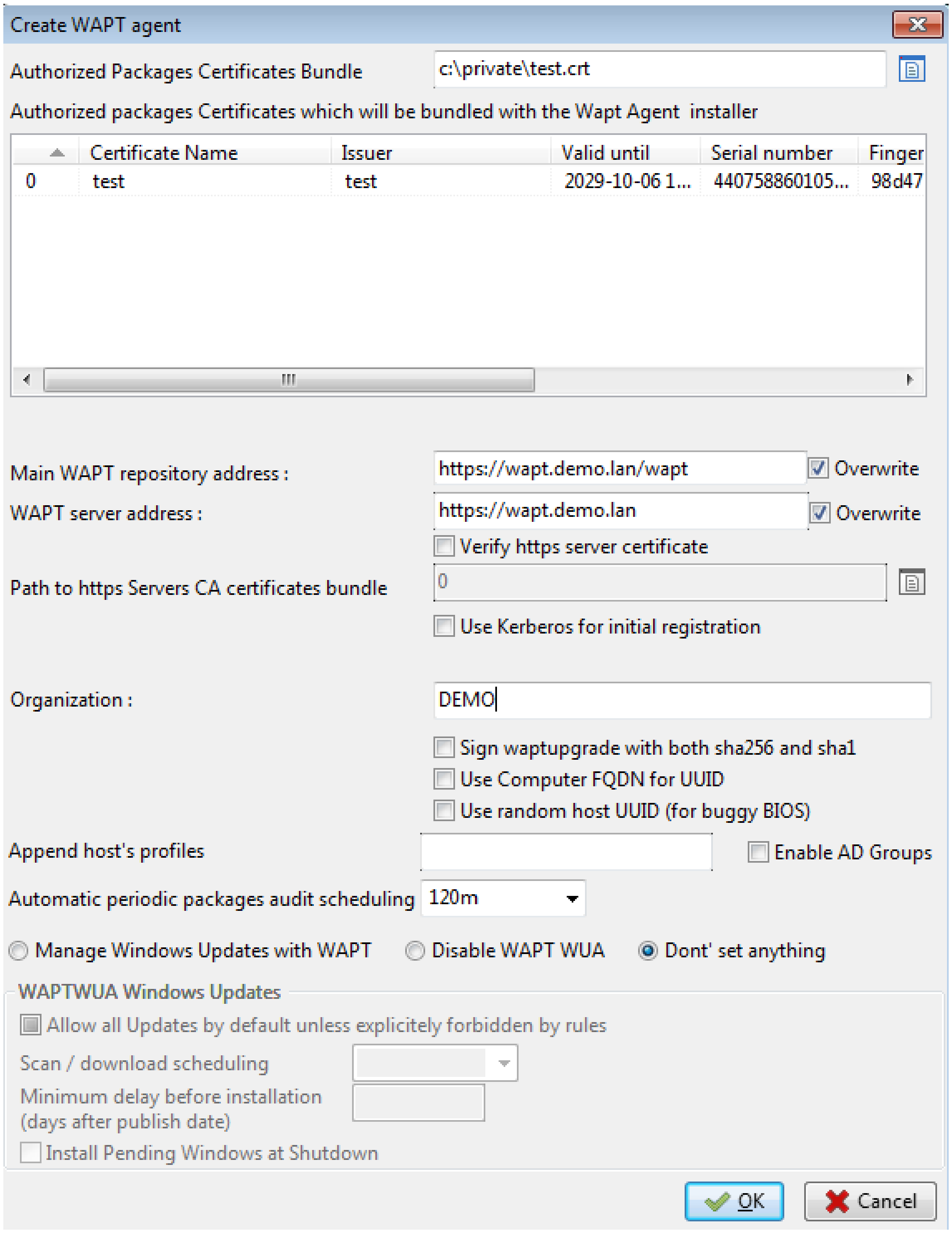
Note: This package is a standard WAPT package designed to upgrade WAPT agents on client machines.

Upgrading the WAPT agents using the *xxx-waptupgrade* package is a two step process:

- first the package copies the new `waptagent.exe` file on the client computer and creates a new scheduled task that will run **`waptagent.exe`** with predefined installation flags two minutes after the creation of the scheduled task. At that point the package itself is installed and the inventory on the server shows the package installation as *OK*, with correct version installed, but the inventory will still show the old version as the agent is not yet updated.
- after two minutes the scheduled task starts and runs **`waptagent.exe`**. **`waptagent.exe`** shutdowns the local WAPT service, upgrades the local WAPT install, and then restarts the service. The scheduled task is then automatically removed and the WAPT agent sends back its inventory to the WAPT server. Now the inventory on the server will show the new version of the agent.

From an administrator point of view, looking at the console you will see the following steps:

- *xxx-waptupgrade* package starts being installed;
- *xxx-waptupgrade* is installed, the machine is up to date from a package list point of view, but the version in the inventory is still the old version of the WAPT agent;
- after two minutes the computer connectivity status switches to *disconnected* as the WAPT agent is updated;


Create WAPT agent

Authorized Packages Certificates Bundle: c:\private\test.crt

Authorized packages Certificates which will be bundled with the Wapt Agent installer

	Certificate Name	Issuer	Valid until	Serial number	Finger
0	test	test	2029-10-06 1...	440758860105...	98d47

Main WAPT repository address : https://wapt.demo.lan/wapt [✓] Overwrite

WAPT server address : https://wapt.demo.lan [✓] Overwrite

[ ] Verify https server certificate

Path to https Servers CA certificates bundle: 0

[ ] Use Kerberos for initial registration

Organization : DEMO

[ ] Sign waptupgrade with both sha256 and sha1

[ ] Use Computer FQDN for UUID

[ ] Use random host UUID (for buggy BIOS)

Append host's profiles: [ ] Enable AD Groups

Automatic periodic packages audit scheduling: 120m

Manage Windows Updates with WAPT / Disable WAPT WUA / [✓ Dont' set anything]

WAPTWUA Windows Updates

[ ] Allow all Updates by default unless explicitely forbidden by rules

Scan / download scheduling

Minimum delay before installation (days after publish date)

[ ] Install Pending Windows at Shutdown

OK Cancel

Fig. 47: Fill in the informations on your Organization

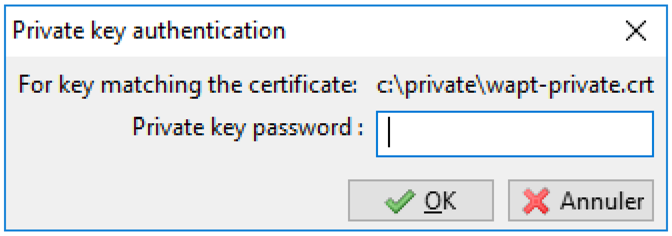

Private key authentication
For key matching the certificate: c:\private\wapt-private.crt
Private key password :
OK
Annuler

Fig. 48: Provide the password for unlocking the private key

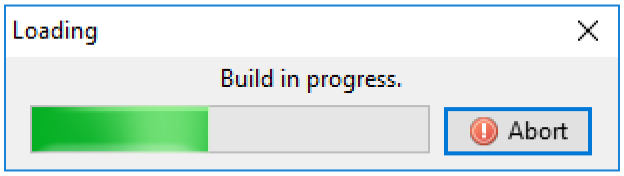
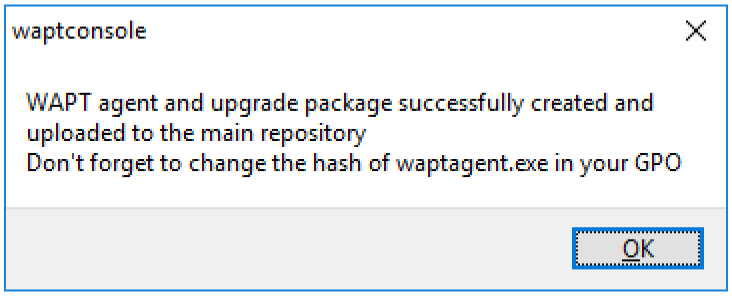
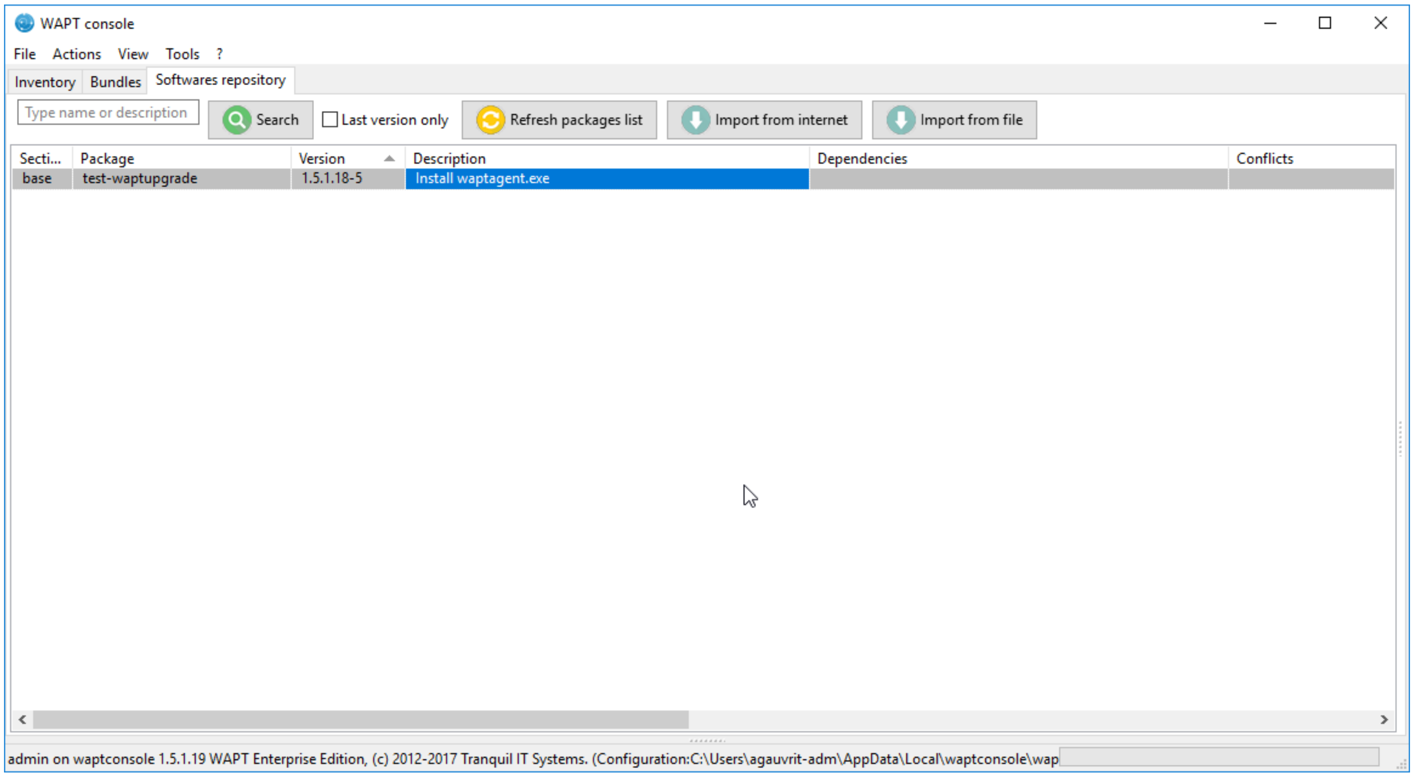


Fig. 49: Progression of WAPT agent installer building



WAPT agent and upgrade package successfully created and uploaded to the main repository
Don't forget to change the hash of waptagent.exe in your GPO

Fig. 50: Confirmation of the WAPT agent loading onto WAPT repository



Secti...	Package	Version	Description	Dependencies	Conflicts
base	test-waptupgrade	1.5.1.18-5	Install waptagent.exe		

admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\wap

Fig. 51: New WAPT agent in the repository

- after around two minutes the client computer gets back up online in the console and updates its inventory and shows the new version;

What can go wrong during the upgrades

- the most common issue with the upgrading process is the local antivirus blocking the installation (WAPT is a software installer that keeps a websocket opened to a central management server, so this behavior may be flagged as suspicious by an antivirus, even though this method is the basis of end point management…). **If you have an issue when deploying the upgrade, please check your antivirus console and whitelist the waptagent.exe**. Another option is to re-sign the **`waptagent.exe`** binary if your organization has an internal code signing certificate;
- the second most common issue is that for some reason another program is locking a *`DLL`* that ships with WAPT. This can happen with poorly designed software installers that pick up the local %PATH% variable first and then find WAPTs own openssl or python DLL;
- the third most common issue is a defective Windows install that does not run scheduled tasks properly, and yes we have seen this!!

6.6.5 Deploying the WAPT agent for Windows

Two methods are available to deploy the **`waptagent.exe`**.

The first method is manual and the procedure must be applied on each machine.

The second one is automated and relies on a GPO.

Note: The **`waptagent.exe`** installer is available at https://srvwapt.mydomain.lan/wapt/waptagent.exe.

If you do not sign the **`waptagent.exe`** installer with a commercial `Code Signing` certificate or a `Code Signing` certificate issued by the *Certificate Authority* of your Organization after having generated it, web browsers will show a warning message when downloading the installer. To remove the warning message, you must sign the *`.exe`* with a `Code Signing` certificate that can be verified by a CA bundle stored in the machine's certificate store.

Hint: When to deploy the WAPT agent manually?

Manual deployment method is efficient in these cases:

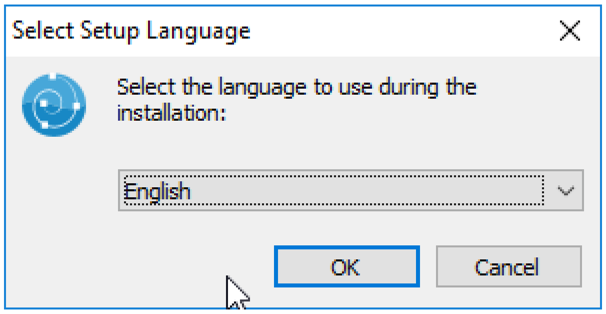
- testing WAPT;
- using WAPT in an organization with a small number of computers, etc;

Deploying waptagent.exe manually

Attention: This operation requires *Local Administrator* rights on the local computer.

Installing *waptagent.exe*

- choose the language and click on *Next* to go to next step;

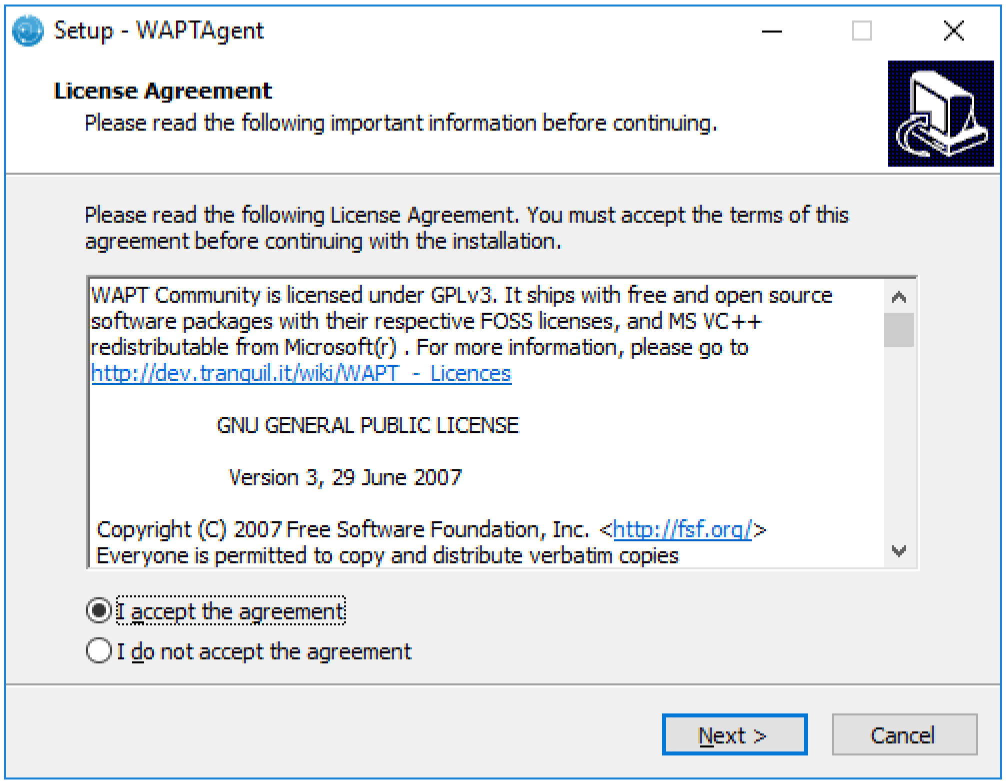


Select the language to use during the installation:

English

Fig. 52: Choose the installation language

- accept the license terms and click on *Next* to go to next step;



License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

WAPT Community is licensed under GPLv3. It ships with free and open source software packages with their respective FOSS licenses, and MS VC++ redistributable from Microsoft(r) . For more information, please go to http://dev.tranquil.it/wiki/WAPT_-_Licences

GNU GENERAL PUBLIC LICENSE

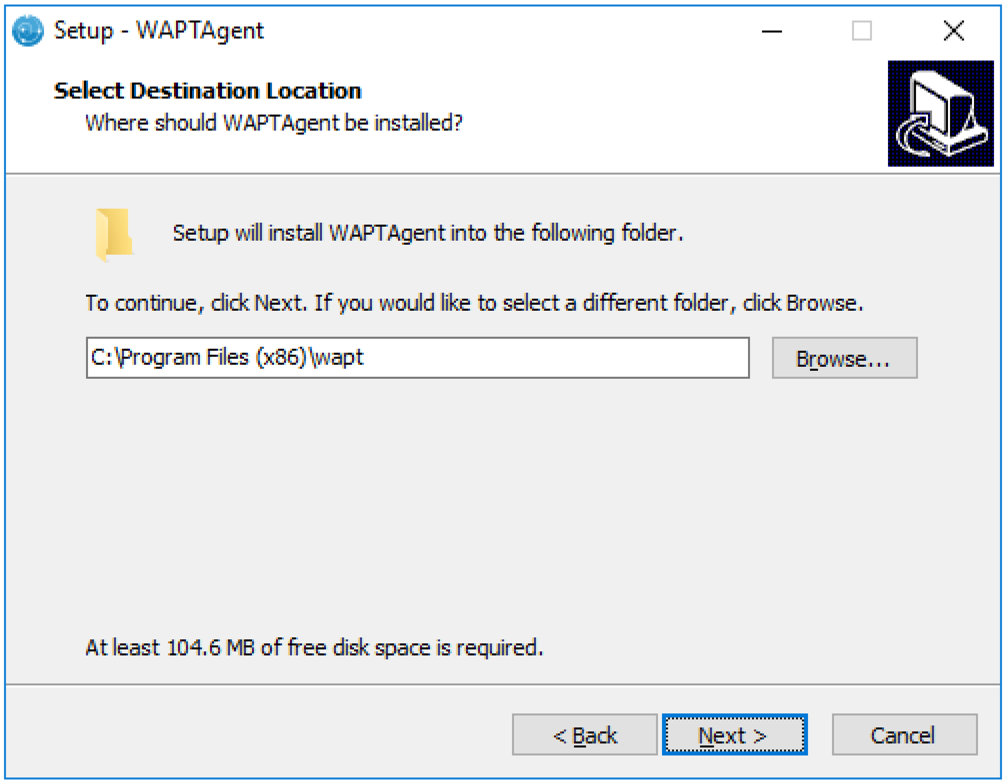
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies

[✓ I accept the agreement]
I do not accept the agreement

Fig. 53: Accepting the EULA

- choose the installation directory and click on *Next* to go to next step;


Setup - WAPTAgent

Select Destination Location

Where should WAPTAgent be installed?

Setup will install WAPTAgent into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\wapt

At least 104.6 MB of free disk space is required.

Fig. 54: Select the installation folder for the WAPT agent

- choose the additional parameters and click on *Next* to go to next step;

Hint: leave *Force-reinstall VC++ enabled* checked. If the option box is ticked it is because its installation is necessary.

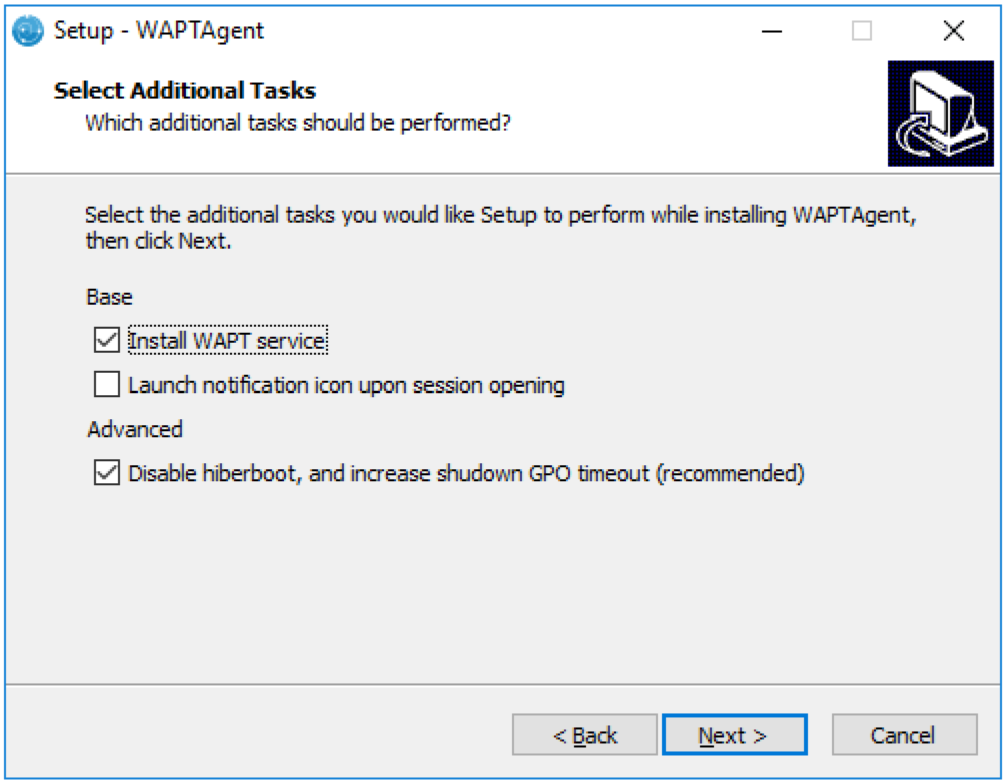
- choose the WAPT repository and the WAPT Server and click on *Next* to go to next step;
- install the WAPT agent by clicking on *Install*;
- wait for the installation of the WAPT agent to finish, then click on *Finish* to exit;

The installation of the WAPT agent is finished. With **`cmd.exe`**, launch a **`register`** to register the machine with the WAPT Server and an **`update`** to display the list of available WAPT packages.

Note:

- tick *Register this host on WAPT Server* to register the computer on the WAPT inventory server;
- tick *Update package list from repository* to update the list of available packages;

To manage your Organization's WAPT clients, visit the *documentation on using the WAPT console*.



Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing WAPTAgent, then click Next.

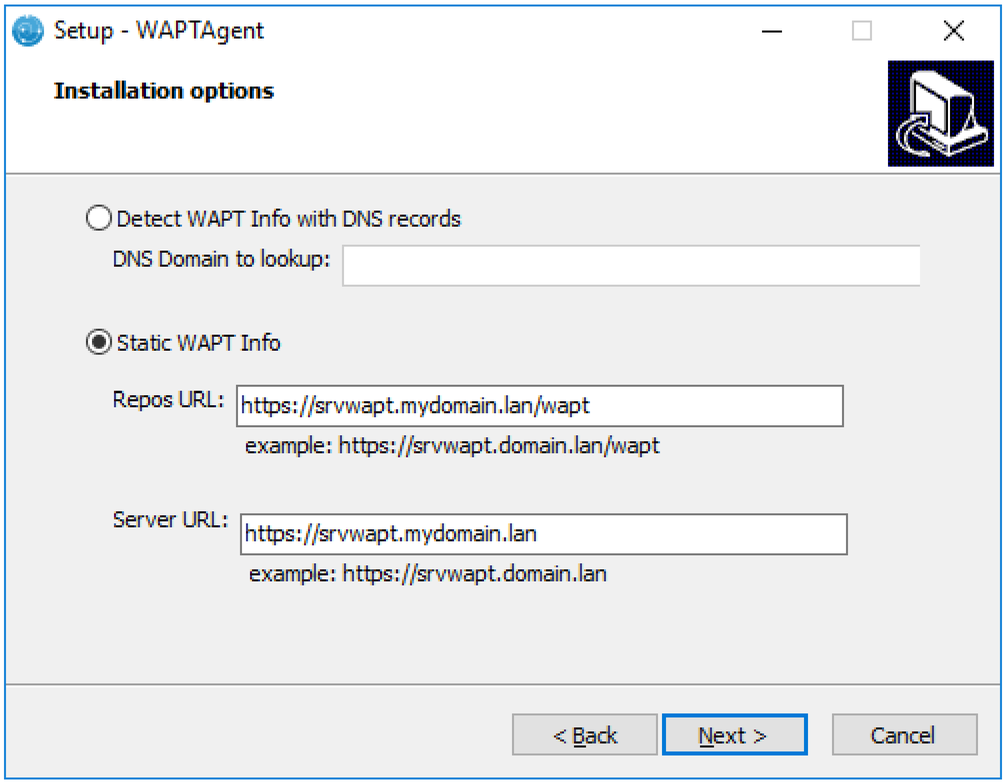
Base

- [✓] Install WAPT service
- [ ] Launch notification icon upon session opening

Advanced

- [✓] Disable hiberboot, and increase shudown GPO timeout (recommended)

Fig. 55: Choose the installer's options


Setup - WAPTAgent

Installation options

Detect WAPT Info with DNS records

DNS Domain to lookup:

Static WAPT Info [✓]

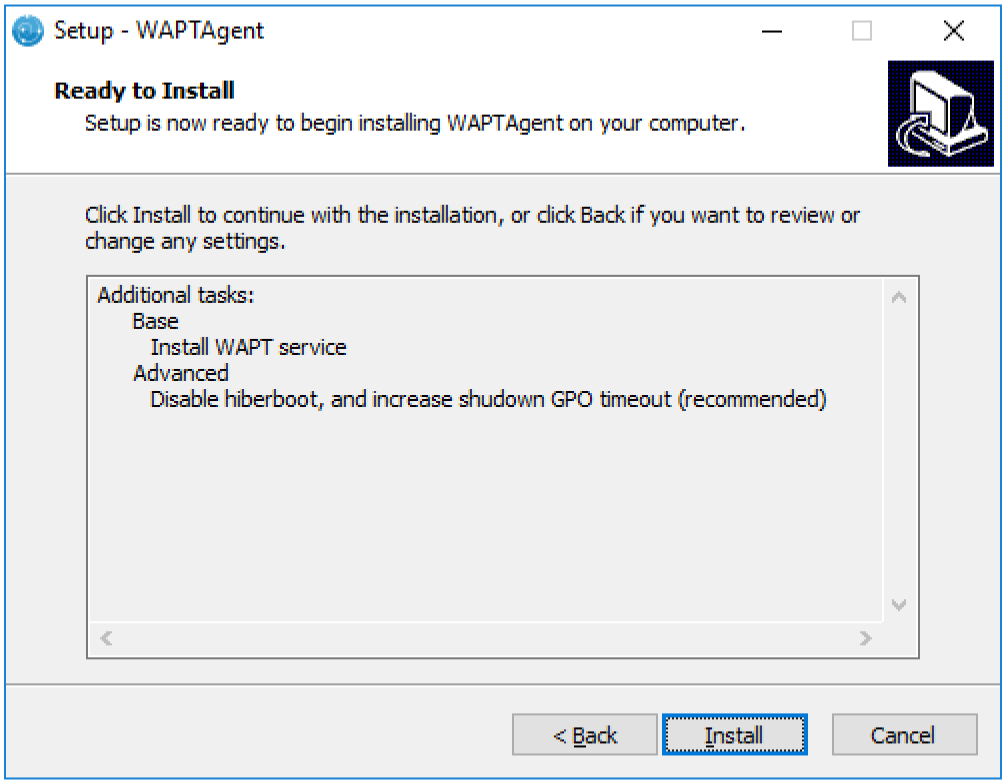
Repos URL: https://srvwapt.mydomain.lan/wapt

example: https://srvwapt.domain.lan/wapt

Server URL: https://srvwapt.mydomain.lan

example: https://srvwapt.domain.lan

Fig. 56: Choose the WAPT repository and server


Setup - WAPTAgent

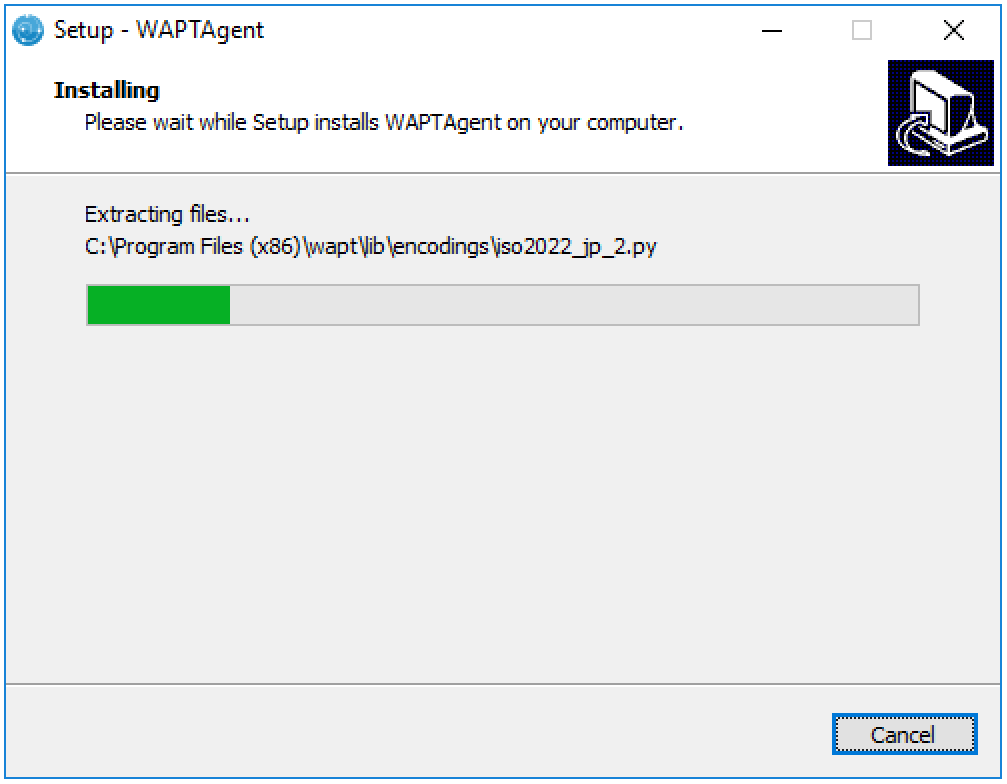
Ready to Install

Setup is now ready to begin installing WAPTAgent on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Additional tasks:
    Base
      Install WAPT service
    Advanced
      Disable hiberboot, and increase shudown GPO timeout (recommended)
```

Fig. 57: Summary of installation options



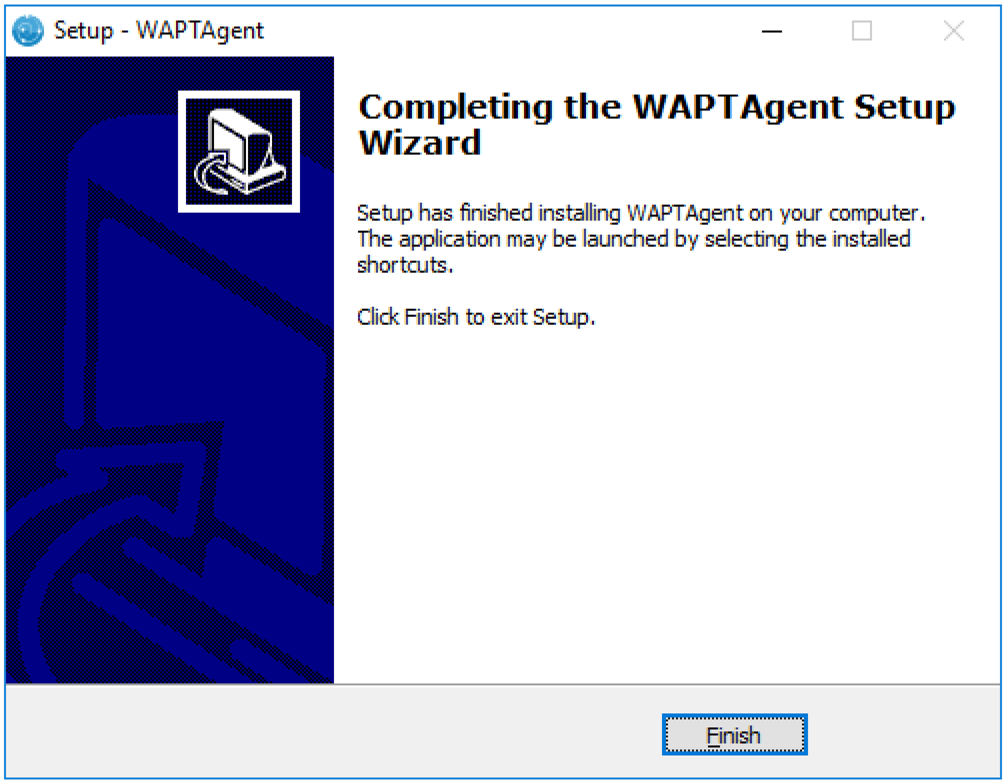
Installing

Please wait while Setup installs WAPTAgent on your computer.

Extracting files...

C:\Program Files (x86)\wapt\lib\encodings\iso2022_jp_2.py

Fig. 58: Installation in progress



Completing the WAPTAgent Setup Wizard

Setup has finished installing WAPTAgent on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

Fig. 59: End of WAPT agent installation

Automatically deploying the WAPT agents

Important: Technical pre-requisites

Advanced network and system administration knowledge is required to achieve this procedure. A properly configured network will ensure its success.

Hint: When to deploy the WAPT agent automatically? The following method is useful in these cases:

- a large organization with many computers;
- a Samba Active Directory or Microsoft Active Directory for which you have enough administration privileges;
- the security and the traceability of actions are important to you or to your *Organization*;
- or just simply, you prefer to act with your head instead of your feet ;)

Deploying the WAPT agents silently

Without waptdeploy

waptagent.exe is an *InnoSetup* installer, it can be executed with these silent switches:

```
waptagent.exe /VERYSILENT
```

- Additional arguments available for waptdeploy

Table 2: Description of available options for deploying the WAPT agent silently

Options	Description
`/dnsdomain` = mydomain.lan	Domain in `wapt-get.ini` filled in during installation.
`/wapt_server` = https://srvwapt.mydomain.lan	URL of the WAPT server in `wapt-get.ini` filled in during installation
`/repo_url` = https://repo1.mydomain.lan/wapt	URL of the WAPT repository in `wapt-get.ini` filled in during installation.
`/StartPackages` = basic-group	Group of WAPT packages to install by default.
`/verify_cert=` = 1 or relative path `ssl\server\srvwapt.mydomain.lan.crt`	Value of `verify_cert` entered during installation
`/CopyServersTrustedCA` = path to a bundle to copy to `ssl\server`.	Certificate bundle for https connections (to be defined by `verify_cert`)
`/CopypackagesTrustedCA` = path to a certificate bundle to copy into `ssl`	Certificate bundle for verifying package signatures

Hint: The `iss` file for the InnoSetup installer is available here: `C:\Program Files (x86)\wapt\waptsetup\waptsetup.iss`.

You may choose to adapt it to your specific needs. Once modified, you'll just have to recreate a **`waptagent`**.

To learn more about the options available with *InnoSetup*, visit this documentation.

With waptdeploy

`waptdeploy` is a small binary that:

- checks the version of the WAPT agent;
- downloads via https the **`waptagent.exe`** installer;
- launches the silent installer with arguments (checked options defined during the compilation of the WAPT agent);

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

- updates the WAPT Server with the WAPT agent status (WAPT version, package status);

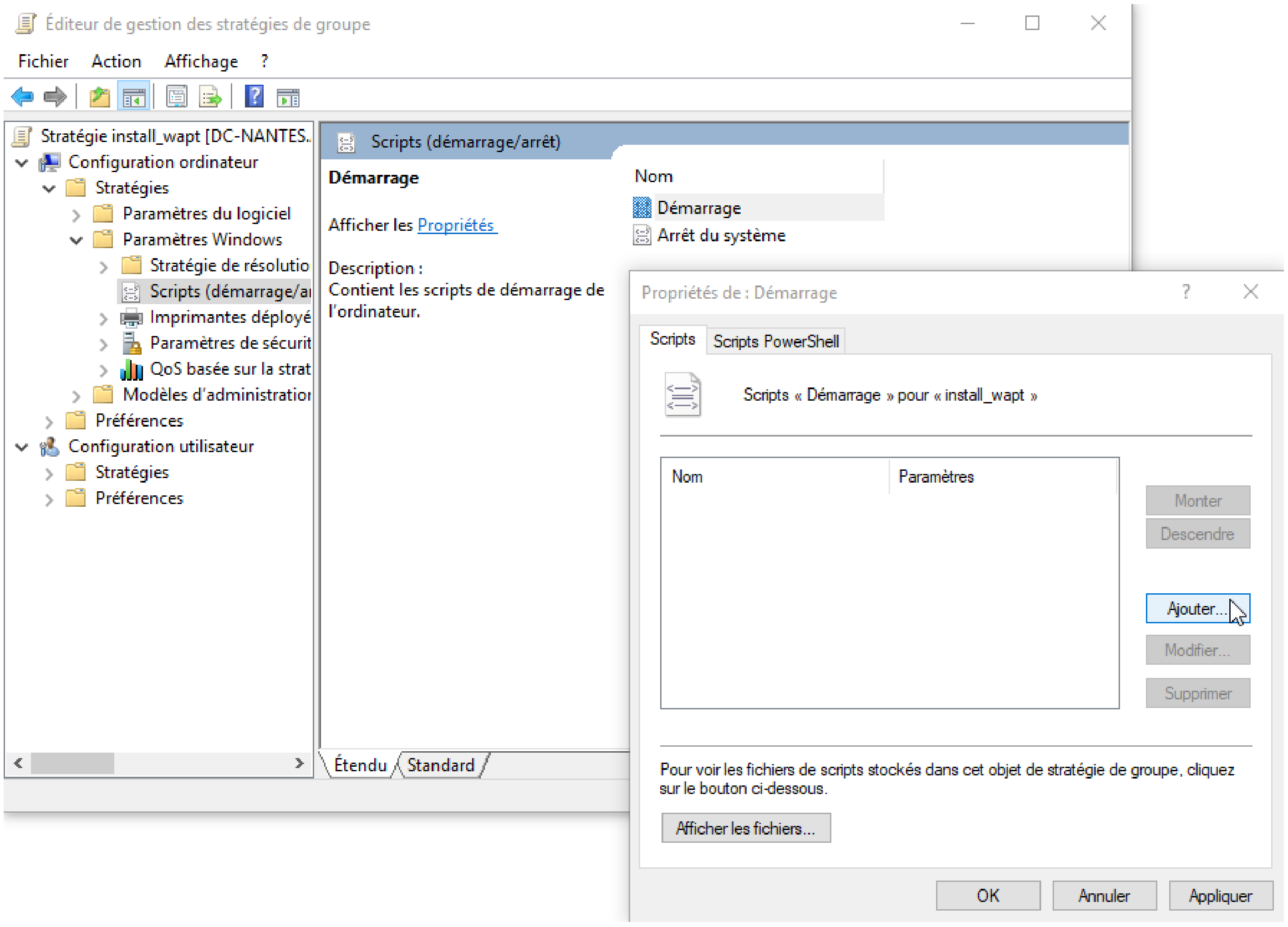
Note: **`waptdeploy`** must be started as *Local Administrator*, that is why we advise you to use a GPO.

Creating a GPO to deploy the WAPT agents

Download `waptdeploy.exe` by visiting: https://wapt.tranquil.it/wapt/releases/latest/waptdeploy.exe.

Creating the GPO

- create a new group strategy called **install_wapt** on the Active Directory server (Microsoft or Samba-AD);
- add a new strategy: *Computer configuration* → *Strategies* → *Windows configuration* → *Scripts* → *Startup* → *Add*;



Description :
Contient les scripts de démarrage de l'ordinateur.

Scripts « Démarrage » pour « install_wapt »

Pour voir les fichiers de scripts stockés dans cet objet de stratégie de groupe, cliquez sur le bouton ci-dessous.

Fig. 60: Creating a group strategy to deploy the WAPT agent

- click on *Browse* to select the `waptdeploy.exe` script;
- copy `waptdeploy.exe` in the destination folder;
- click on *Open* to import the `waptdeploy.exe` script;
- click on *Open* to confirm the importation of the **`waptdeploy`** binary;

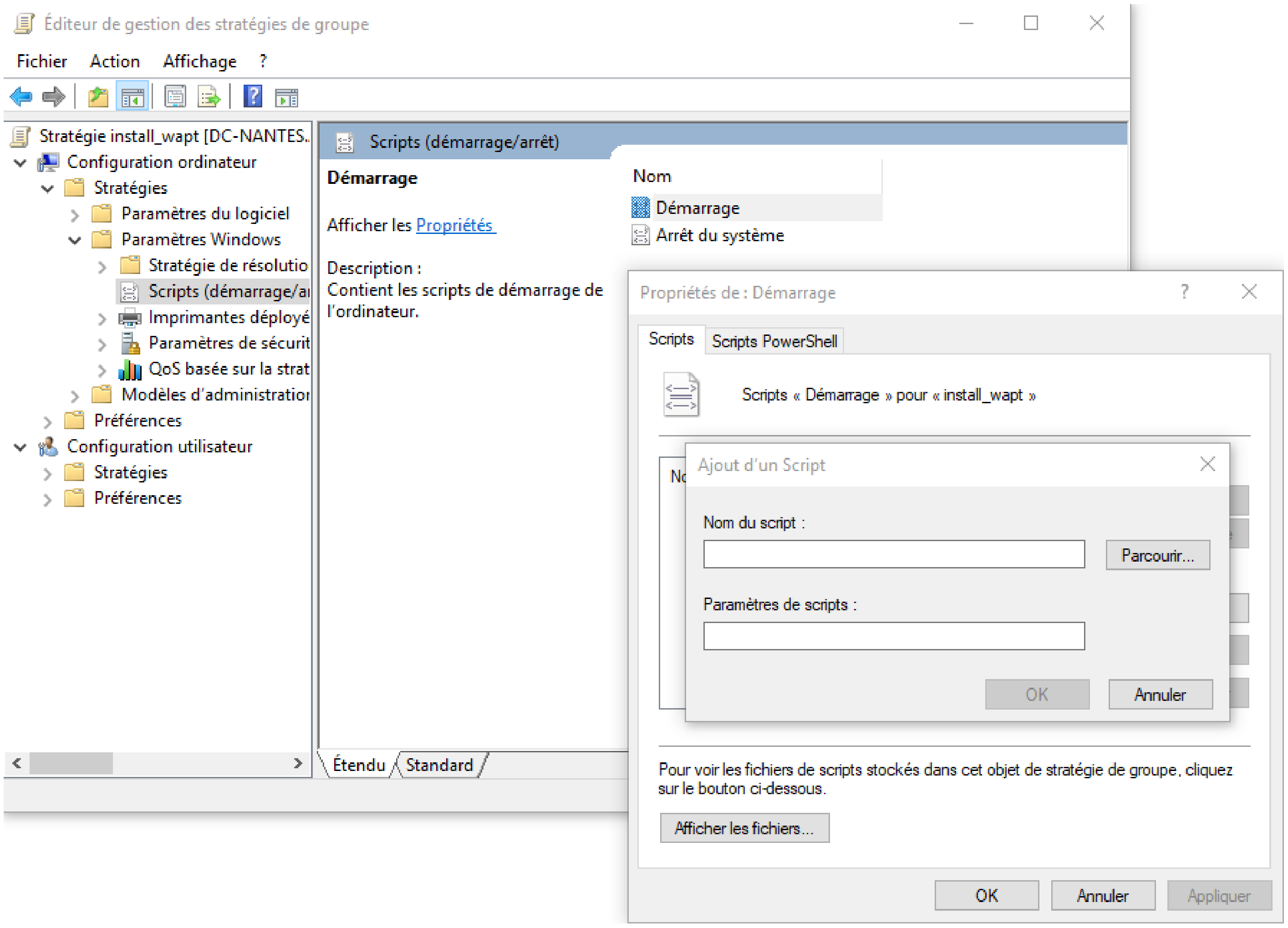


Fig. 61: Finding the waptdeploy.exe file on your computer

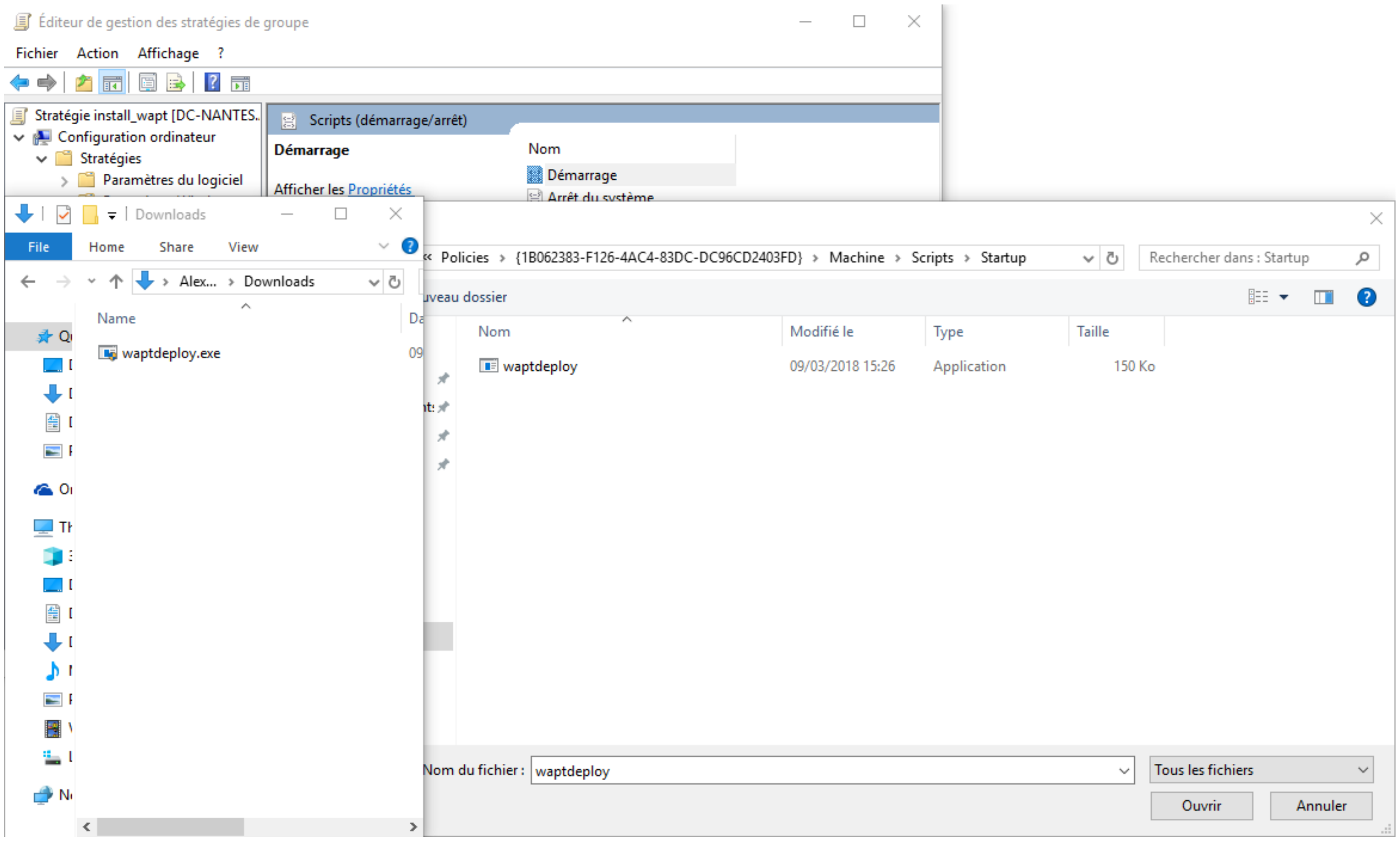
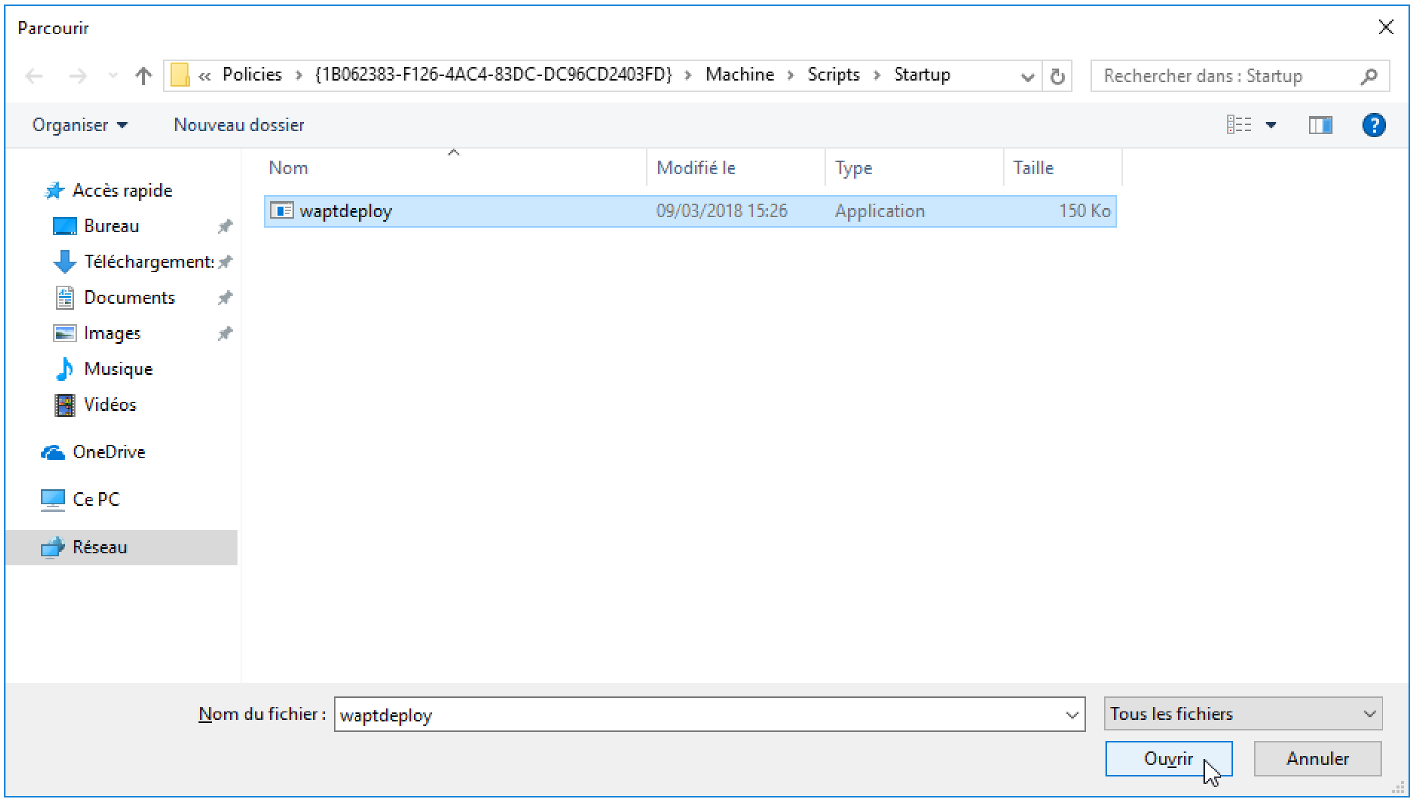

Éditeur de gestion des stratégies de groupe
Fichier Action Affichage ?
Stratégie install_wapt [DC-NANTES.
Configuration ordinateur
Stratégies
Paramètres du logiciel
Scripts (démarrage/arrêt)
Démarrage
Afficher les Propriétés
Nom
Démarrage
Arrêt du système
Downloads
File Home Share View
Alex... > Downloads
Name
waptdeploy.exe
Policies > {1B062383-F126-4AC4-83DC-DC96CD2403FD} > Machine > Scripts > Startup
Rechercher dans : Startup
Nom Modifié le Type Taille
waptdeploy 09/03/2018 15:26 Application 150 Ko
Nom du fichier : waptdeploy
Tous les fichiers
Ouvrir
Annuler

Fig. 62: Selecting the waptdeploy.exe script



Nom	Modifié le	Type	Taille
waptdeploy	09/03/2018 15:26	Application	150 Ko

Nom du fichier : waptdeploy

Tous les fichiers

Fig. 63: Selecting the waptdeploy.exe script

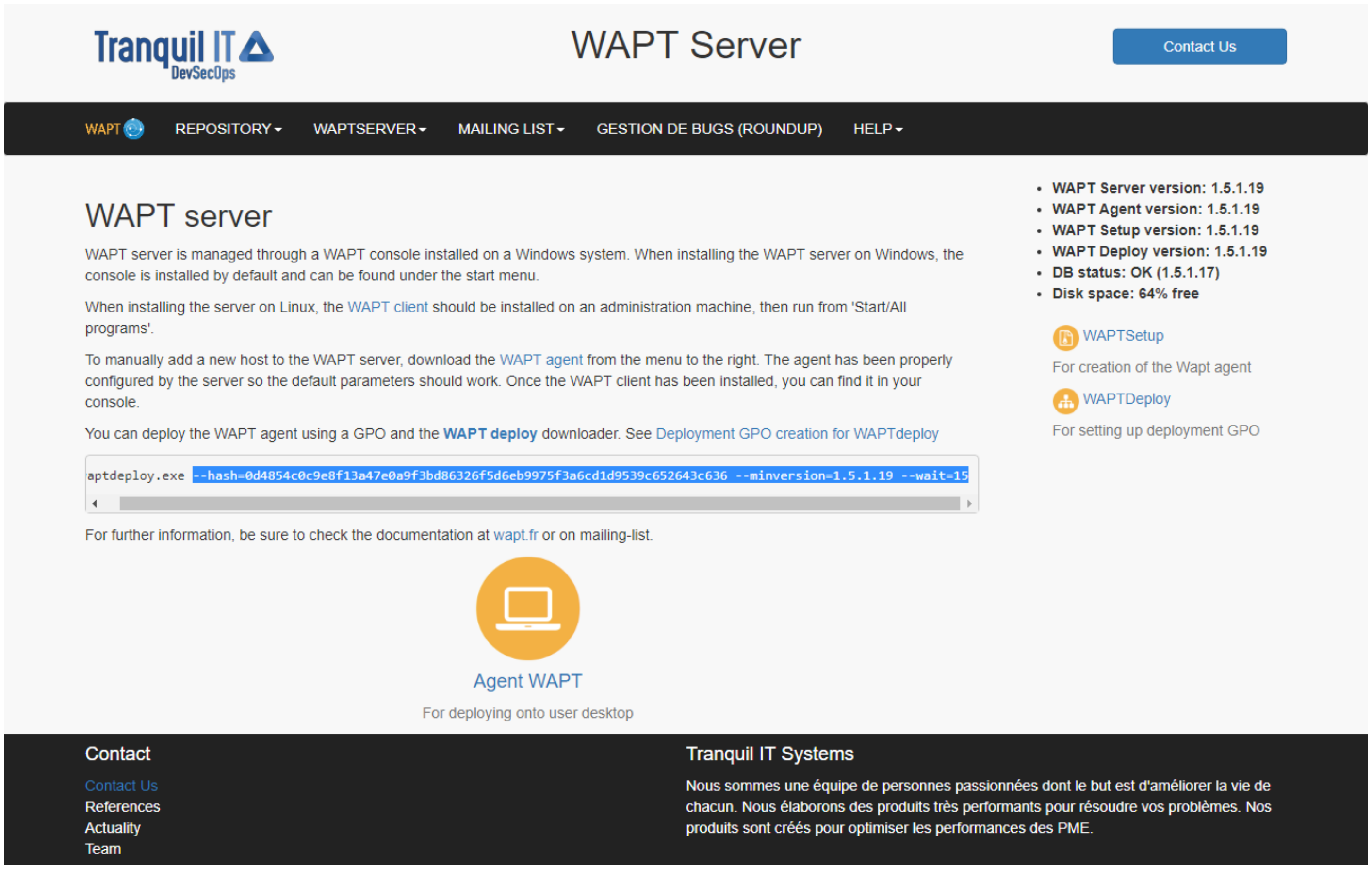
Passing arguments

Hint: Starting with version 1.3.7, it is necessary to provide the checksum of the `waptagent.exe` as an argument to the *waptdeploy* GPO.

This will prevent the remote machine from executing an erroneous/ corrupted **`waptagent`** binary.

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

Note: Parameters and **`waptagent.exe`** checksum to use for the *waptdeploy* GPO are available on the WAPT Server by visiting https://srvwapt.mydomain.lan.



WAPT server

WAPT server is managed through a WAPT console installed on a Windows system. When installing the WAPT server on Windows, the console is installed by default and can be found under the start menu.

When installing the server on Linux, the WAPT client should be installed on an administration machine, then run from 'Start/All programs'.

To manually add a new host to the WAPT server, download the WAPT agent from the menu to the right. The agent has been properly configured by the server so the default parameters should work. Once the WAPT client has been installed, you can find it in your console.

You can deploy the WAPT agent using a GPO and the **WAPT deploy** downloader. See Deployment GPO creation for WAPTdeploy

```
aptdeploy.exe --hash=0d4854c0c9e8f13a47e0a9f3bd86326f5d6eb9975f3a6cd1d9539c652643c636 --minversion=1.5.1.19 --wait=15
```

For further information, be sure to check the documentation at wapt.fr or on mailing-list.

Agent WAPT

For deploying onto user desktop

- **WAPT Server version: 1.5.1.19**
- **WAPT Agent version: 1.5.1.19**
- **WAPT Setup version: 1.5.1.19**
- **WAPT Deploy version: 1.5.1.19**
- **DB status: OK (1.5.1.17)**
- **Disk space: 64% free**

WAPTSetup

For creation of the Wapt agent

WAPTDeploy

For setting up deployment GPO

Contact

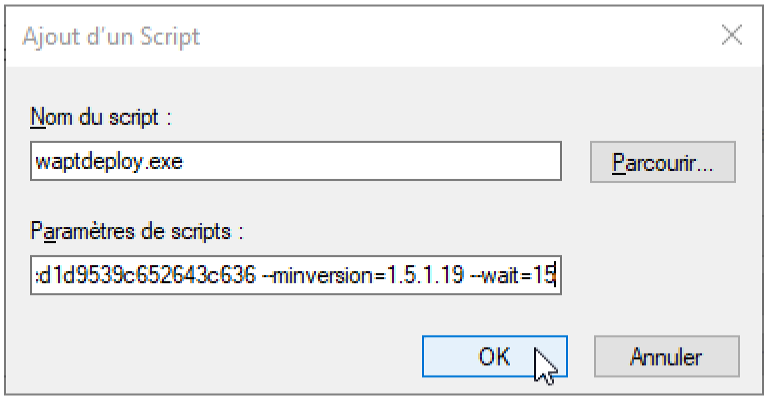
Contact Us
References
Actuality
Team

Tranquil IT Systems

Nous sommes une équipe de personnes passionnées dont le but est d'améliorer la vie de chacun. Nous élaborons des produits très performants pour résoudre vos problèmes. Nos produits sont créés pour optimiser les performances des PME.

Fig. 64: Web console of the WAPT Server

- copy the required parameters;
- click on *OK* to go on to the next step;
- click on *OK* to go on to the next step;



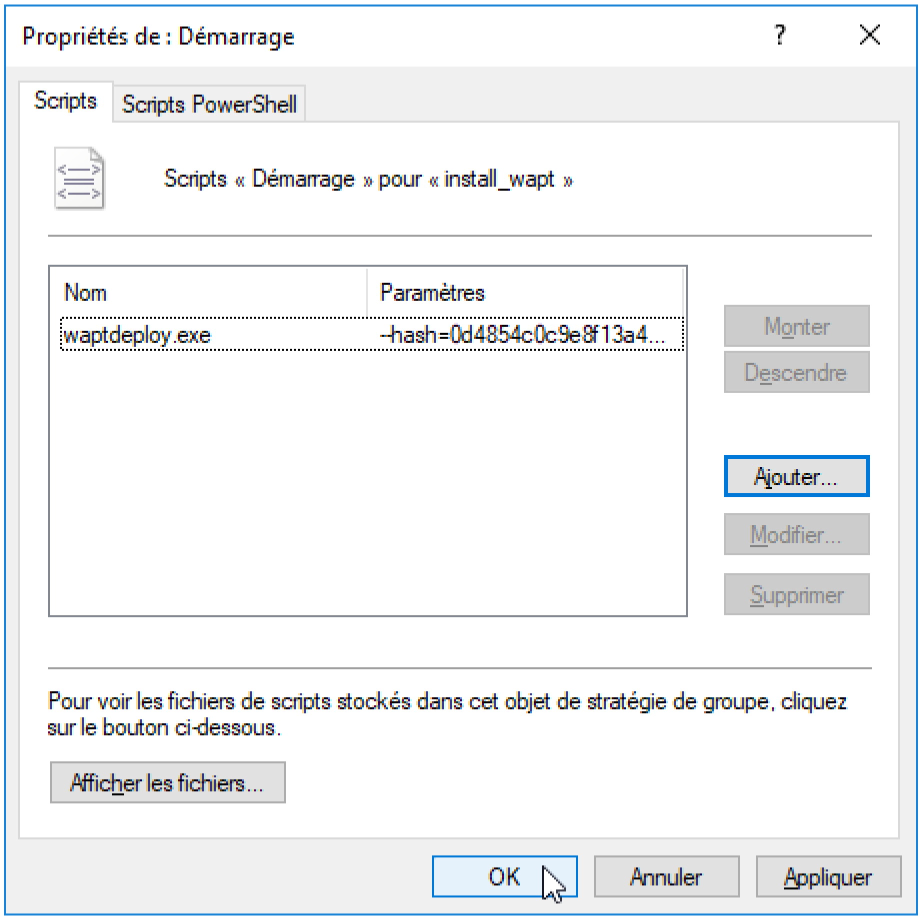
Nom du script :

waptdeploy.exe

Paramètres de scripts :

:d1d9539c652643c636 --minversion=1.5.1.19 --wait=15

Fig. 65: add the *waptdeploy* script to the startup GPO



Scripts « Démarrage » pour « install_wapt »

Nom	Paramètres
waptdeploy.exe	--hash=0d4854c0c9e8f13a4...

Pour voir les fichiers de scripts stockés dans cet objet de stratégie de groupe, cliquez sur le bouton ci-dessous.

Fig. 66: WAPTdeploy GPO to be deployed on next startup

- apply resulting GPO strategy to the Organization's Computers OU;

Additional arguments available for waptdeploy

Table 3: Additional arguments available for waptdeploy

Options	Value	Description
`--force`		Forces the installation of **`waptagent.exe`** even if the WAPT agent is already installed.
`--waptsetupurl`	https://srvwapt.mydomain.lan/wapt/waptagent.exe	Gives explicitly the WAPT agent URL/path to use to download the WAPT agent
`--tasks`	autorunTray,installService,installredist2008,autoUpgradePolicy	Sets **`waptagent`** installation tasks
`--wait`	10	Timeout for installing the WAPT agent.
`--setupargs=`	/dnsdomain=mydomain.lan /wapt_server= /repo_url=	Passing additional parameters to **`waptagent`**

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

Launching waptdeploy with a scheduled task

For **`waptdeploy`** to work best, you may execute the GPO upon computer shutdown;

You may also choose to launch **`waptdeploy`** using a scheduled task that has been set by GPO.

Hint: This method is particularly effective for deploying WAPT on workstations when the network is neither available on starting up or shutting down.

The method consists of using a GPO to copy `waptdeploy.exe` and `waptagent.exe`:

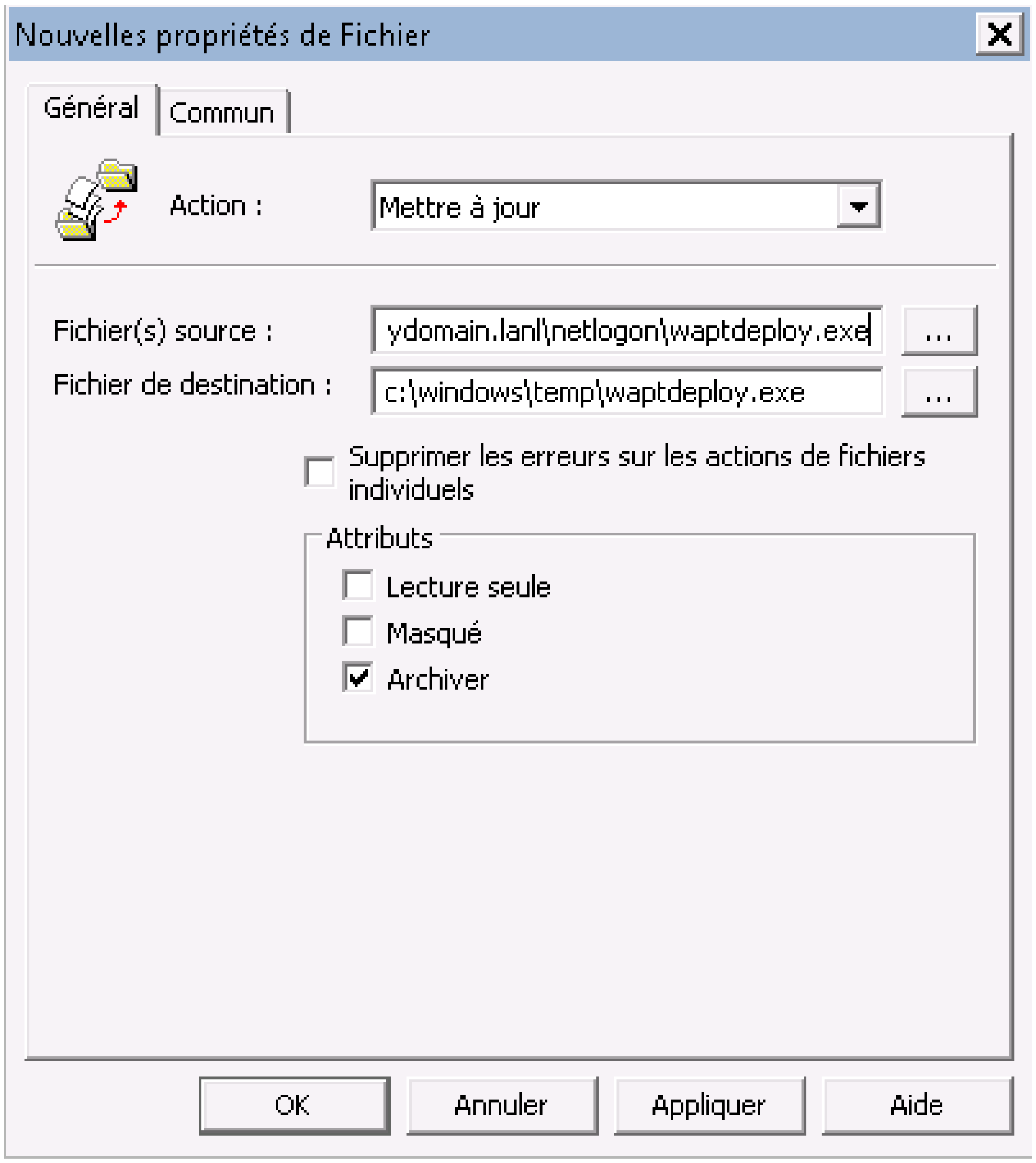
- Source : `\mydomain.lan\netlogon\waptagent.exe`
- Destination : `C:\windows\temp\waptagent.exe`
- copy `waptdeploy.exe` and `waptagent.exe` in the netlogon share of your Active Directory Server;
- then create a GPO to set up a scheduled task that will launch **`waptdeploy`**:

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

  Arguments:

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

Attention: The `hash` and `min_version` arguments will change in reality compared to the documentation as WAPT continues to improve.


Nouvelles propriétés de Fichier

Général | Commun

Action : Mettre à jour

Fichier(s) source : ydomain.lan\netlogon\waptdeploy.exe

Fichier de destination : c:\windows\temp\waptdeploy.exe

☐ Supprimer les erreurs sur les actions de fichiers individuels

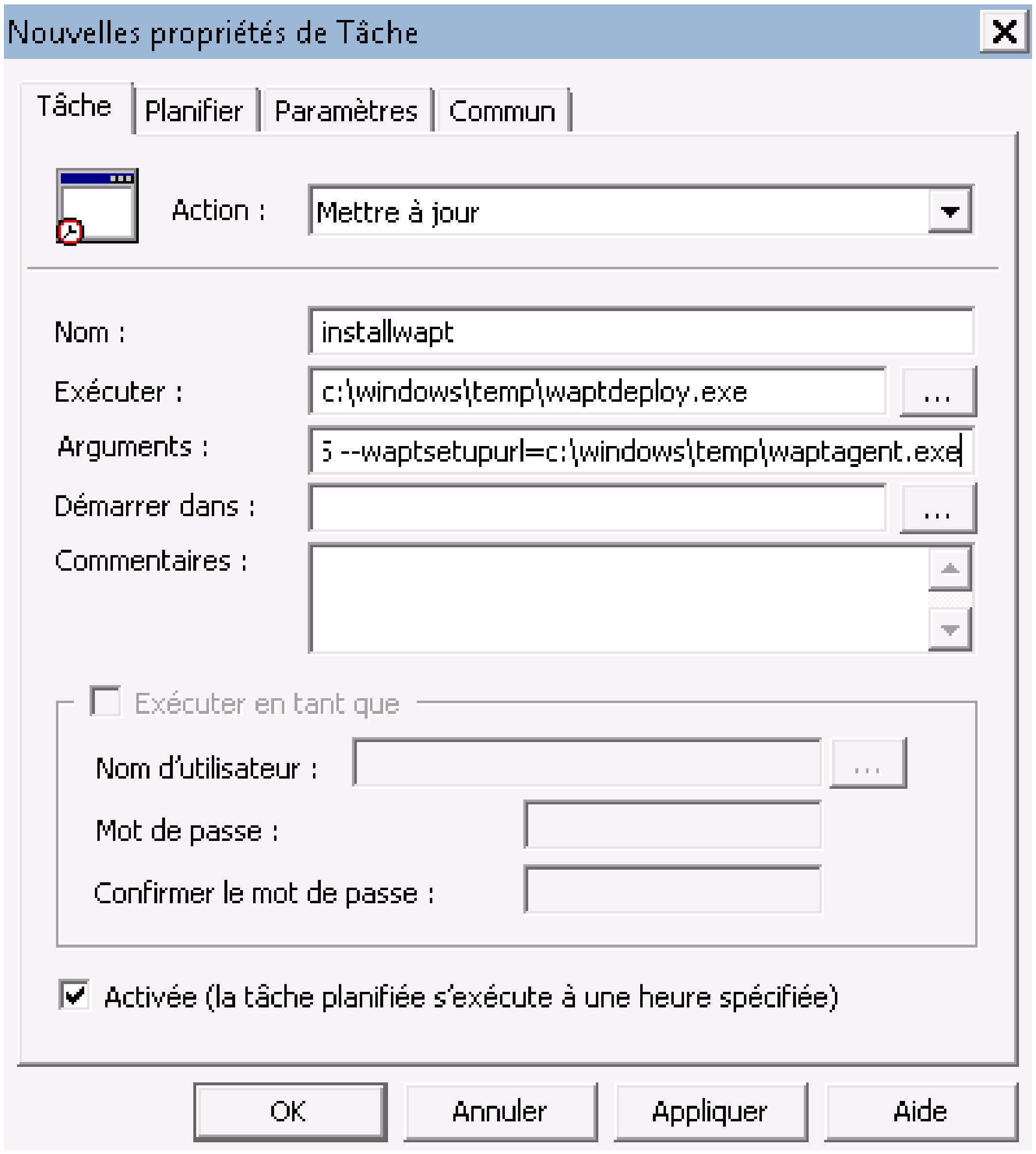
Attributs

☐ Lecture seule

☐ Masqué

☑ Archiver

Fig. 67: WAPT agent installation progress



Action : Mettre à jour

Nom : installwapt

Exécuter : c:\windows\temp\waptdeploy.exe

Arguments : 5 --waptsetupurl=c:\windows\temp\waptagent.exe

Démarrer dans :

Commentaires :

☐ Exécuter en tant que

Nom d'utilisateur :

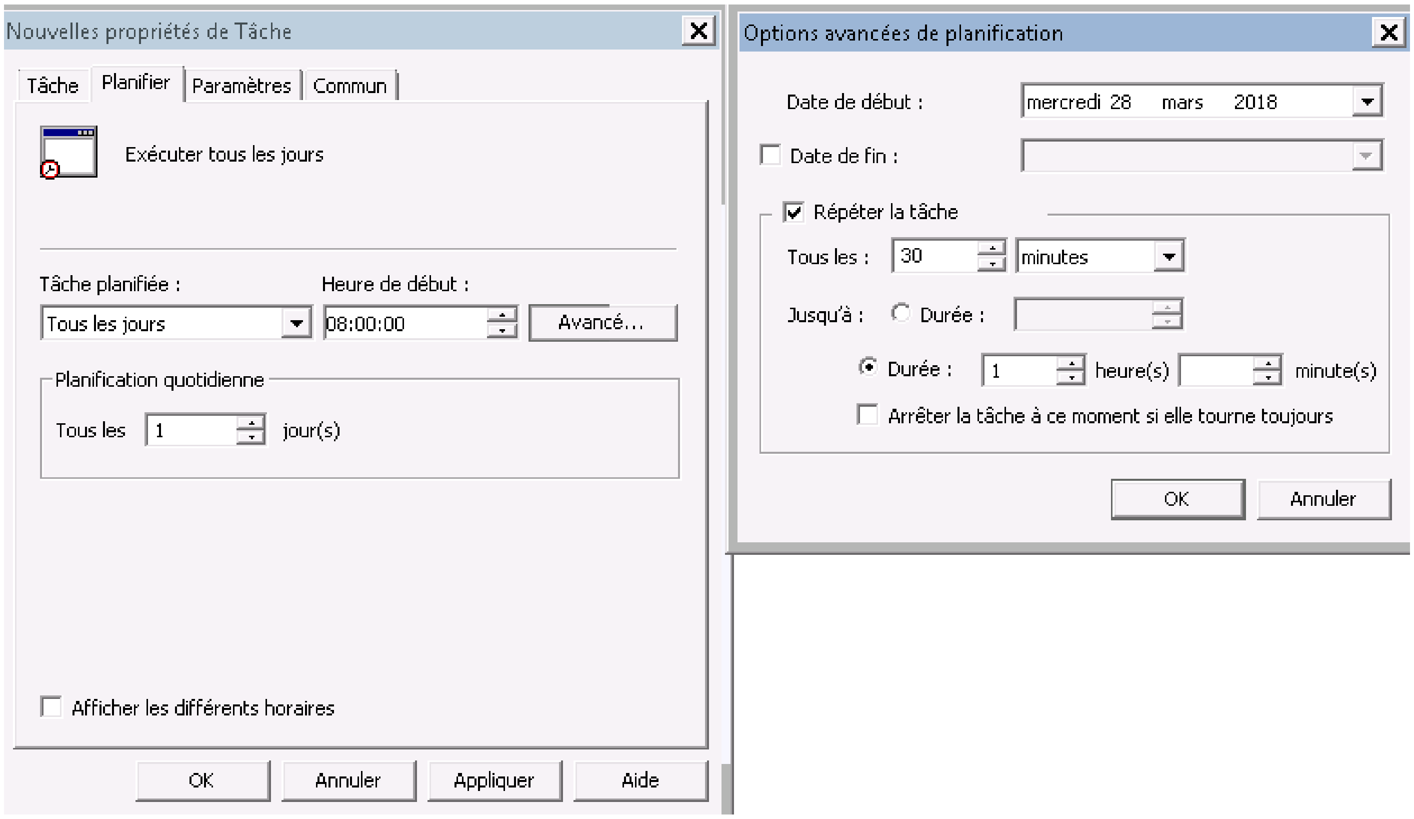
Mot de passe :

Confirmer le mot de passe :

☑ Activée (la tâche planifiée s'exécute à une heure spécifiée)

Fig. 68: Task installation properties

- choose a time after which the scheduled task will trigger and set the re-triggering of the task every 30 minutes until success:
- allow the scheduled task to start even if the device is powered on battery:


Nouvelles propriétés de Tâche

Exécuter tous les jours

Tâche planifiée : Tous les jours

Heure de début : 08:00:00

Planification quotidienne

Tous les 1 jour(s)

☐ Afficher les différents horaires

Options avancées de planification

Date de début : mercredi 28 mars 2018

☐ Date de fin :

☑ Répéter la tâche

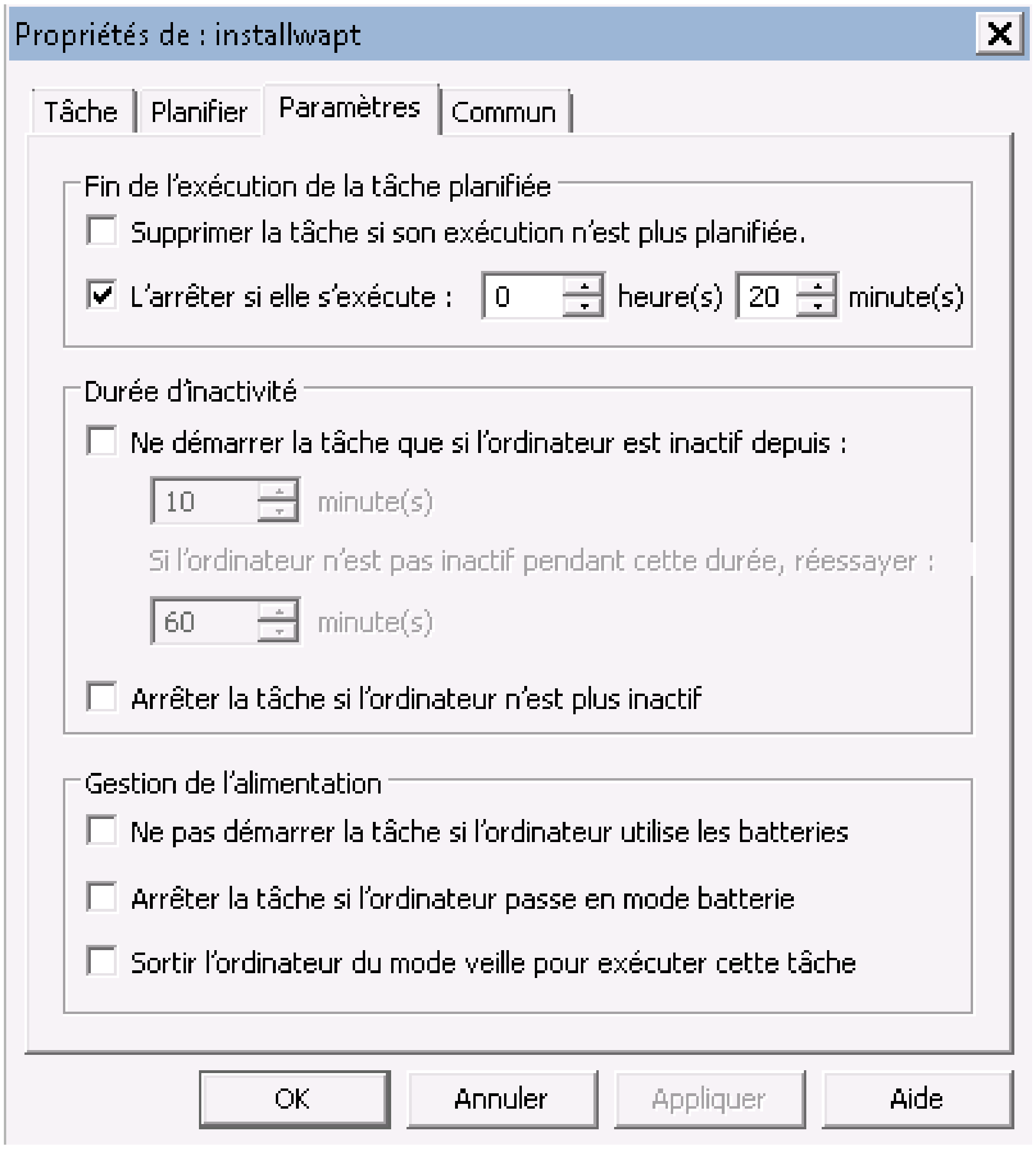
Tous les : 30 minutes

Jusqu'à : ○ Durée :

● Durée : 1 heure(s) minute(s)

☐ Arrêter la tâche à ce moment si elle tourne toujours

Fig. 69: Advanced properties of the installation task



Propriétés de : installwapt

Fin de l'exécution de la tâche planifiée

☐ Supprimer la tâche si son exécution n'est plus planifiée.

☑ L'arrêter si elle s'exécute : 0 heure(s) 20 minute(s)

Durée d'inactivité

☐ Ne démarrer la tâche que si l'ordinateur est inactif depuis :

10 minute(s)

Si l'ordinateur n'est pas inactif pendant cette durée, réessayer :

60 minute(s)

☐ Arrêter la tâche si l'ordinateur n'est plus inactif

Gestion de l'alimentation

☐ Ne pas démarrer la tâche si l'ordinateur utilise les batteries

☐ Arrêter la tâche si l'ordinateur passe en mode batterie

☐ Sortir l'ordinateur du mode veille pour exécuter cette tâche

Fig. 70: Power settings

6.6.6 Deploying the WAPT Agent on Linux

New in version 1.8.

Starting with WAPT 1.8, a Linux agent is available for / and .

Note:

- the following procedure installs a WAPT agent using Tranquil IT's repositories for Debian/CentOS;
- if you wish to install it manually, you can look for your corresponding version;
- copy the link of the binary that you need, download and install it with dpkg / rpm;

Installing the WAPT agent on Debian

The most secure and reliable way to install the latest WAPT agent on Linux Debian is using Tranquil IT's public repository.

- add Tranquil IT's repository in apt repository lists:

Important: Follow this procedure for getting the right packages for the WAPT Enterprise Edition. For WAPT Community Edition please refer to the next block.

To access WAPT Enterprise ressources, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **deb** parameter to access WAPT Enterprise repository.

```
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: Follow this procedure for getting the right packages for the WAPT Community Edition. For WAPT Enterprise Edition please refer to the previous block.

```
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
```

- install WAPT agent using apt-get:

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

Installing the WAPT agent on CentOS

The most secure and reliable way to install the latest WAPT agent on Linux CentOS is using Tranquil IT's public repository.

- add Tranquil IT's repository in yum repository lists:

Important: Follow this procedure for getting the right packages for the WAPT Enterprise Edition. For WAPT Community Edition please refer to the next block.

To access WAPT Enterprise ressources, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **baseurl** parameter to access WAPT Enterprise repository.

```
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: Follow this procedure for getting the right packages for the WAPT Community Edition. For WAPT Enterprise Edition please refer to the previous block.

```
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
```

- install WAPT agent using yum:

```
yum install tis-waptagent
```

Creating the agent configuration file

The requisites for your WAPT agent to work are:

- `wapt-get.ini` config file in `/opt/wapt/`;
- a public certificate of the package-signing authority in `/opt/wapt/ssl/`;

You need to create and configure the `wapt-get.ini` file in `/opt/wapt` (*Configuring the WAPT agent*).

An example of what it should look like is present further down on this page. You may use it after changing the parameters to suit your needs.

```
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
```

Copying the package-signing certificate

You need to copy manually, or by script, the public certificate of your package signing certificate authority.

The certificate should be located on your Windows machine in `C:\Program Files (x86)\wapt\ssl\`.

Copy your certificate(s) in `/opt/wapt/ssl` using **WinSCP** or **rsync**.

Copying the SSL/TLS certificate

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

The certificate should be located on your Windows machine in `C:\Program Files (x86)\wapt\ssl\server\`.

Copy your certificate(s) in `/opt/wapt/ssl/server/` using **WinSCP** or **rsync**.

Then, modify in your config file the path to your certificate.

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

And give absolute path of your cert.

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

Attention: If you are not using SSL/TLS certificates with your WAPT Server, you must change it in `/opt/wapt/wapt-get.ini` the following lines to 0:

```
verify_cert=0
```

Registering your Linux agent

Attention:

- beware, by default, WAPT takes the system language by default for packages, you may have to define the language in `wapt-get.ini` with `locales=`.

- restart the WAPT service:

  ```
  systemctl restart waptservice.service
  ```

- finally, execute the following command to register your Linux host with the WAPT server:

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

Congratulations, your Linux Agent is now installed and configured and it will now appear in your WAPT Console with a icon!!

Supported features

Most features are now supported in version 1.8.2 of WAPT.

Unsupported features

- installing updates on shutdown

;
- WAPT console is not currently available on linux

;
- any Windows specific feature;

Particularities with domain functionality

- testing was carried out with sssd with an Active Directory domain and kerberos authentication;
- to integrate a machine in the Active Directory domain, you can choose to follow this documentation
- to force the update of Organisational Units on the host, you can apply a **`gpupdate`** from the WAPT console;
- in order for Active Directory groups to function properly, you must verify that the **`id hostname$`** command returns the list of groups the host is member of;

Attention: We have noticed that the Kerberos LDAP query does not work if the reverse DNS record is not configured correctly for your domain controllers. These records must therefore be created if they do not exist.

6.6.7 Deploying the WAPT agent on MacOS

New in version 1.8.

Attention: Currently, the agent has only been tested on High Sierra (version 10.13) and Mojave (10.14) while the latest MacOS version is Catalina (10.15). Catalina may have introduced changes that could prevent the agent from working.

Installing the WAPT Agent package from Tranquil IT's public repository

- 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
```

- install the downloaded package:

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

Creating the agents configuration file

The requisites for your WAPT agent to work are:

- `wapt-get.ini` config file in `/opt/wapt/`;
- a public certificate of the package-signing authority in `/opt/wapt/ssl/`;

You need to create and configure the `wapt-get.ini` file in `/opt/wapt` (*Configuring the WAPT agent*).

An example of what it should look like is present further down on this page. You may use it after changing the parameters to suit your needs.

```
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
```

Copying the package-signing certificate

You need to copy manually, or by script, the public certificate of your package signing certificate authority.

The certificate should be located on your Windows machine in `C:\Program Files (x86)\wapt\ssl\`.

Copy your certificate(s) in `/opt/wapt/ssl` using **WinSCP** or **rsync**.

Copying the SSL/TLS certificate

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.

The certificate should be located on your Windows machine in `C:\Program Files (x86)\wapt\ssl\server\`.

Copy your certificate(s) in `/opt/wapt/ssl/server/` using **`WinSCP`** or **`rsync`**.

Then, modify in your `wapt-get.ini` config file the path to your certificate.

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

And give absolute path of your cert.

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

Attention: If you are not using SSL/TLS certificates with your WAPT Server, you must set the following lines to 0 in `/opt/wapt/wapt-get.ini`:

```
verify_cert=0
```

Registering your MacOS agent

Attention:

- beware, by default, WAPT takes the system language by default for packages, you may have to define the language in `wapt-get.ini` with `locales=`.

- restart the WAPT service:

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

- finally, execute the following command to register your MacOS host with the WAPT server:
- you must logon as root to run :

```
wapt-get register
```

- then switch back to normal user for the following :

```
sudo wapt-get update
```

👏 **Congratulations**, your MacOS Agent is now installed and configured and it will now appear in your WAPT Console with a icon!

Supported features

Most features are now supported in version 1.8.2 of WAPT.

Unsupported features

- installing updates on shutdown ;
- WAPT console is not currently available on linux ;
- any Windows specific feature;

Particularities with domain functionality

- testing was carried out with sssd with an Active Directory domain and kerberos authentication;
- to integrate a machine in the Active Directory domain, you can choose to follow this documentation
- to force the update of Organisational Units on the host, you can apply a **`gpupdate`** from the WAPT console;
- in order for Active Directory groups to function properly, you must verify that the **`id hostname$`** command returns the list of groups the host is member of;

Attention: We have noticed that the Kerberos LDAP query does not work if the reverse DNS record is not configured correctly for your domain controllers. These records must therefore be created if they do not exist.

6.6.8 Deploying the Linux WAPT Agent with Ansible

To avoid mistakes and automate your WAPT agents deployment on Linux, we provide Ansible roles for installing WAPT agents on:

-
-

You can explore the role source code by visiting this link on Github.

Requirements

- Debian Linux or CentOS hosts;
- a sudoers user on these hosts;
- Ansible 2.8;

Installing the Ansible role

- install `tranquilit.waptagent` Ansible role;

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

- to install the role elsewhere, use the *-p* subcommand like this;

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

Using the Ansible role

- ensure you have a working ssh key deployed on your hosts, if not you can generate and copy one like below;

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

- edit Ansible hosts inventory (`./hosts`) and add the Linux hosts;

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

- create a playbook with the following content in `./playbooks/deploywaptagent.yml`;

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

- ensure all variables are correctly set (see *wapt-get.ini variables*);
  - `wapt_server_url`;
  - `wapt_repo_url`;
  - `wapt_crt`;

Important: Variables configuration is important as it will configure the behavior of the WAPT.

You **must** replace the default certificate with your Code-Signing public certificate.

- run your playbook with the following command;

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

Congratulations, you have installed your WAPT agent on your Linux hosts!

Role variables

Available variables are listed below, along with default values (see `defaults/main.yml`).

WAPT agent variables

- version of WAPT that will be installed from WAPT Deb/RPM repository;

```
wapt_version: "1.8"
```

- version of CentOS used for RPM repository address;

```
centos_version: "centos7"
```

wapt-get.ini variables

The `wapt_server_url` parameter points to your WAPT server and is used by default for the `wapt_repo_url`.

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

You can override it like so:

```
wapt_server_url: "https://wapt.landomain.lan"
wapt_repo_url: "https://wapt.otherdomain.com/wapt/"
```

Certificate filename located in `files/` subdirectory of the role:

```
wapt_crt: "wapt_ca.crt"
```

Example Ansible playbook

Here is an example of an Ansible playbook.

```yaml
- hosts: hosts
  vars_files:
    - vars/main.yml
  roles:
    - tranquilit.waptagent
```

To go further, more configuration options are available in this part of the documentation:

6.6.9 Configuring the WAPT agent

The configuration file `C:\Program Files(x86)\wapt\wapt-get.ini` defines the behavior of the WAPT agent.

The `[global]` section is required:

```ini
[global]
```

Description of available options for the WAPT agent

Note:

- if `repo_url` and `wapt_server` fields are empty, the WAPT agent will look for a repository using SRV records in the `dnsdomain` zone;
- if there is no `wapt_server` attribute in the `[global]` section, no WAPT Server will be used;
- if there is no `repo_url` attribute in the `[global]` section, a repository in the `[wapt]` section will have to be explicitly defined;
- it will have to be enabled by adding it to the `repositories` attribute to the `[global]` section;

Table 4: Description of available options for the WAPT agent

Options	Description
`use_hostpackages` = 1	Use host packages (default 1).
`waptupdate_task_period` = 120	Update frequency (120 minutes by default).
`waptupgrade_task_period` = 360	Upgrade frequency (disabled by default)
`waptservice_port` = 8088	WAPT agent loopback port. It is not accessible from the network.
`dbpath = C:\Program Files(x86)\wapt\db\waptdb.sqlite`	Path to the local database file.
`loglevel` = warning	Log level of the WAPT agent. Possible values are: `debug`, `info`, `warning`, `critical`.
`maturities` = PROD	List of packages maturities than can be viewed and installed by WAPT Agent. Default value is `PROD`. Any value can be used.
`use_fqdn_as_uuid` = 1	Allows you to use the fqdn name rather than the uuid BIOS as the unique machine identifier in wapt.
`waptaudit_task_period` = 120	Define the frequency where the agent checks if he has audits to perform.
`locales` = en	Allows you to set the list of wapt agent languages to modify the list of packages visible by wapt (for package filtering). You can add multiple language (eg. `locales=fr,en`) in order of preference.
`host_profiles` = tis-firefox,tis-java	Allows you to define a wapt package list that the wapt agent must install.
`language` = en	Force default langauge for GUI (not for package filtering)
`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	Define whether a download of pending packages should be started after an update with waptupdate_task_period
`log_to_windows_events` = False	Send the log wapt in the window events
`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.

WAPT Server configuration attributes

These options will set WAPT agent behavior when connecting to WAPT Server.

Table 5: Description of available options for the WAPT Server

Options	Description
`wapt_server` =	WAPT Server URL. If the attribute is not present, no WAPT Server will be contacted. If the attribute is empty, a DNS query will be triggered to find the WAPT Server using the `dnsdomain` attribute for the DNS zone.
`dnsdomain` =	DNS zone on which the DNS SRV `_waptserver._tcp` is searched.
`wapt_server_timeout` = 10	WAPT Server HTTPS connection timeout in seconds
`use_kerberos` = 1	Use Kerberos authentication for initial registration on the WAPT Server.
`verify_cert` = C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt	See the documentation on activating the *verification of HTTPS certificates*
`public_certs_dir = C:\Program Files (x86)\wapt\ssl`	Folder of certificates authorized to verify the signature of WAPT packages, by default, `<wapt_base_dir>\\ssl`. Only files in this directory with `.crt` or `.pem` extension are taken into account. There may be several X509 certificates in each file. Authorized packages in WAPT are those whose signature may be verified by one of the certificates contained in the PEM files of this directory. Each repository may have its own folder of authorized certificates.

Using several repositories

There can be more sections in the `wapt-get.ini` file to define more repositories:

- `[wapt]`: main repository. Relevent attributes: `repo_url`, `verify_cert`, `dnsdomain`, `http_proxy`, `use_http_proxy_for_repo`, `timeout`. If this section does not exist, parameters are read from the `[global]` section;
- `[wapt-template]`: external remote repository that will be used in the WAPT console for importing new or updated packages;
- `[wapt-host]`: repository for host packages. If this section does not exist, default locations will be used on the main repository;

More information on that usage can be found in *this article on working with multiple public or private repositories*.

Note: Active repositories are listed in the `repositories` attribute of the `[global]` section.

Table 6: Description of available options for repositories

Options	Description
`repositories` = repo1, repo2	List of enabled repositories, separated by a comma. Each value defines a section of the `wapt-get.ini` file. In each section, it is possible to define `repo_url`, `dnsdomain`, `public_certs_dir`, `http_proxy`.

Note: This parameter can be configured both in the WAPT agent configuration and in the WAPT console configuration file `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

For information on configuring the WAPT console, please refer to *this documentation*.

Settings for waptexit

Table 7: Description of available options for WAPTexit

Options	Description
`allow_cancel_upgrade` = 1	Prevents users from canceling package upgrades on computer shutdown.
`pre_shutdown_timeout` = 180	Timeout for scripts at computer shutdown.
`max_gpo_script_wait` = 180	Timeout for GPO execution at computer shutdown.
`hiberboot_enabled` = 0	Disables Hiberboot on Windows 10 to make **`waptexit`** work correctly.

Settings for WAPT Self-Service and Waptservice Authentification

Table 8: Description of available options for the WAPT Self-Service and Waptservice Authentification

Options	Description
`waptservice_admin_filter` = True	Apply *selfservice package* view filtering for Local Administrators.
`service_auth_type` = system	Defines the authentication system of the wapt service, available value are *system*, *waptserver-ldap*, *waptagent-ldap*.
`ldap_auth_ssl_enabled` = False	Useful with *waptagent-ldap*, defines if the LDAP request must be encrypted.
`verify_cert_ldap` = True	Useful with *waptagent-ldap*, define whether the certificate should be verified.
`ldap_auth_base_dn` = dc=domain,dc=lan	Useful with *waptagent-ldap*, defines the base dn for the LDAP request.
`ldap_auth_server` = srvads.domain.lan	Useful with *waptagent-ldap*, defines the LDAP server to contact.
`waptservice_user` = admin	Forces a user to authenticate on the WAPT service.
`waptservice_password` = 5e884898da	sha256 hashed password when *waptservice_user* is used (the value *NOPASSWORD* disables the requirement for a password).

Settings for wapttray

Table 9: Description of available options for the WAPT tray

Options	Description
`notify_user` = 0	Prevents `wapttray` from sending notifications (popup).

Proxy settings

Table 10: Description of available options for the WAPT Server

Options	Description
`http_proxy` = http://user:pwd@host_fqdn:port	HTTP proxy address
`use_http_proxy_for_repo` = 0	Use the proxy to access the repositories.
`use_http_proxy_for_server` = 0	Use a proxy to access the WAPT Server.
`use_http_proxy_for_templates` = 0	Use a proxy to access package template server.

Settings for creating packages

Table 11: Description of available options for creating WAPT packages

Options	Description
personal_certificate_path = C:\private\org-coder.crt	Path to the Administrator's private key.
default_sources_root = C:\waptdev	Directory for storing packages in development.
default_sources_root_host = C:\waptdev\hosts	Directory for storing host packages in development.
default_package_prefix = tis	Default prefix for new or imported packages.
default_sources_suffix = wapt	Default prefix for new or imported packages.

Settings for `WAPT Windows Updates`

Refer to *this article on configuring WAPTWUA on the WAPT agent*.

Overriding settings of *upload* functions

It's possible to override **upload** commands to define a particular behavior when uploading packages. It's possible for example to upload packages on several repositories, or via another protocol, etc.

To upload packages on the repository (**wapt-get upload-package** or **build-upload**), use:

```
upload_cmd="C:\\Program Files (x86)\\WinSCP\\WinSCP.exe" admin@srvwapt.mydomain.lan /upload
↪%(waptfile)s
```

To upload host-packages on the repository (**upload-package** or **build-upload** of a host package), use:

```
upload_cmd_host="C:\\Program Files (x86)"\\putty\\pscp -v -l admin %(waptfile)s srvwapt.mydomain.
↪lan:/var/www/wapt-host/
```

To launch a command after a package **upload**, use:

```
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/"
```

Configuration of WAPT agents

After standard installation, the default configuration is:

```
[global]
waptupdate_task_period=120
waptserver=https://srvwapt.mydomain.lan
repo_url=https://srvwapt.mydomain.lan/wapt/
use_hostpackages=1
```

Making changes in `wapt-get.ini` and regenerating an agent is not sufficient to push the new configuration.

You can create a WAPT package to push updated `wapt-get.ini` settings.

The package is available from the Tranquil IT repository: 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 Configuring the WAPT console

Hint: The configuration file for the WAPT console is stored in `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`. This file is automatically generated when the **waptconsole** is first launched and it is generated from the `wapt-get.ini` file configured on the *Administrator*'s workstation.

Several options are available in the `[global]` section of the `waptconsole.ini` file:

Table 12: Description of available options for the WAPT console

Options	Description
`wapt_server` = https://srvwapt.mydomain.lan	Address of the WAPT Server.
`repo_url` = https://srvwapt.mydomain.lan/wapt	Address of the main WAPT repository.
`last_usage_report` = 03/01/2017 18:45:51	Date when the WAPT console was last used.
`http_proxy` =	Address of the proxy server in the console.
`use_http_proxy_for_server` = 0	Use a proxy to connect to the WAPT Server from the console.
`use_http_proxy_for_repo` = 0	Use a proxy to connect to the main WAPT repository from the console.
`default_package_prefix` = tis	Prefix used for naming WAPT packages.
`default_sources_root` = C:\waptdev	WAPT base package development folder.
`personal_certificate_path` = C:\private\mykey.crt	Path to the certificate associated with the *Administrator*'s private key.
`send_usage_report` = 1	Allows the WAPT console to send anonymous statistics to Tranquil IT.
`language` = en	Language of the WAPT console.
`advanced_mode` = 0	Launches the console in debug mode.
`verify_cert` = C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt	For *verifying HTTPS certificates*.
`waptservice_timeout` = 2	Timeout for actions applied to WAPT agents (ex: **`update`**).
`enable_external_tools` = 0	Displays the actions that call external applications (RDP, VNC, etc...).
`enable_management_features` = 0	Displays the button to create self-signed certificates or to create the WAPT agent's installer.
`hide_unavailable` = 0	Hides actions that are not available for the WAPT agent.
`check_certificates_validity` = 1	Forces the package certificate's date and CRL to be verified.
`sign_digests` = `sha256,sha1`	List of allowed signature algorithms for the WAPT packages.

Configuring external repositories

You may add several external repositories by adding `[sections]` in `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

Example:

```
[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
```

Table 13: Description of available options for external repositories

Options	Description
`repo_url` = http://srvwapt.mydomain.lan/wapt	Address of the external WAPT repository.
`http_proxy` = http://proxy.mydomain.lan:8080	Address of the proxy to use to access the external repository referenced in the `[section]`.
`verify_cert` = 1	For *verifying HTTPS certificates*.
`public_certs_dir` =	Folder that contains the certificates used to authenticate downloaded external packages.
`timeout` = 2	Timeout for the external repository referenced in the `[section]`. If left empty, no verification is performed.

Settings for creating WAPT packages

Table 14: Description of available options for creating WAPT packages

Options	Description
`personal_certificate_path` = C:\private\coder.crt	Path to the private key to be used to sign packages.
`default_sources_root` = C:\waptdev	WAPT base package development folder.
`default_sources_root_host` = C:\waptdev\hosts	WAPT host package development folder.
`default_package_prefix` = tis	Default prefix for new WAPT packages.
`default_sources_suffix` = wapt	Default suffix for new WAPT packages.

6.6.11 Configuring the WAPT Server

The WAPT Server configuration file on GNU/ Linux systems is found in `/opt/wapt/conf/waptserver.ini`.

The WAPT Server configuration file on Windows systems is found in `C:\wapt\conf\waptserver.ini`.

Attention: Modification of these files is reserved for advanced users!!

Section [option]

Several options can be defined in the section:

```
[options]
```

Table 15: Available parameters for the [option] section of waptserver.ini

Options	Description
`allow_unauthenticated_connect` = False	Defines whether websocket connections should be authenticated
`allow_unauthenticated_registration` = True	Allows the initial registration of the WAPT agent using a login and password
`allow_unsigned_status_data` = False	Debug only - Allows unsigned status data from agent

continues on next page

Table 15 – continued from previous page

Options	Description
`application_root` = ‘’	Set custom WAPT server application root path (ex: wapt)
`auto_create_ldap_users` = True	Related to user ACLs
`client_certificate_lifetime` = 3650	Host certificate lifetime
`clients_read_timeout` = 5	Websocket client timeout
`clients_signing_certificate` =	Host certificates signing cert
`clients_signing_crl_days` =	Host certificates signing CRL day
`clients_signing_crl` =	Host certificates signing CRL
`clients_signing_crl_url` =	Host certificates signing CRL URL
`clients_signing_key` =	Host certificates signing key
`client_tasks_timeout` = 1	Maximum allowed delay before WAPT agent requests timeout
`db_connect_timeout` = 10	Maximum allowed delay before PostgreSQL queries timeout
`db_host` =	Address of the PostgreSQL server (empty by default, it will use a local Unix Socket).
`db_max_connections` = 100	Maximum simultaneous connections to the PostgreSQL database
`db_name` = wapt	Name of the PostgreSQL database that the WAPT Server will connect to.
`db_password` =	Password for authenticating the user on the PostgreSQL database (default: empty, it will use a local UNIX socket)
`db_port` = 5432	Port of the PostgreSQL server
`db_stale_timeout` = 300	Database stale timeout, default to 300 seconds
`db_user` =	Name of the PostgreSQL user connecting to the database (default: empty, it will use a local UNIX socket).
`enable_store` = False	Enables WAPT Store Webui (WAPT Enterprise only)
`encrypt_host_packages` = False	Encrypt host package with client certificate
`htpasswd_path` = None	Adds basic authentication to WAPT Server
`http_proxy` = http://srvproxy.mydomain.lan:3128	Defines the proxy server to allow the WAPT server to recover its CRL (Certificate Revocation List)
`known_certificates_folder` = /opt/wapt/ssl/	Adds additional knowed CA for certificate validation
`ldap_auth_base_dn` = None	Defines LDAP authentication base DN
`ldap_auth_server` = None	Defines LDAP authentication server
`ldap_auth_ssl_enabled` = True	Sets SSL auth on LDAP connections
`loglevel` = debug	Debug level. default level is warning
`max_clients` = 4096	Sets maximum simultaneous WAPT clients connection
`min_password_length` = 10	Sets minimum admin password length
`nginx_http` = 80	Defines Nginx http port (Windows only)
`nginx_https` = 443	Defines Nginx https port (Windows only)
`remote_repo_diff` = False	Enable remote repositories diff
`remote_repo_support` = True	Enables remote repositories functionnality on WAPT Server
`remote_repo_websockets` = True	Enables websocket communication with remote repositories agents
`secret_key` = FKjfzjfkF687fjrkeznfkj7678jknk78687	Random string for initializing the Python Flask application server. It is generated when first installing the WAPT Server and is unique for every WAPT Server.
`server_uuid` = 76efezfa6-b309-1fez5-92cd-8ea48fc122dc	WAPT Server *UUID* (this anonymous id is used for WAPT statistics).
`signature_clockskew` = 72000	Maximum allowed time difference for the websockets

continues on next page

Table 15 – continued from previous page

Options	Description
`token_lifetime` = 43200	Authentication token lifetime
`trusted_signers_certificates_folder` = None	Path to trusted signers certificate directory
`trusted_users_certificates_folder` = None	Path to trusted users CA certificate directory
`use_kerberos` = True	Requires a Kerberos authentication when first registering the WAPT agent.
`use_ssl_client_auth` = False	Enables client certification authentication
`wapt_admin_group_dn` = CN=waptadmins,OU=groups,DC=ad,DC=mydomain,DC=lan	LDAP DN of Active Directory User Group allowed to connect to WAPT console
`wapt_admin_group` = None	CN of Active Directory User Group allowed to connect to WAPT console
`wapt_folder` = /var/www/wapt	Directory of the WAPT repository.
`wapt_huey_db = C:\Program Files(x86)\wapt\db\waptservertasks.sqlite`	Path to database that handles tasks
`wapt_password` = 46642dd2b1dfezfezgfezgadf0ezgeezgezf53d	*SuperAdmin* password for connecting to the WAPT console.
`waptserver_port` = 8080	Specify WAPT Server python service port, default to `8080`
`wapt_user` = admin	Defines the *SuperAdmin* username in the WAPT console.
`waptwua_folder` = /var/www/waptwua	Location of WAPT WUA folder
`wol_port` = 9,123,4000	List of WakeOnLAN UDP ports to send magic packets to
`wapt_bind_interface` = 127.0.0.1	Define how to listen to the waptserver service

Configuring Nginx

The default Nginx configuration is as follows:

```
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";
    }

  location / {
    proxy_set_header X-Real-IP  $remote_addr;
    proxy_set_header Host $host;
```

(continues on next page)

(continued from previous page)

```
    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 Configuring WAPT Server for large deployments

The default operating system, Nginx and Postgresql settings are adapted for around 400 WAPT agents. If you have more than 400 clients it is necessary to modify a few system level parameters along with PostgreSQL database, Nginx web and WAPT Server python server.

In the future the **postconf.sh** script might take charge of this configuration depending on the expected number of client computers.

With the following parameters, one WAPT Server should scale up to around 5000 concurrent active clients. You may have more clients in the database if they are not all running at the same time. If you have more than 5000 clients it is recommended to have more than one WAPT Server.

The limit in the number of end point clients is due to the bottleneck in the python code and the PostgreSQL backend. WAPT performance gets better with time and in the future WAPT Server might support a large base on a single server. However the Nginx part scales very well and it can takes full advantage of a 10Gbps connection for high load package deployments.

Configuration changes for better performance

Note: The parameters to be modified below are linked together and should be modified globally and not individually.

Configuring 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;
```

Configuring the Linux System

Increase the number of *filedescriptors*. The system unit file asks for an increase in the allowed number of *filedescriptors* (LimitNOFILE=32768). We should have the same thing for Nginx. There are a few limits to modify.

First we modify system wide the number of *filedescriptors* allowed for Nginx and WAPT.

- create the `/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 serves as a reverse proxy and makes quite a lot of connections. Each WAPT client keeps a *websocket* connection up all the time in order to respond to actions from the WAPT Server.

The Linux kernel has a protection against having too many TCP connections opened at the same time and one may get the *SYN flooding on port* message in the Nginx log. In order to avoid these messages, it is necessary to modify the two following parameters. It must around 1.5 times the number of WAPT clients.

```
cat > /etc/sysctl.d/wapt.conf <<EOF
net.ipv4.tcp_max_syn_backlog=4096
net.core.somaxconn=4096
EOF

sysctl --system
```

Configuring the PostgreSQL database

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
```

Configuring the WAPT Server

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 for large package upload

Depending on the partitioning of your WAPT server you might have to be careful with the Nginx temporary file upload directory. Nginx acts as a reverse proxy for the WAPTServer Python engine and its does a caching of packages uploaded when uploading a new package from the console.

The packages are stored in the `/var/lib/nginx/proxy` directory. You have to make sure that the partition hosting this directory is large enough. You may change this directory location using the following Nginx configuration parameter.

```
$client_body_temp_path
```

To uninstall WAPT agent properly

6.6.13 Uninstalling WAPT agent from clients

Uninstalling WAPT agent on Windows

If you need to uninstall WAPT agents from clients, the uninstaller is automatically created in the WAPT install location, by default it is `C:\Program Files (x86)\wapt\unins000.exe`.

- default silent uninstall of a WAPT agent can be achieved with the following command:

  ```
  unins000.exe /VERYSILENT
  ```

- an additional argument can be passed to **`unins000.exe`** to cleanup everything:

  ```
  unins000.exe /VERYSILENT /purge_wapt_dir=1
  ```

Complete list of command-line arguments for **`unins000.exe`**:

Settings	Description
/VERYSILENT	Launches unins000.exe silently
/purge_wapt_dir=1	Purges WAPT directory (removes all folders and files)

Uninstalling a WAPT agent on Linux

- default uninstall of a WAPT agent can be achieved with the following command, depending on your Linux OS:

```
# Debian / Ubuntu
apt remove --purge tis-waptagent

# CentOS / Redhat
yum remove tis-waptagent
```

- an additional step can be done using these commands (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
```

Uninstalling a WAPT agent on MacOS

- default uninstall of a WAPT agent can be achieved with the following command:

```
# 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
```

Re-enabling Windows Updates before uninstalling

In the case you have used WAPT to manage Windows Updates, you might want to re-enable Windows Updates default behavior before uninstalling the WAPT agent.

To do so, here is an example package to push before uninstalling the WAPT agent:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('Disable WAPT WUA')
    inifile_writestring(WAPT.config_filename,'waptwua','enabled','false')
```

(continues on next page)

(continued from previous page)

```
    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 How to use WAPT

This section of the documentation covers the daily use of WAPT.

All WAPT functionalities are explained in detail for the *Administrators*, the *Users* and the *Package Deployers*.

6.7.1 Using the WAPT console

Installing the WAPT agent on the devices in your Organization

If you have not done so already, install the WAPT agent on a computer.

The installation of the WAPT agent on computers will register them on the WAPT inventory server.

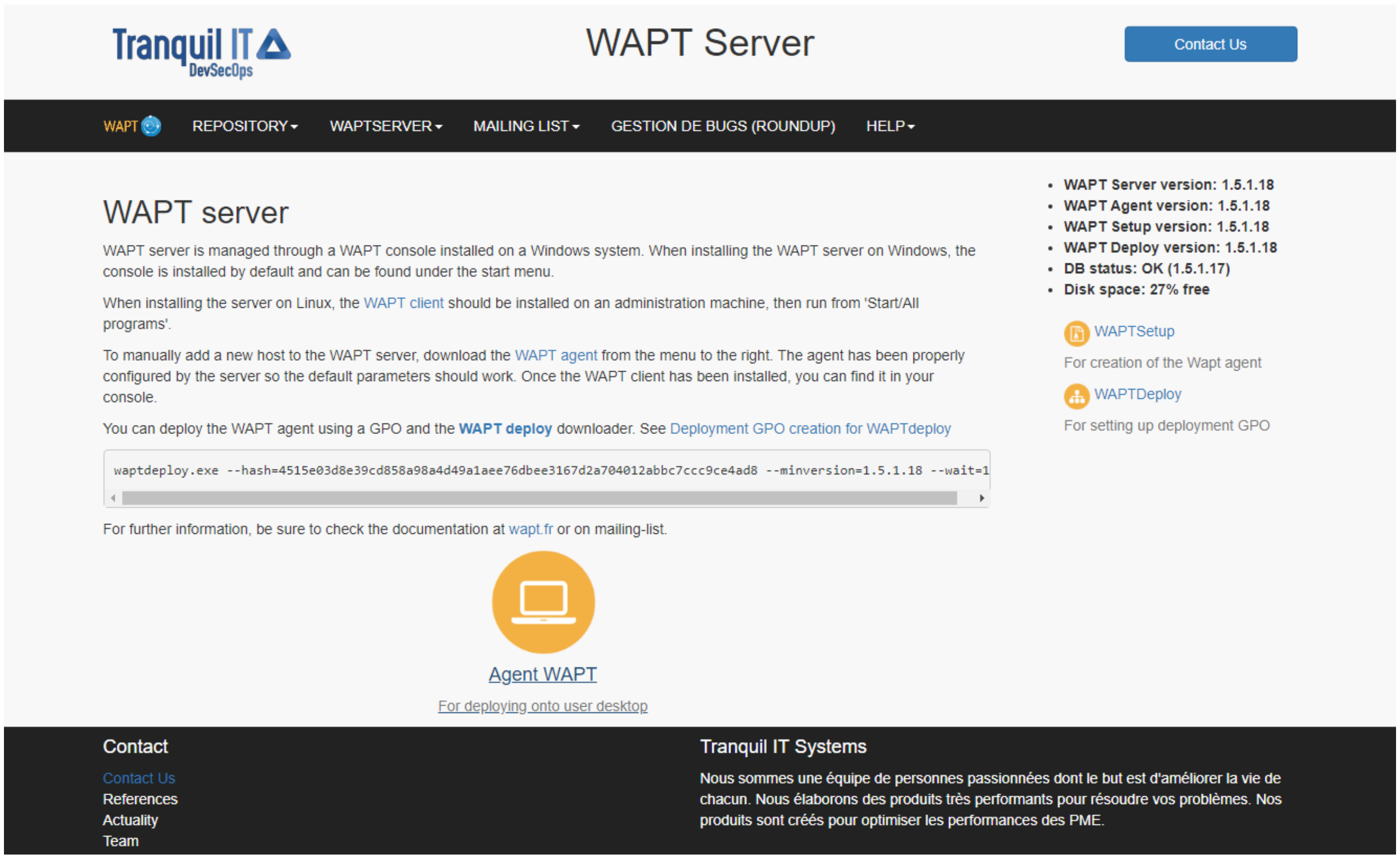
The hosts will then appear in the WAPT console.

To install the WAPT agent manually on a computer, download the WAPT agent from https://srvwapt.mydomain.lan/wapt/waptagent.exe then launch its installation.

Note: If you have skipped the step for creating the WAPT agent, return to the documentation on *installing the WAPT agent*.

On your **management computer**, hosts are displayed in the WAPT console.

Note: If a host does not appear in the console after having installed the WAPT agent, open the Windows command line utility **`cmd.exe`** on the host and type **`wapt-get register`**.



WAPT server

WAPT server is managed through a WAPT console installed on a Windows system. When installing the WAPT server on Windows, the console is installed by default and can be found under the start menu.

When installing the server on Linux, the WAPT client should be installed on an administration machine, then run from 'Start/All programs'.

To manually add a new host to the WAPT server, download the WAPT agent from the menu to the right. The agent has been properly configured by the server so the default parameters should work. Once the WAPT client has been installed, you can find it in your console.

You can deploy the WAPT agent using a GPO and the **WAPT deploy** downloader. See Deployment GPO creation for WAPTdeploy

```
waptdeploy.exe --hash=4515e03d8e39cd858a98a4d49a1aee76dbee3167d2a704012abbc7ccc9ce4ad8 --minversion=1.5.1.18 --wait=1
```

For further information, be sure to check the documentation at wapt.fr or on mailing-list.

Agent WAPT

For deploying onto user desktop

- **WAPT Server version: 1.5.1.18**
- **WAPT Agent version: 1.5.1.18**
- **WAPT Setup version: 1.5.1.18**
- **WAPT Deploy version: 1.5.1.18**
- **DB status: OK (1.5.1.17)**
- **Disk space: 27% free**

WAPTSetup

For creation of the Wapt agent

WAPTDeploy

For setting up deployment GPO

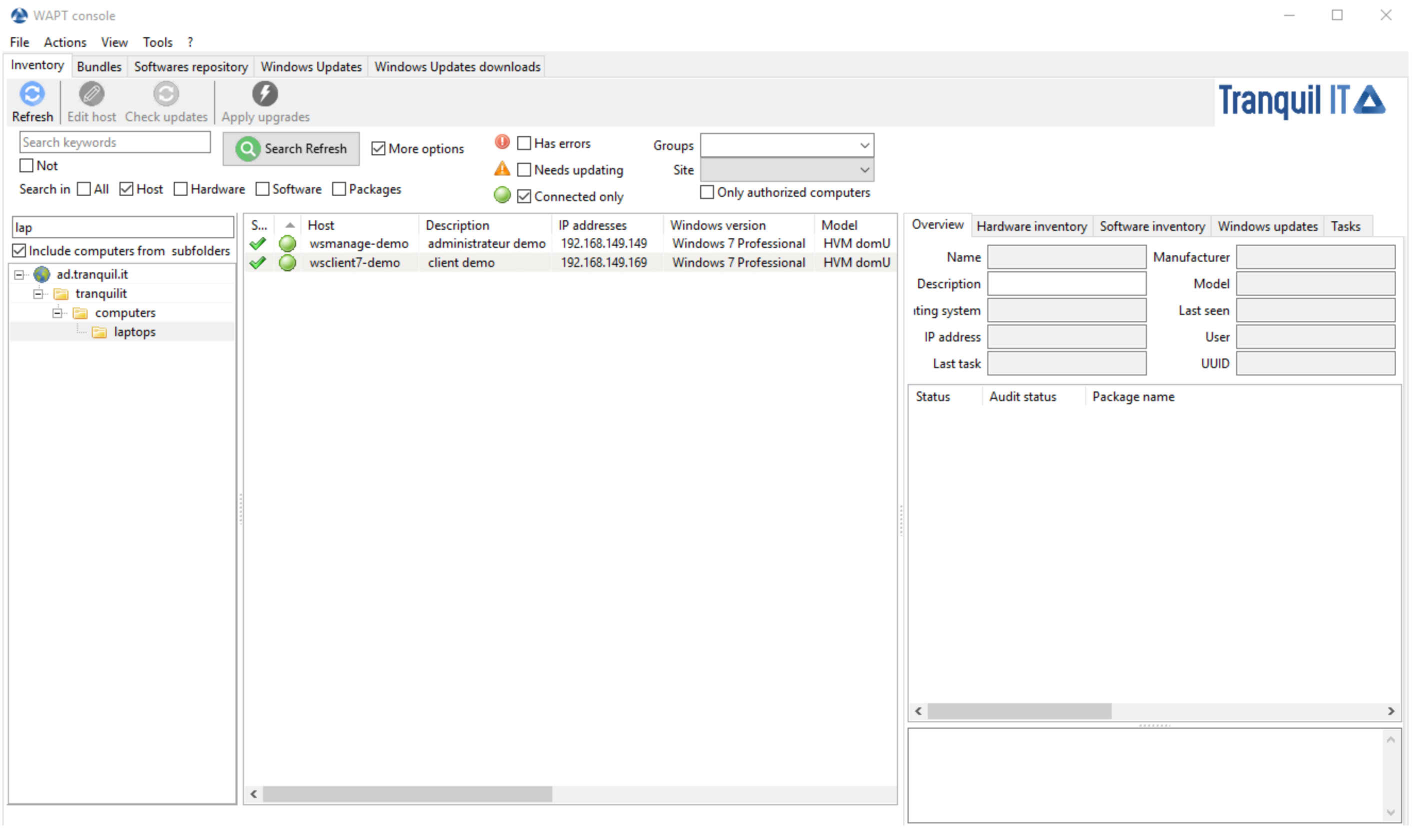
Contact

Contact Us
References
Actuality
Team

Tranquil IT Systems

Nous sommes une équipe de personnes passionnées dont le but est d'améliorer la vie de chacun. Nous élaborons des produits très performants pour résoudre vos problèmes. Nos produits sont créés pour optimiser les performances des PME.

Fig. 71: Download the WAPT agent to be deployed on computers


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository Windows Updates Windows Updates downloads
Refresh Edit host Check updates Apply upgrades
Tranquil IT
Search keywords
Search Refresh
More options
Has errors
Needs updating
Connected only
Groups
Site
Only authorized computers
Not
Search in All Host Hardware Software Packages
Overview Hardware inventory Software inventory Windows updates Tasks

[✓ More options] [✓ Connected only] Search in [✓ Host]

lap
[✓ Include computers from subfolders]

- ad.tranquil.it
  - tranquilit
    - computers
      - laptops

S...		Host	Description	IP addresses	Windows version	Model
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Professional	HVM domU
✓	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Professional	HVM domU

Name		Manufacturer	
Description		Model	
...ting system		Last seen	
IP address		User	
Last task		UUID	

Status	Audit status	Package name

Fig. 72: Inventory of hosts registered with WAPT

Duplicating packages from external repositories

Package duplication principles

Duplicating a WAPT package consists of:

- importing an existing WAPT package from an external repository;
- changing its prefix (for example from *tis* to *test*);
- resigning the WAPT package with the *Administrator*'s private key to allow the deployment of the duplicated package on your WAPT equipped hosts;
- finally, uploading it on the main WAPT repository;

Attention: By importing a package in your repository and signing it, you then become responsible for that package and for what it does. **It has been signed with your own private key**.

Tranquil IT disclaims any liability if you choose to use WAPT packages retrieved from her repositories. Without a support contract, Tranquil IT does not guarantee the suitability of the package for your own particular use case, nor do they guarantee the ability of the package to comply with your *Organization*'s internal security policies.

- go to the *Private repository* tab;

Every software package version available on the WAPT repository is shown.

If no package has been imported, the list is empty. Only the *test-waptupgrade* package will be present if the WAPT agent has been generated previously. Visit the documentation on *creating a WAPT agent*.

Two options are available to import packages:

Import a package from an external repository on the Internet

That first method allows you to download packages directly from a WAPT repository external to your *Organization*.

To import from a different repository than Tranquil IT, define a new repository address in the WAPT console preferences. For example: http://wapt.otherorganization.com/wapt/.

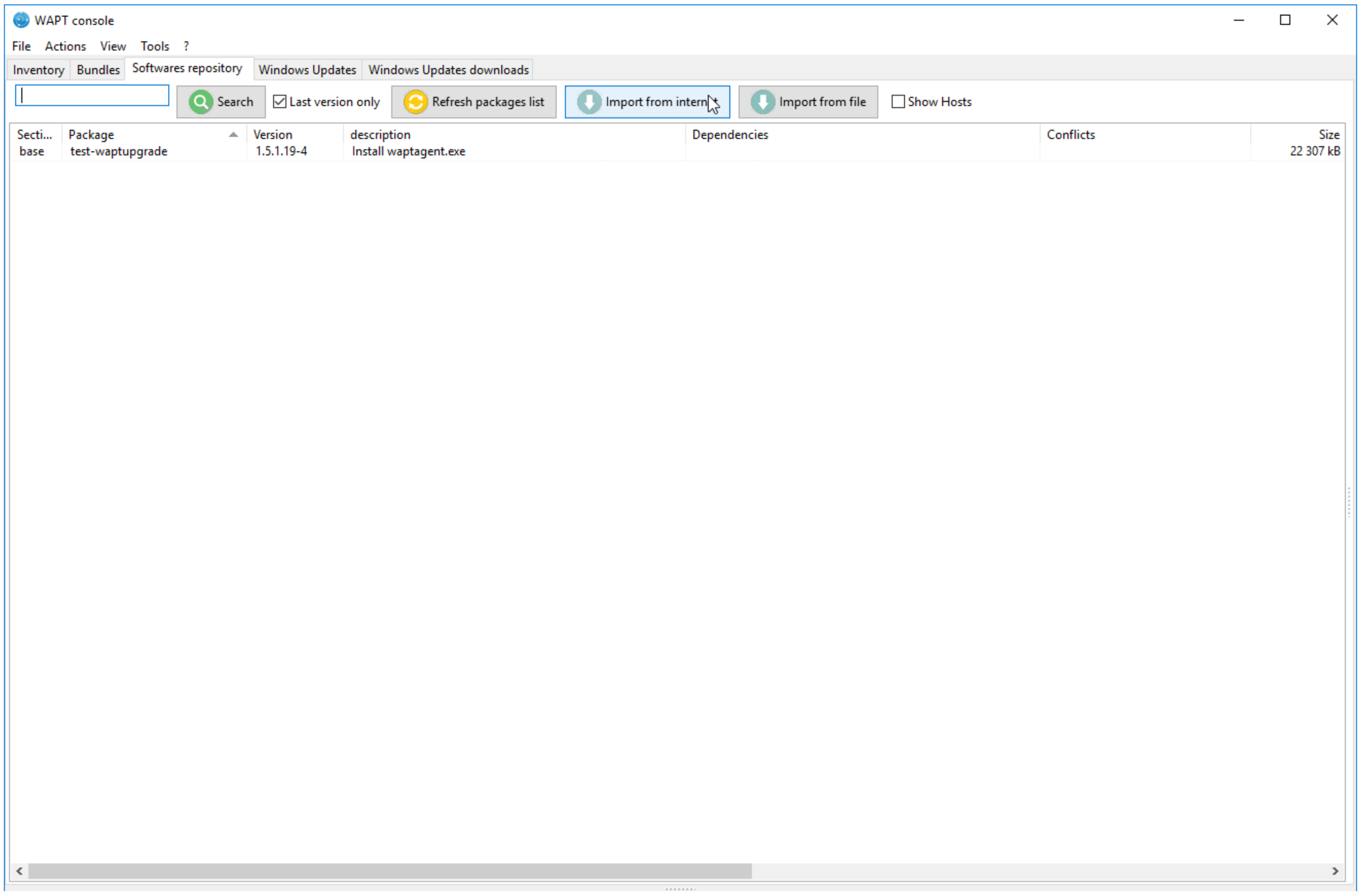
Note:

- If no repository is set, the repository https://wapt.tranquil.it/wapt will be implicitly set.
- Starting with WAPT 1.3.12.13, **external repository SSL/ TLS certificates are verified by default**.

- click on *Import from Internet*;

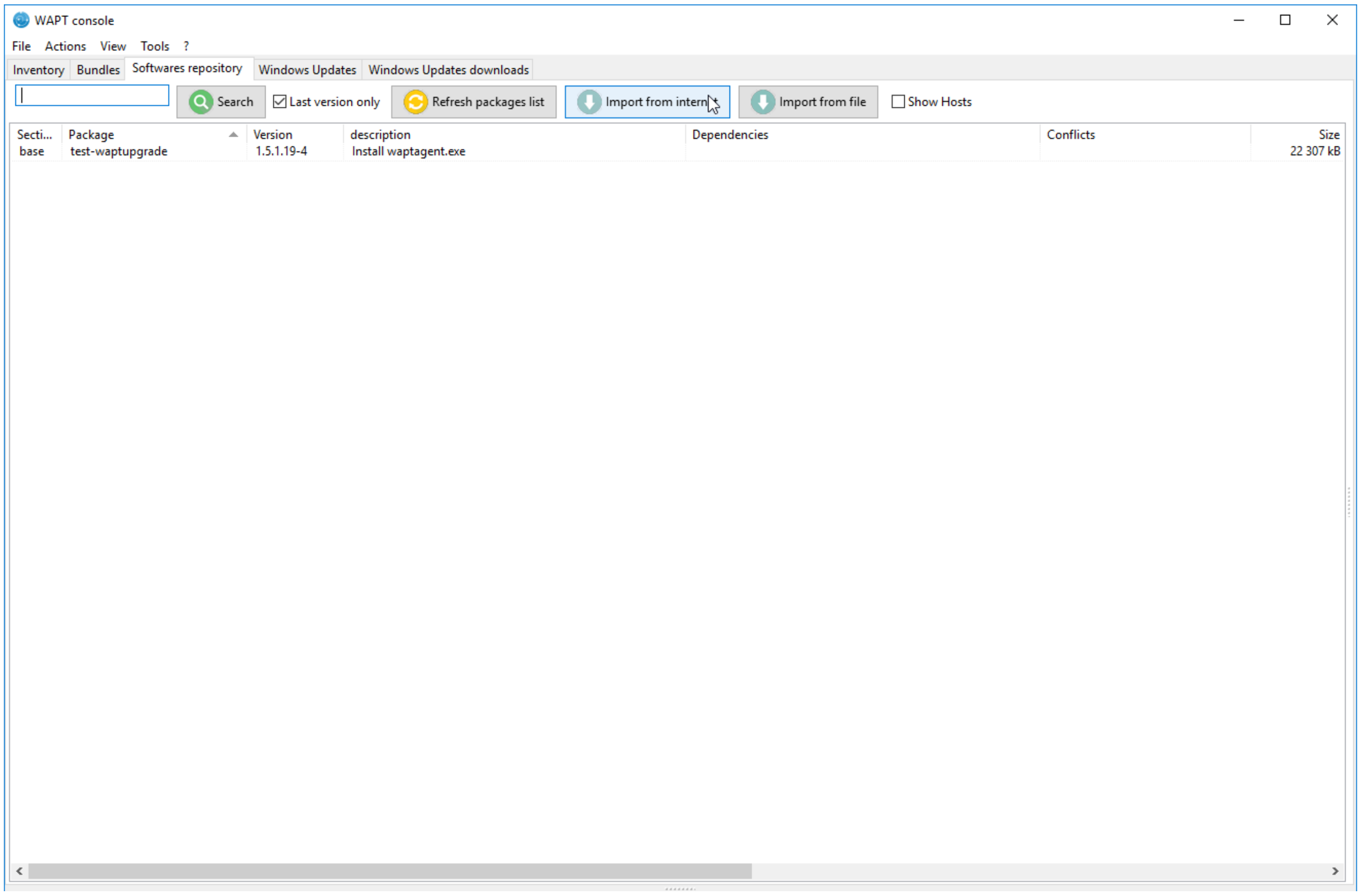
The grid view displays the list of available packages on the remote repository.

- to import a package, select a package then *Right-click* → *Import*;
- validate the duplication in your local repository;
- click on *Yes* to confirm the duplication;
- the download of the package starts …
- then, enter your private key password…



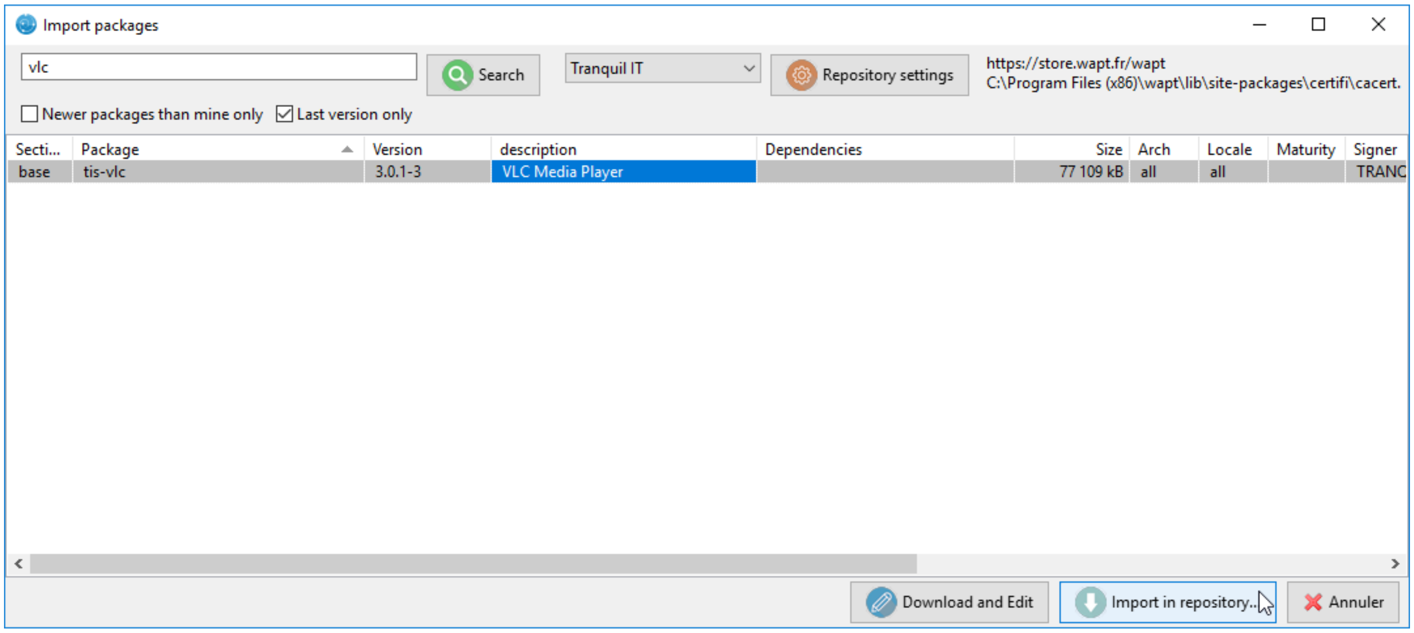
Secti...	Package	Version	description	Dependencies	Conflicts	Size
base	test-waptupgrade	1.5.1.19-4	Install waptagent.exe			22 307 kB

Fig. 73: Available software displayed in the WAPT console



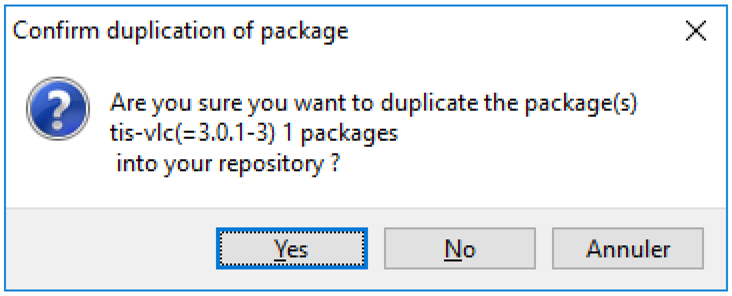
Secti...	Package	Version	description	Dependencies	Conflicts	Size
base	test-waptupgrade	1.5.1.19-4	Install waptagent.exe			22 307 kB

Fig. 74: Import a package from Internet



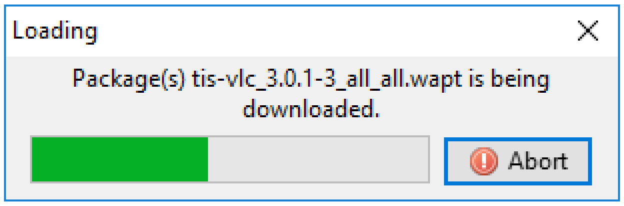
Secti...	Package	Version	description	Dependencies	Size	Arch	Locale	Maturity	Signer
base	tis-vlc	3.0.1-3	VLC Media Player		77 109 kB	all	all		TRANC

Fig. 75: Imported WAPT package in your local WAPT repository



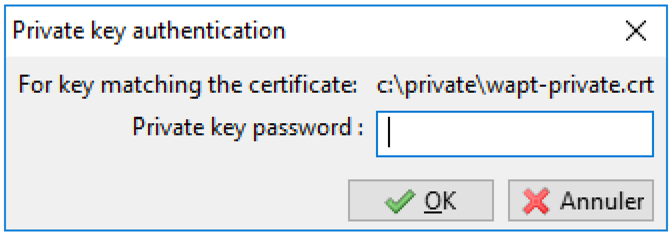
Are you sure you want to duplicate the package(s) tis-vlc(=3.0.1-3) 1 packages into your repository ?

Fig. 76: Confirm the duplication of the package



Package(s) tis-vlc_3.0.1-3_all_all.wapt is being downloaded.

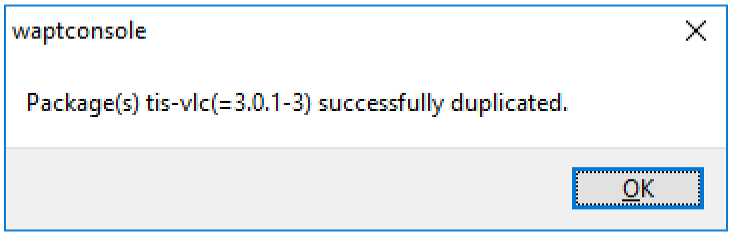
Fig. 77: Progress of the package duplication process



For key matching the certificate: c:\private\wapt-private.crt
Private key password :

Fig. 78: Enter the password for unlocking the private key

The WAPT console confirms that the package has been duplicated in your local WAPT repository.



```
waptconsole

Package(s) tis-vlc(=3.0.1-3) successfully duplicated.
```

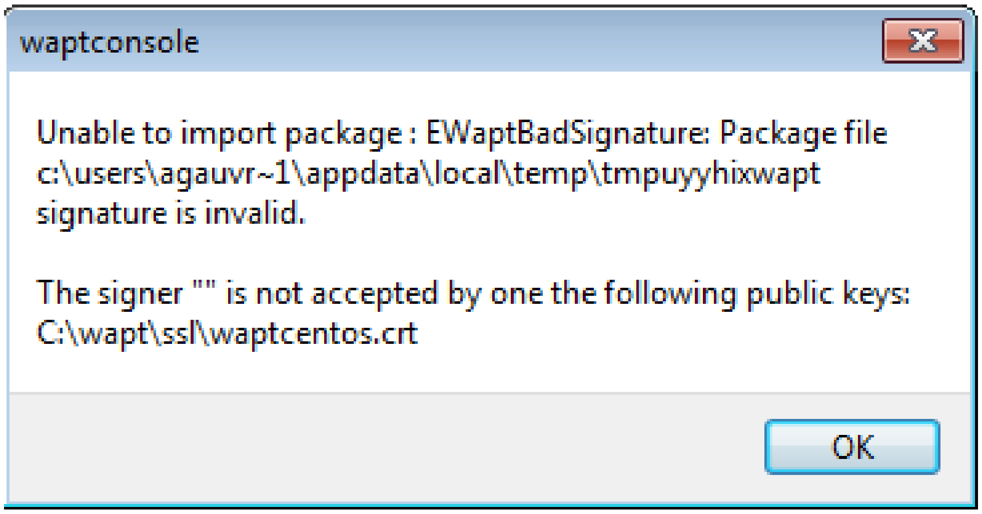
Fig. 79: Confirmation of successful duplication

The package then appears in your local WAPT repository with your Organization's prefix.

Attention: If the verification of the package signature is enabled, the public certificate of the signer must be located in one of the following folders:

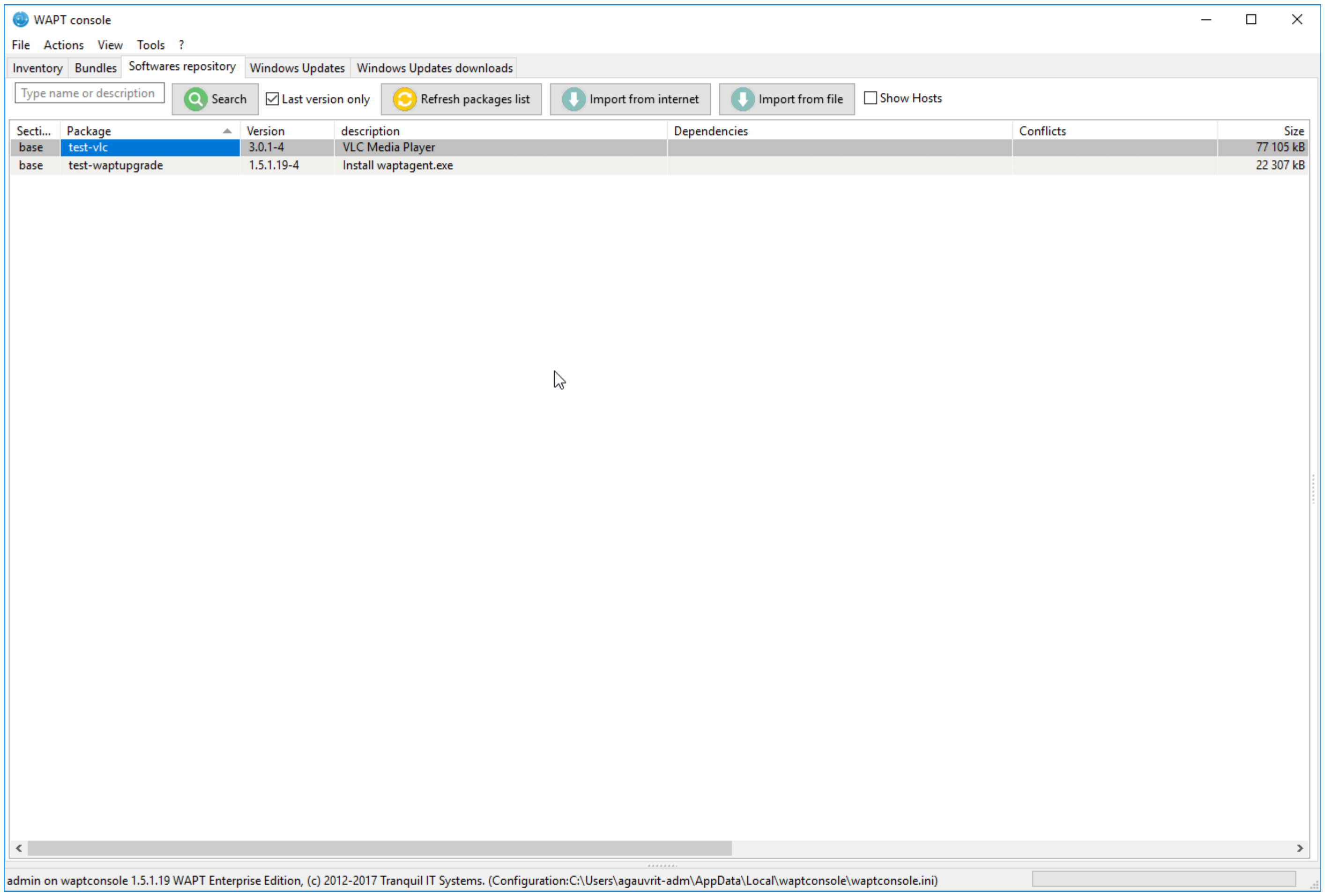
- `C:\Program Files (x86)\wapt\ssl;`
- `%appdata%\waptconsole\ssl;`

If the certificate is not found in one of these two folders, then the following error will occur and the package will not be imported.



```
waptconsole

Unable to import package : EWaptBadSignature: Package file
c:\users\agauvr~1\appdata\local\temp\tmpuyyhixwapt
signature is invalid.

The signer "" is not accepted by one the following public keys:
C:\wapt\ssl\waptcentos.crt
```

Fig. 81: Error while validating the signature of the external repository



Secti...	Package	Version	description	Dependencies	Conflicts	Size
base	test-vlc	3.0.1-4	VLC Media Player			77 105 kB
base	test-waptupgrade	1.5.1.19-4	Install waptagent.exe			22 307 kB

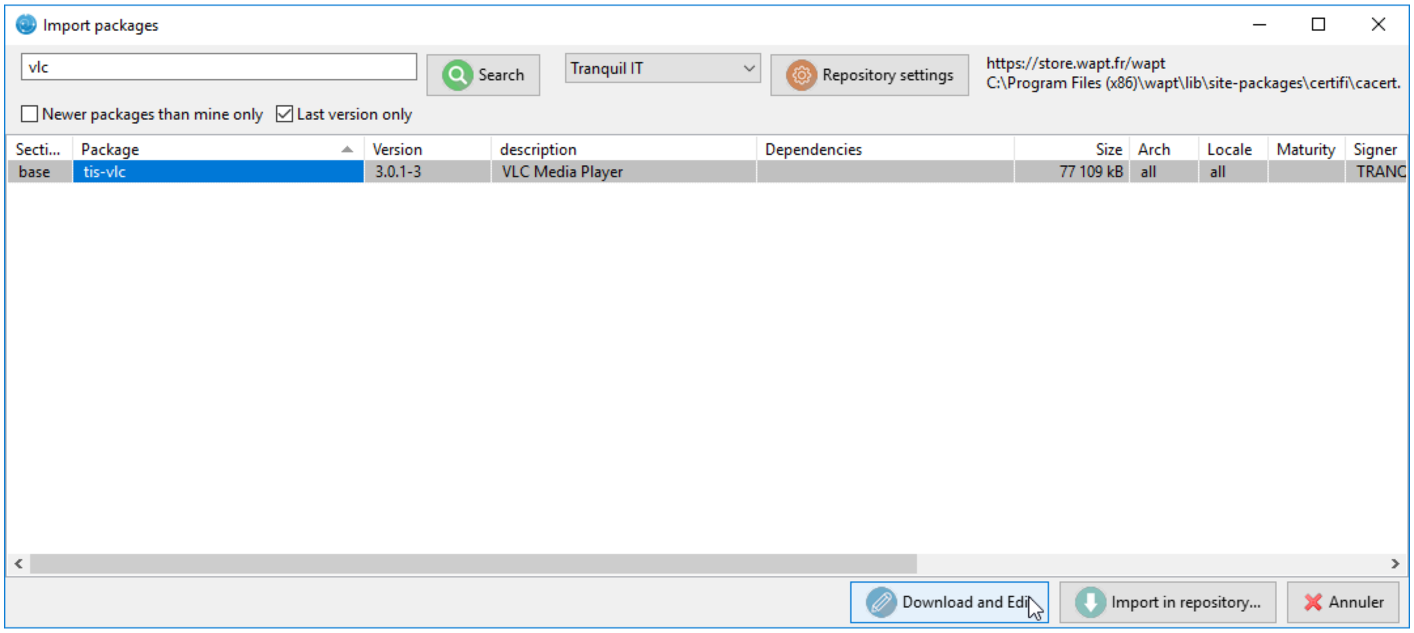
admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\waptconsole.ini)

Fig. 80: WAPT console displaying the duplicated package

Editing a package before importing it

Starting with WAPT 1.3.12.13, it is now possible to edit a package downloaded from an external repository before importing it in your main WAPT repository.

To achieve this, choose instead the second option *Download and Edit* to import the package from an external WAPT repository.



Secti...	Package	Version	description	Dependencies	Size	Arch	Locale	Maturity	Signer
base	tis-vlc	3.0.1-3	VLC Media Player		77 109 kB	all	all		TRANC

Fig. 82: Process for importing and editing a package

PyScripter, if installed, opens the WAPT package.

Please refer to the documentation on *creating WAPT packages from scratch*.

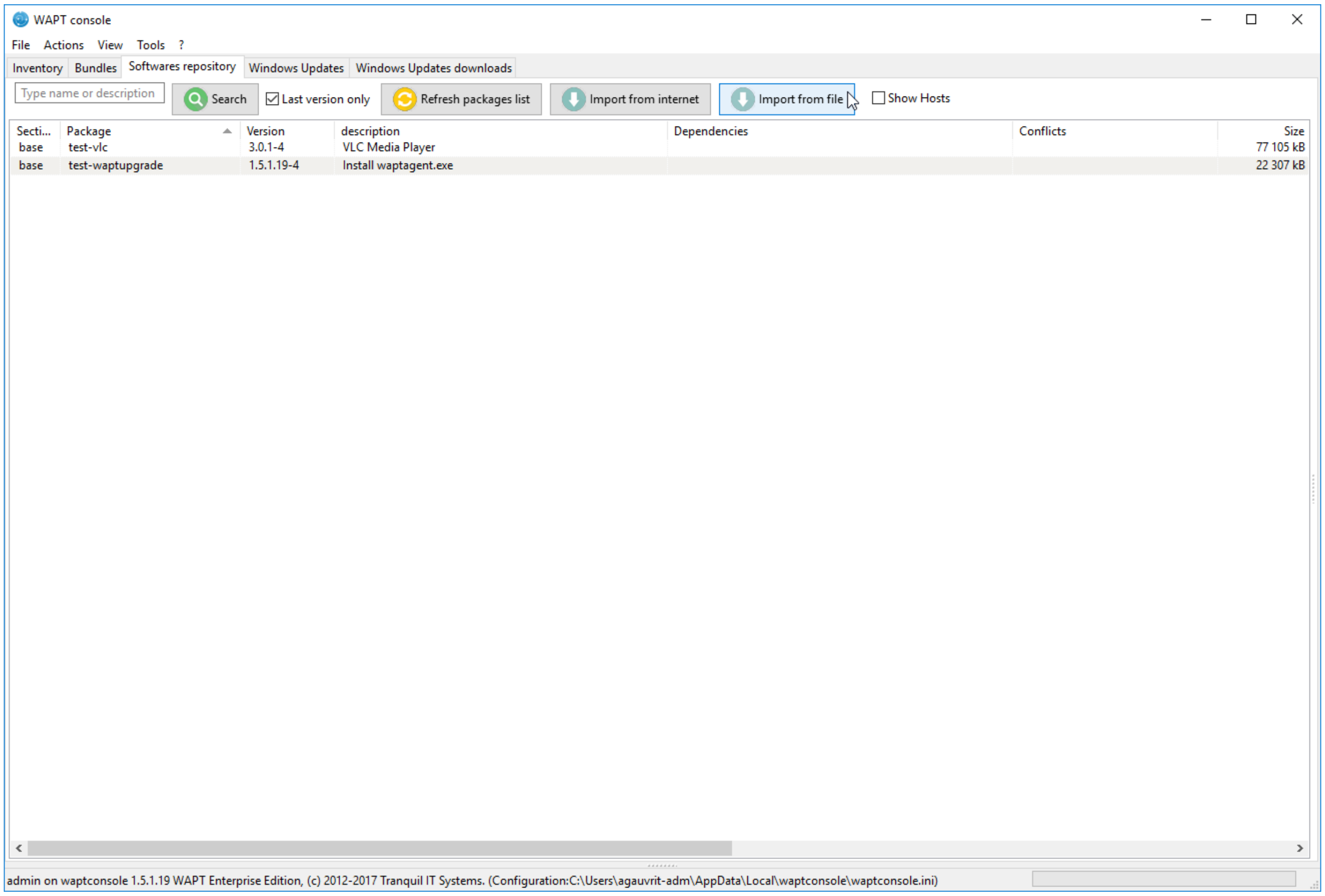
Importing a WAPT package from a file

That second method allows you to import a `.wapt` file from any medium.

- click on *Import from File*;
- select the file to import;
- click on *Open* to import the file;

The WAPT console confirms that the package has been duplicated in your local WAPT repository.

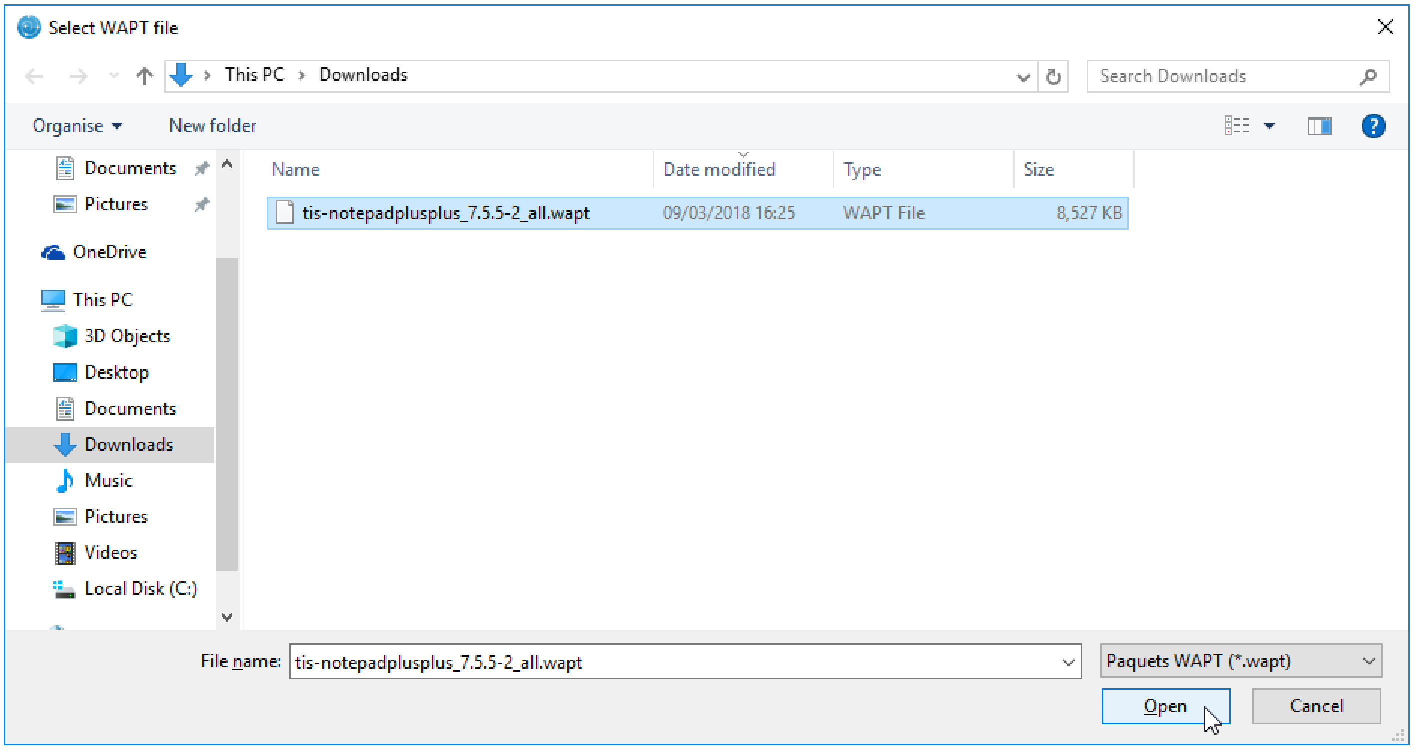
The package then appears in your local WAPT repository with your Organization's prefix.



Secti...	Package	Version	description	Dependencies	Conflicts	Size
base	test-vlc	3.0.1-4	VLC Media Player			77 105 kB
base	test-waptupgrade	1.5.1.19-4	Install waptagent.exe			22 307 kB

admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\waptconsole.ini)

Fig. 83: Import from a file

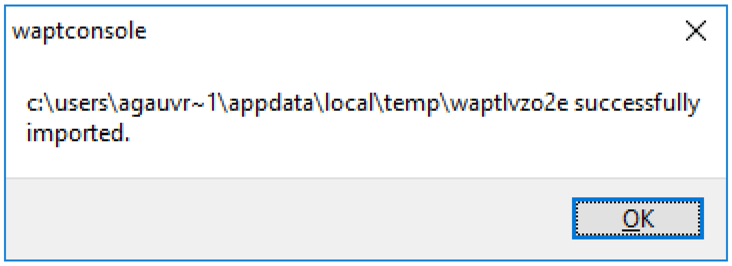


Name	Date modified	Type	Size
tis-notepadplusplus_7.5.5-2_all.wapt	09/03/2018 16:25	WAPT File	8,527 KB

File name: tis-notepadplusplus_7.5.5-2_all.wapt

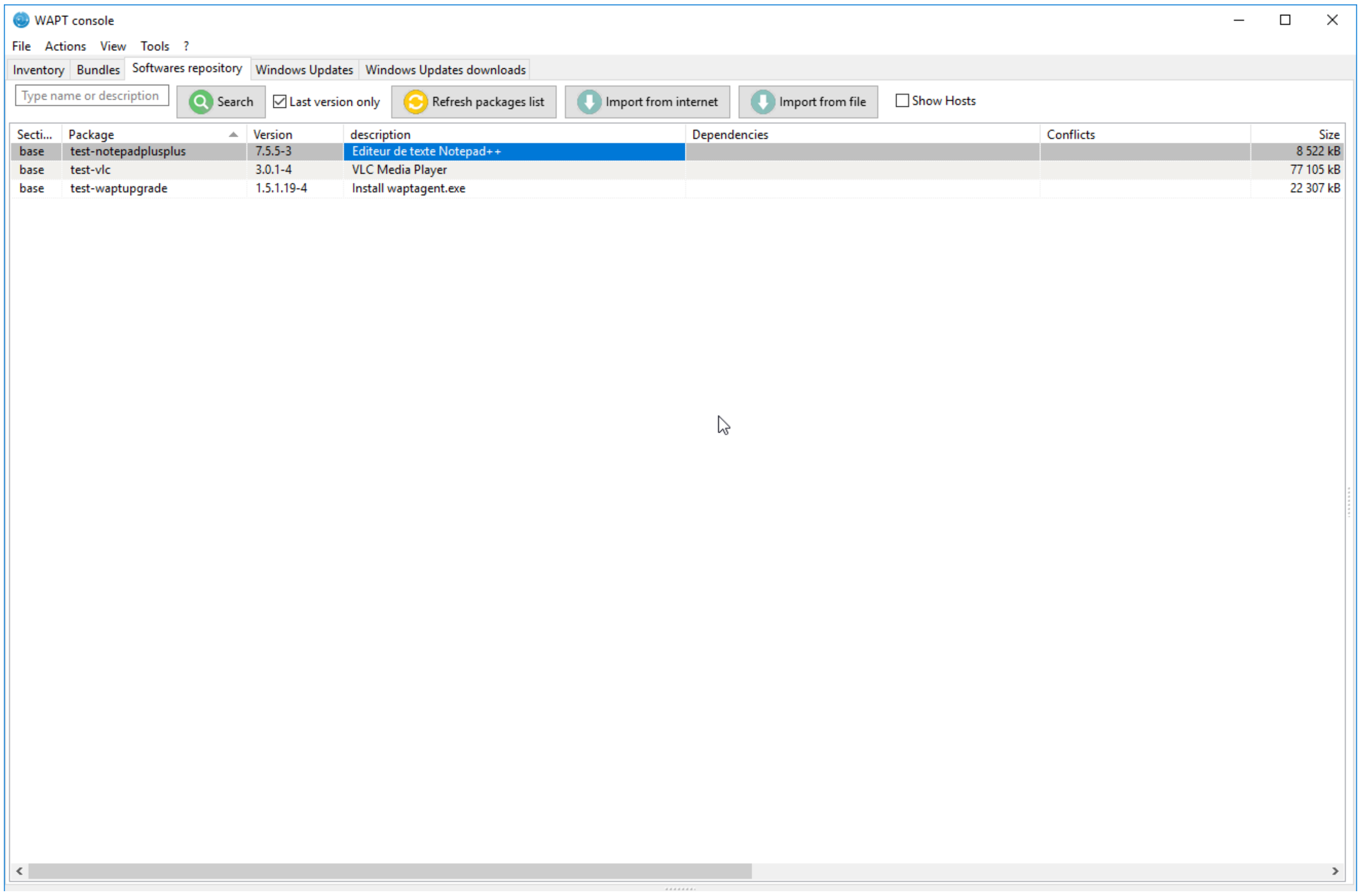
Paquets WAPT (*.wapt)

Fig. 84: Select the file to import


waptconsole

c:\users\agauvr~1\appdata\local\temp\waptlvzo2e successfully imported.

Fig. 85: File imported successfully



Secti...	Package	Version	description	Dependencies	Conflicts	Size
base	test-notepadplusplus	7.5.5-3	Editeur de texte Notepad++			8 522 kB
base	test-vlc	3.0.1-4	VLC Media Player			77 105 kB
base	test-waptupgrade	1.5.1.19-4	Install waptagent.exe			22 307 kB

Fig. 86: Imported WAPT package in your local WAPT repository

Changing the prefix and re-signing a WAPT package

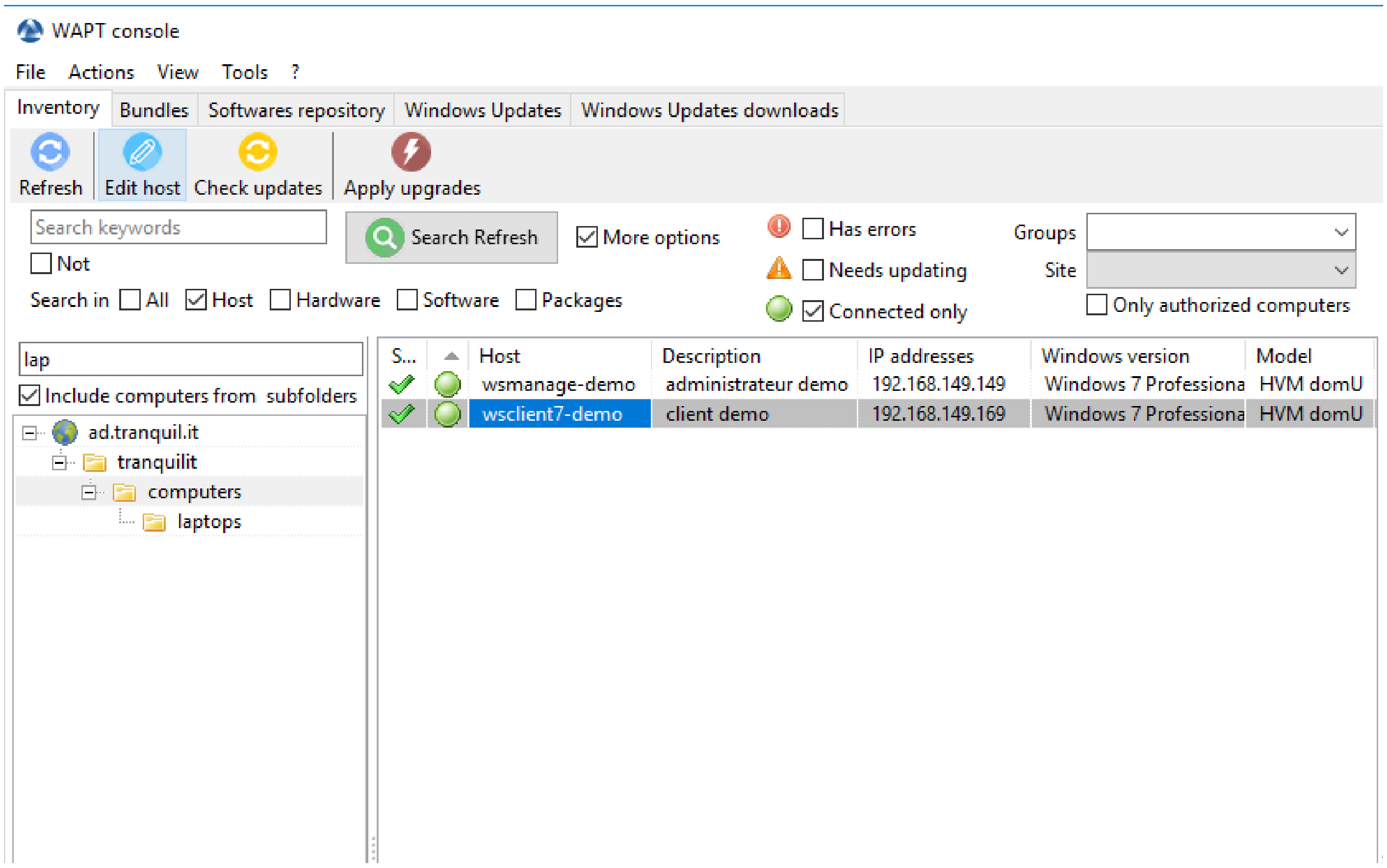
When importing, the changing of the prefix and the re-signing of the WAPT package are transparent and automatic.

Once the package is ready, the WAPT package is uploaded onto the main WAPT repository.

Deploying WAPT packages from the WAPT console

- edit the host onto which you want to deploy a WAPT package;

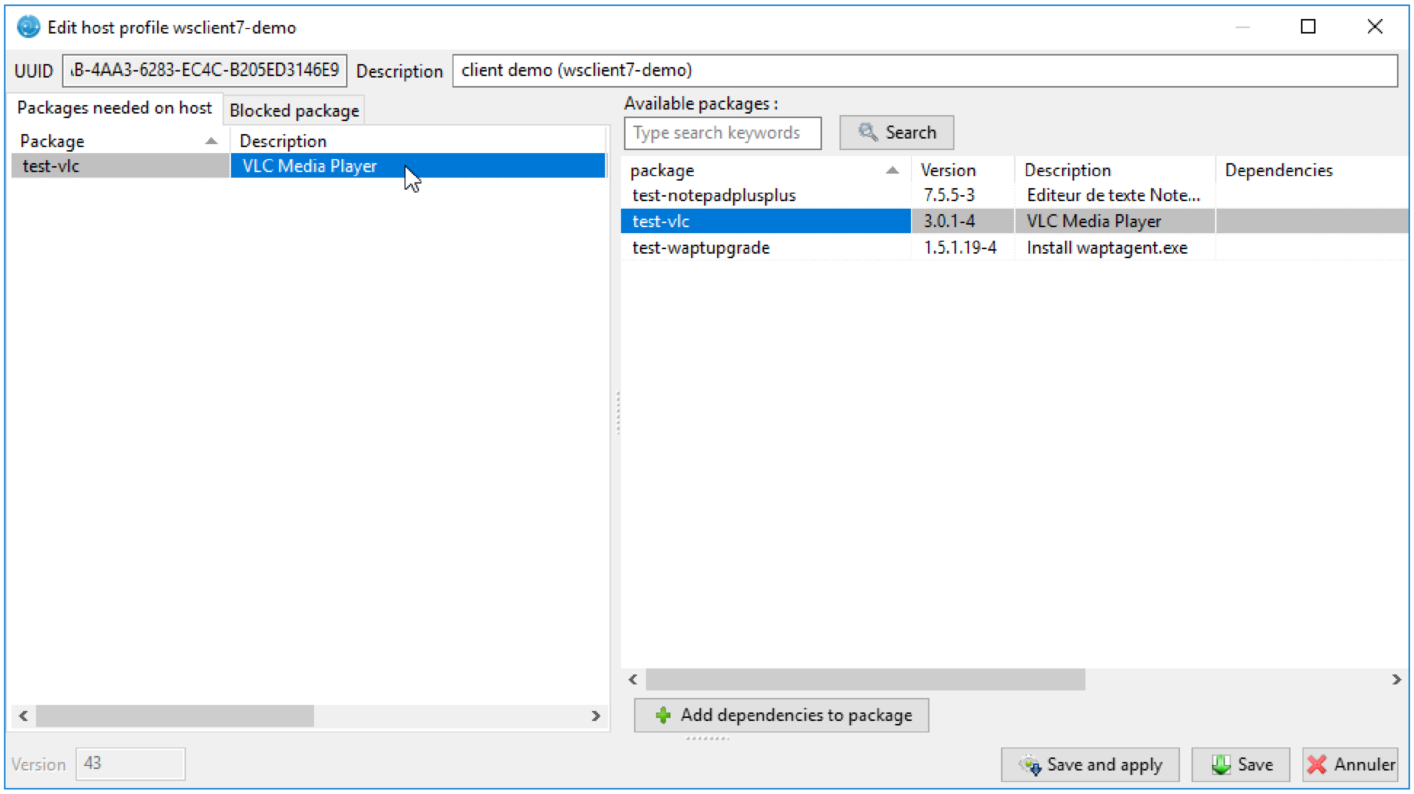
Note: Selecting multiple hosts using common shortcut keys `Control-A` or `Shift-Arrow` is possible.



S...		Host	Description	IP addresses	Windows version	Model
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Professiona	HVM domU
✓	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Professiona	HVM domU

Fig. 87: Select the host to configure

- A window opens, on the right side appears the list of packages available on the local WAPT repository, and on the left side it shows the list of packages currently assigned to the host.
- drag and drop packages from the right pane to the left pane;
- clicking on *Save and Apply to hosts* will launch the installation of the package(s) immediately on the selected host(s) that are connected to the WAPT Server;

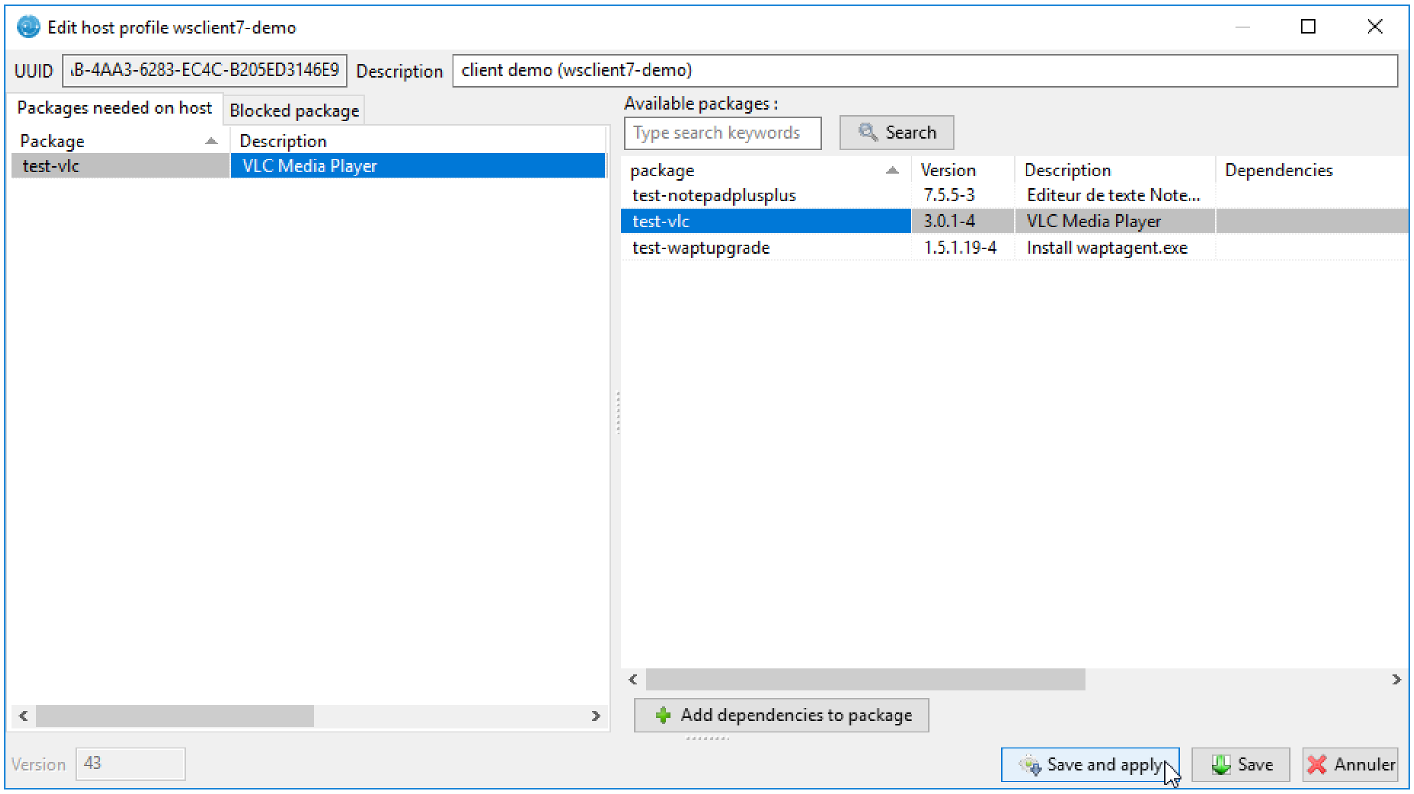


Package	Description
test-vlc	VLC Media Player

package	Version	Description	Dependencies
test-notepadplusplus	7.5.5-3	Editeur de texte Note...	
test-vlc	3.0.1-4	VLC Media Player	
test-waptupgrade	1.5.1.19-4	Install waptagent.exe	

Fig. 88: Drag and drop the package on the host or the selection of hosts

- clicking on *Save* will save the current configuration. Upgrading of the packages will occur during the WAPT agents' next update cycle;


Edit host profile wsclient7-demo

UUID ‹B-4AA3-6283-EC4C-B205ED3146E9 Description client demo (wsclient7-demo)

Packages needed on host | Blocked package

Package	Description
test-vlc	VLC Media Player

Available packages :

Type search keywords | Search

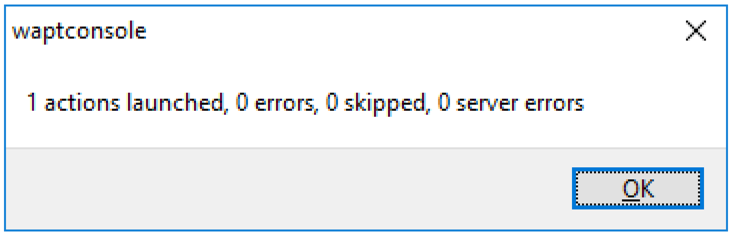
package	Version	Description	Dependencies
test-notepadplusplus	7.5.5-3	Editeur de texte Note...	
test-vlc	3.0.1-4	VLC Media Player	
test-waptupgrade	1.5.1.19-4	Install waptagent.exe	

Add dependencies to package

Version 43

Save and apply | Save | Annuler

Fig. 89: Save and apply configuration on selected host(s)


waptconsole

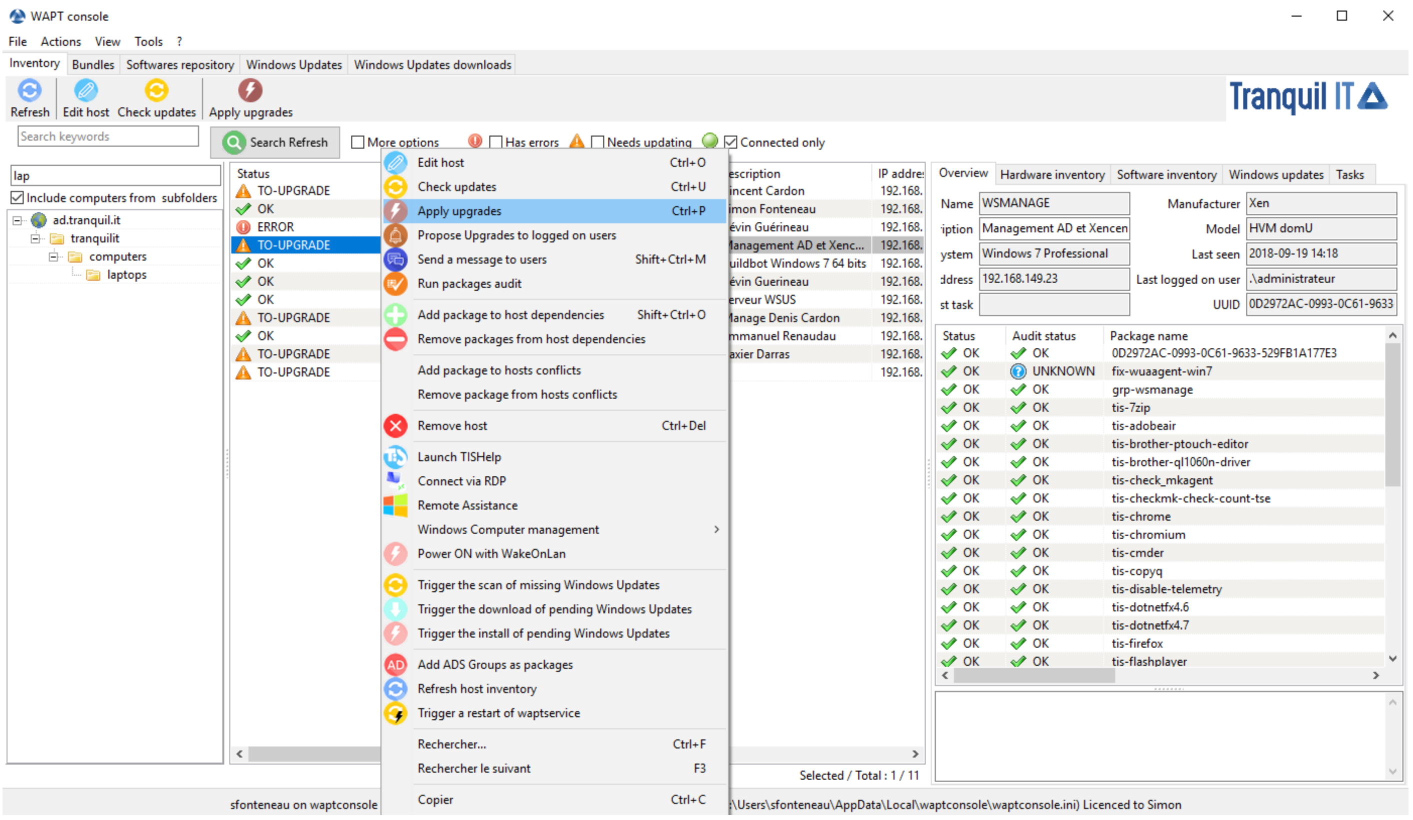
1 actions launched, 0 errors, 0 skipped, 0 server errors

OK

Fig. 90: Update launched

To launch the installation of WAPT packages, click successively on *Update available packages* then *Apply updates*.

The installation of the WAPT package(s) is launched on the selected host(s) connected to the WAPT Server.


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository Windows Updates Windows Updates downloads
Refresh Edit host Check updates Apply upgrades
Tranquil IT
Search keywords
Search Refresh
More options Has errors Needs updating Connected only
Include computers from subfolders
ad.tranquil.it
tranquilit
computers
laptops
Edit host Ctrl+O
Check updates Ctrl+U
Apply upgrades Ctrl+P
Propose Upgrades to logged on users
Send a message to users Shift+Ctrl+M
Run packages audit
Add package to host dependencies Shift+Ctrl+O
Remove packages from host dependencies
Add package to hosts conflicts
Remove package from hosts conflicts
Remove host Ctrl+Del
Launch TISHelp
Connect via RDP
Remote Assistance
Windows Computer management
Power ON with WakeOnLan
Trigger the scan of missing Windows Updates
Trigger the download of pending Windows Updates
Trigger the install of pending Windows Updates
Add ADS Groups as packages
Refresh host inventory
Trigger a restart of waptservice
Rechercher... Ctrl+F
Rechercher le suivant F3
Copier Ctrl+C
Overview Hardware inventory Software inventory Windows updates Tasks

lap

Status	escription	IP addres
TO-UPGRADE	incent Cardon	192.168.
OK	imon Fonteneau	192.168.
ERROR	évin Guérineau	192.168.
TO-UPGRADE	lanagement AD et Xenc...	192.168.
OK	uildbot Windows 7 64 bits	192.168.
OK	évin Guerineau	192.168.
OK	erveur WSUS	192.168.
TO-UPGRADE	Ianage Denis Cardon	192.168.
OK	mmanuel Renaudau	192.168.
TO-UPGRADE	axier Darras	192.168.
TO-UPGRADE		192.168.

Selected / Total : 1 / 11

Name	WSMANAGE	Manufacturer	Xen
iption	Management AD et Xencen	Model	HVM domU
ystem	Windows 7 Professional	Last seen	2018-09-19 14:18
ddress	192.168.149.23	Last logged on user	.\administrateur
st task		UUID	0D2972AC-0993-0C61-9633

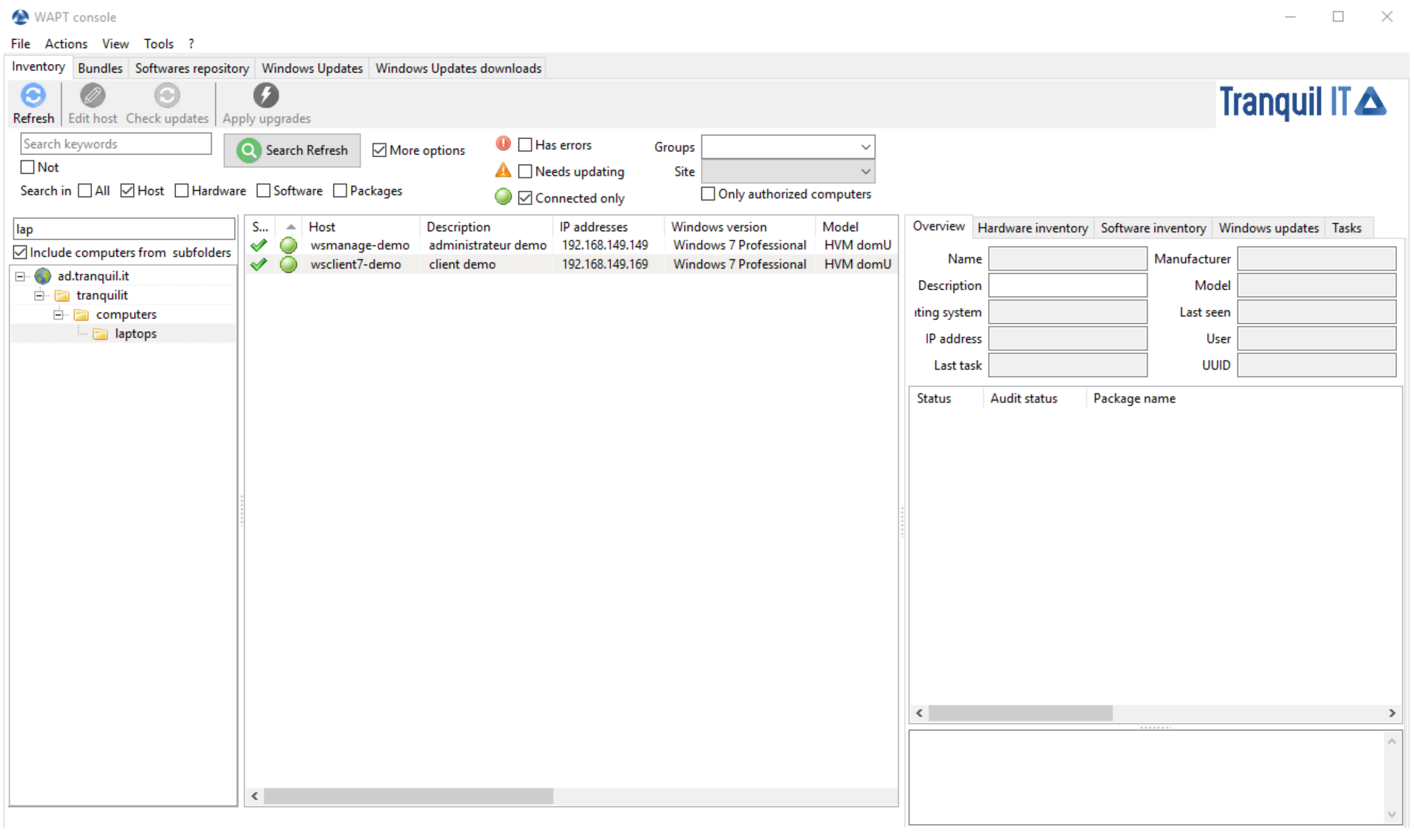
Status	Audit status	Package name
OK	OK	0D2972AC-0993-0C61-9633-529FB1A177E3
OK	UNKNOWN	fix-wuaagent-win7
OK	OK	grp-wsmanage
OK	OK	tis-7zip
OK	OK	tis-adobeair
OK	OK	tis-brother-ptouch-editor
OK	OK	tis-brother-ql1060n-driver
OK	OK	tis-check_mkagent
OK	OK	tis-checkmk-check-count-tse
OK	OK	tis-chrome
OK	OK	tis-chromium
OK	OK	tis-cmder
OK	OK	tis-copyq
OK	OK	tis-disable-telemetry
OK	OK	tis-dotnetfx4.6
OK	OK	tis-dotnetfx4.7
OK	OK	tis-firefox
OK	OK	tis-flashplayer

sfonteneau on waptconsole :\Users\sfonteneau\AppData\Local\waptconsole\waptconsole.ini) Licenced to Simon

Fig. 91: Applying updates

6.7.2 Using the WAPT console (detailed)

Note: Some functionalities detailed here are only available with the **Enterprise** version of WAPT.



[ ] Not
Search in [ ] All [✓] Host [ ] Hardware [ ] Software [ ] Packages
[✓] More options
[ ] Has errors
[ ] Needs updating
[✓] Connected only
[ ] Only authorized computers

lap
[✓] Include computers from subfolders

- ad.tranquil.it
  - tranquilit
    - computers
      - laptops

S...		Host	Description	IP addresses	Windows version	Model
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Professional	HVM domU
✓	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Professional	HVM domU

Fig. 92: Software inventory as registered in the Windows registry of the host

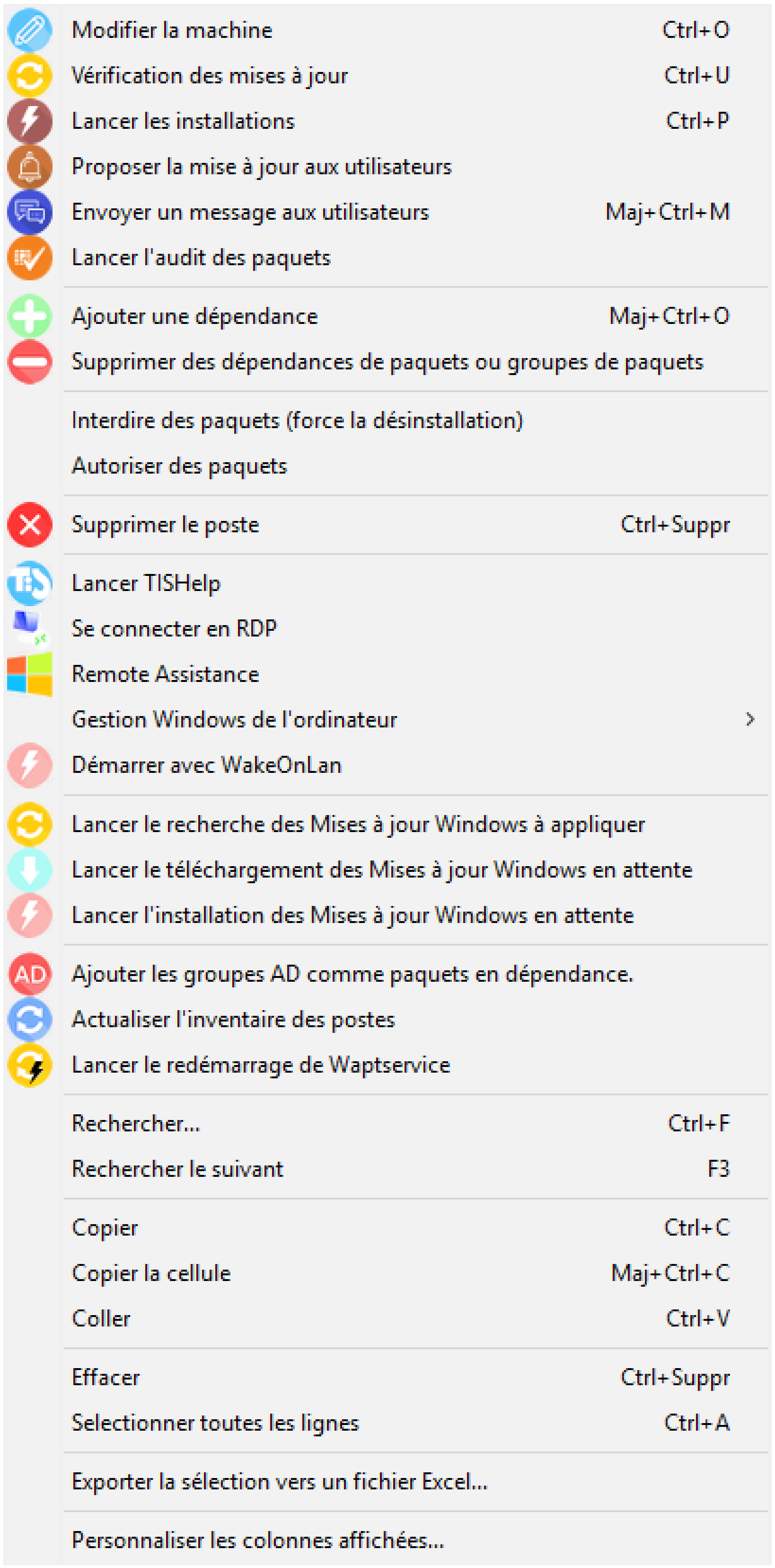

Modifier la machine Ctrl+O
Vérification des mises à jour Ctrl+U
Lancer les installations Ctrl+P
Proposer la mise à jour aux utilisateurs
Envoyer un message aux utilisateurs Maj+Ctrl+M
Lancer l'audit des paquets
Ajouter une dépendance Maj+Ctrl+O
Supprimer des dépendances de paquets ou groupes de paquets
Interdire des paquets (force la désinstallation)
Autoriser des paquets
Supprimer le poste Ctrl+Suppr
Lancer TISHelp
Se connecter en RDP
Remote Assistance
Gestion Windows de l'ordinateur
Démarrer avec WakeOnLan
Lancer le recherche des Mises à jour Windows à appliquer
Lancer le téléchargement des Mises à jour Windows en attente
Lancer l'installation des Mises à jour Windows en attente
Ajouter les groupes AD comme paquets en dépendance.
Actualiser l'inventaire des postes
Lancer le redémarrage de Waptservice
Rechercher... Ctrl+F
Rechercher le suivant F3
Copier Ctrl+C
Copier la cellule Maj+Ctrl+C
Coller Ctrl+V
Effacer Ctrl+Suppr
Selectionner toutes les lignes Ctrl+A
Exporter la sélection vers un fichier Excel...
Personnaliser les colonnes affichées...

Fig. 93: Host configuration menu

How to perform actions on the hosts?

Table 16: List of actions available to be performed on the hosts from the WAPT console

Description	Multi-selection
Edit the configuration of the host	*yes*
Refresh the list of available packages	*yes*
Offer the user to launch an upgrade	*yes*
Send a message to the selected hosts	*yes*
Audit the packages installed on selected hosts	*yes*
Add one or more packages on selected hosts	*yes*
Remove one or more packages from selected hosts	*yes*
Prevent packages from being installed on hosts	*yes*
Add one or more packages on selected hosts	*yes*
Remove the host and/or the host package	*yes*
Launch TIShelp with a remote host	*no*
Initiate a VNC connection with the remote host	*no*
Connect via Veyon	*no*
Launch a RDP connection on the selected host	*no*
Launch remote user support (msra.exe)	*no*
Manage hosts with **`compmgmt.msc`**	*no*
Manage Users and Groups with **`lusrmgr.msc`**	*no*
Manage Services with **`services.msc`**	*no*
Send a WakeOnLAN special packet to selected hosts	*yes*
Search for applicable Windows updates	*yes*
Download pending Windows updates	*yes*
Trigger the installation of pending Windows updates	*yes*
Add Active Directory groups to the selected machines	*yes*
Trigger an inventory update on selected hosts	*yes*
Restart the WAPT service	*yes*

Searching a host

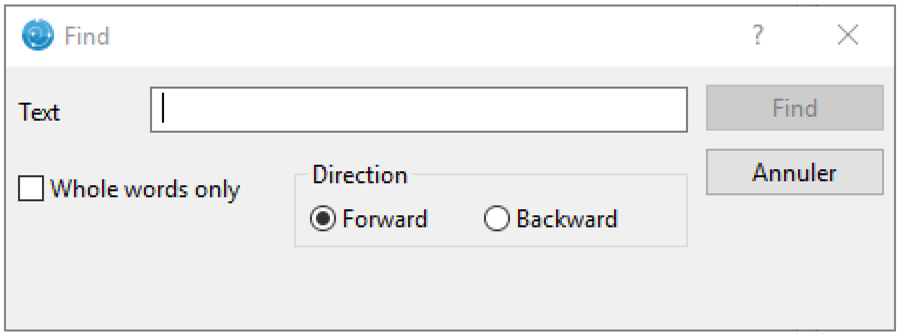


Fig. 94: *Search* function in WAPT

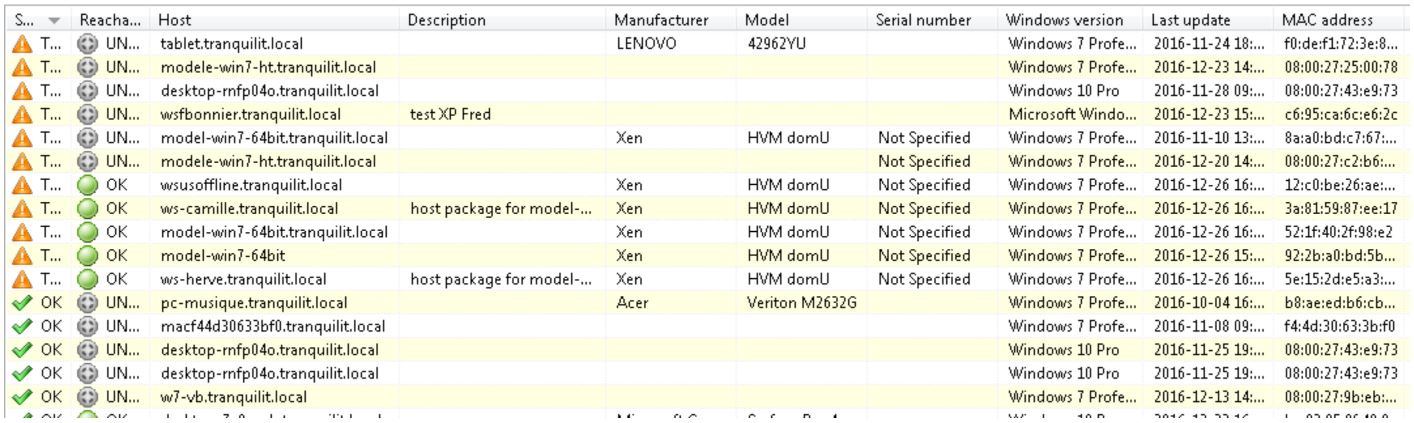
Allows you to search for a value in the selected column.

Show the inventory

When the WAPT agents **`register`**, they send some information to the WAPT Server.

Information displayed in the console is not updated in real-time, you have to refresh the display to view new status and information.

Click on the *Refresh* button or press `F5` on the keyboard.



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:...

Fig. 95: WAPT console displaying inventory

The WAPT console lists hosts that are registered with the WAPT Server and some information that is useful for managing the hosts.

Selecting a host displays its information in the right panel of the WAPT console (*Hardware inventory* and *Software inventory*).

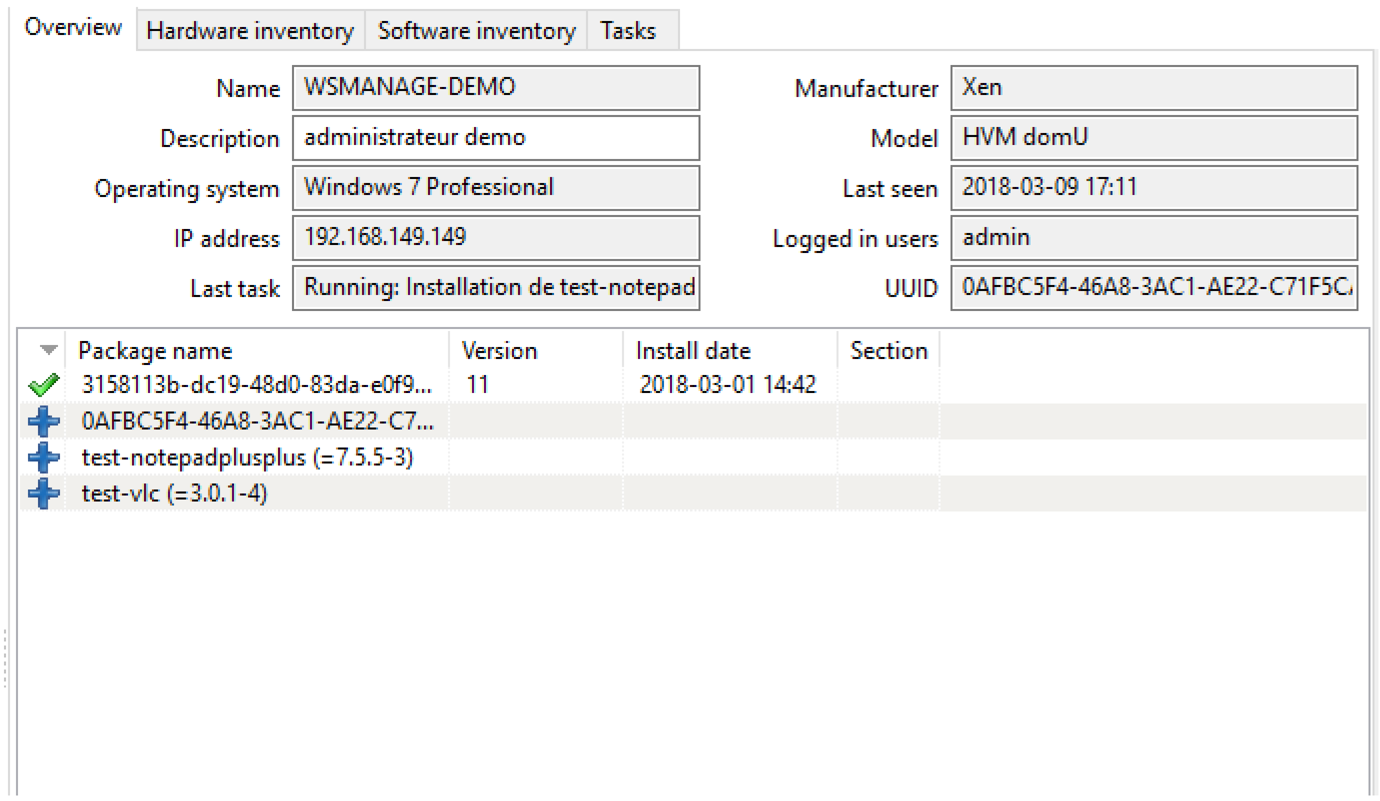
Hardware inventory displays the hardware inventory of the host

Common informations displayed in the *Hardware inventory* tab is:

- the name of the host;
- the description of the host;
- the operating system running on the host;
- the IP address of the host;
- the last WAPT task that was run on the host;
- the manufacturer of the host;
- the model of the host;
- the date of the latest update on the host;
- the name of the user last or currently connected on host;

Table 17: Status of packages in the WAPT console

Description	Status
List of installed WAPT packages	Status: **OK**
List of packages waiting to be installed	Status: **MISSING**
List of packages pending updates	Status: **NEED-UPGRADE**
List of packages that have failed to install	Status: **ERROR**



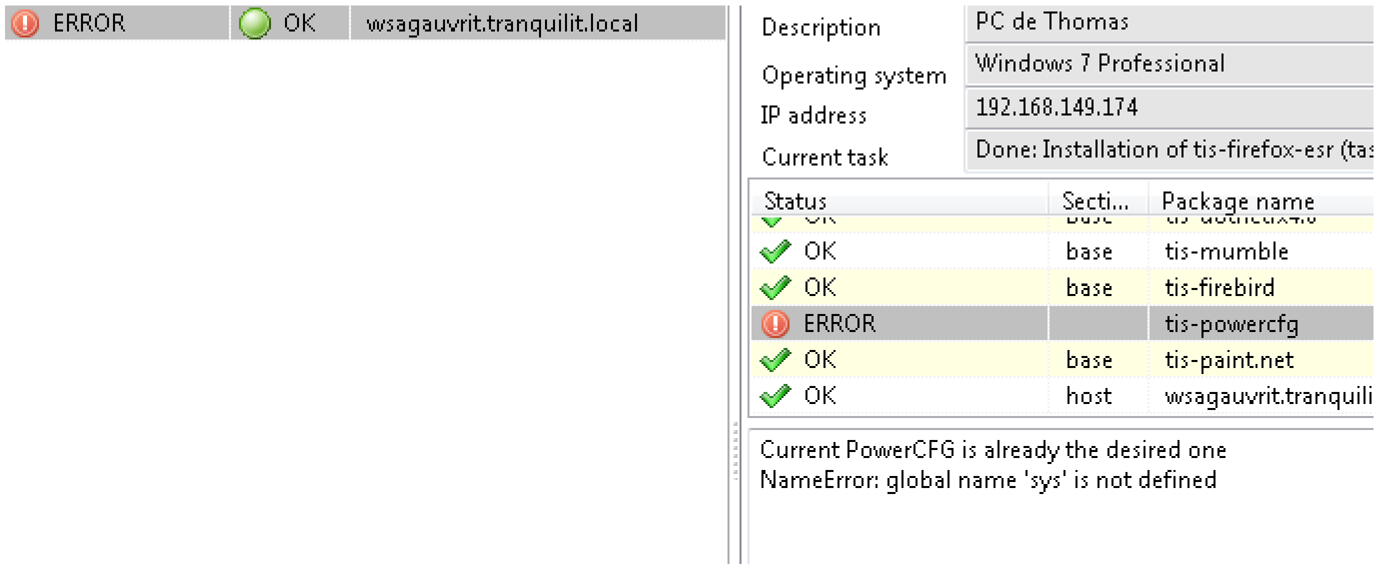
Overview | Hardware inventory | Software inventory | Tasks

Name	WSMANAGE-DEMO	Manufacturer	Xen
Description	administrateur demo	Model	HVM domU
Operating system	Windows 7 Professional	Last seen	2018-03-09 17:11
IP address	192.168.149.149	Logged in users	admin
Last task	Running: Installation de test-notepad	UUID	0AFBC5F4-46A8-3AC1-AE22-C71F5C

	Package name	Version	Install date	Section
✓	3158113b-dc19-48d0-83da-e0f9...	11	2018-03-01 14:42	
+	0AFBC5F4-46A8-3AC1-AE22-C7...			
+	test-notepadplusplus (=7.5.5-3)			
+	test-vlc (=3.0.1-4)			

Fig. 96: Host summary

When a package returns a status **ERROR**, click on it to show the details of the error. Errors are print messages in the `setup.py` of your packages.



ERROR	OK	wsagauvrit.tranquilit.local

Description	PC de Thomas
Operating system	Windows 7 Professional
IP address	192.168.149.174
Current task	Done: Installation of tis-firefox-esr (tas

Status	Secti...	Package name
✓ OK	base	tis-mumble
✓ OK	base	tis-firebird
ERROR		tis-powercfg
✓ OK	base	tis-paint.net
✓ OK	host	wsagauvrit.tranquili

```
Current PowerCFG is already the desired one
NameError: global name 'sys' is not defined
```

Fig. 97: Error detail

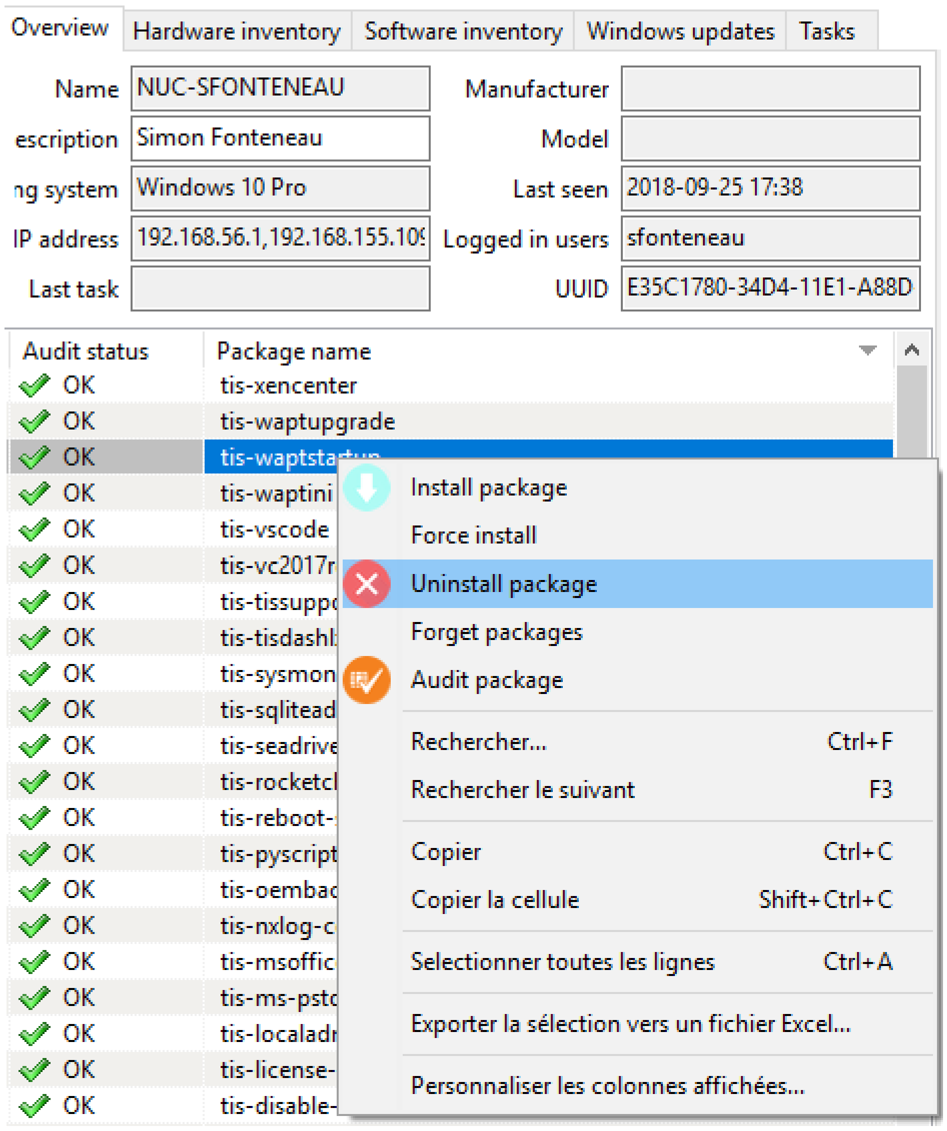
Acting on packages installed on a host

Hint:

- multiple selection of packages is possible;
- the host must be seen by the WAPT Server when the action is launched;
- if several hosts are selected, the action will be launched on all selected hosts;

Table 18: Acting on packages installed on a specific host

Action	Description
Install a package	installs the selected package on selected hosts
Force a package	forces the re-installation of a selected package on selected hosts
Remove a package	removes the selected package from the selected hosts
Forget a package	tells the selected hosts not to use WAPT for managing the selected package
Audit the package	triggers an audit on the selected package



Name	NUC-SFONTENEAU	Manufacturer	
escription	Simon Fonteneau	Model	
ng system	Windows 10 Pro	Last seen	2018-09-25 17:38
IP address	192.168.56.1,192.168.155.10!	Logged in users	sfonteneau
Last task		UUID	E35C1780-34D4-11E1-A88D

Audit status	Package name
OK	tis-xencenter
OK	tis-waptupgrade
OK	tis-waptstartup
OK	tis-waptini
OK	tis-vscode
OK	tis-vc2017r
OK	tis-tissuppc
OK	tis-tisdashb
OK	tis-sysmon
OK	tis-sqlitead
OK	tis-seadrive
OK	tis-rocketcl
OK	tis-reboot-
OK	tis-pyscript
OK	tis-oembac
OK	tis-nxlog-c
OK	tis-msoffic
OK	tis-ms-pstc
OK	tis-localadr
OK	tis-license-
OK	tis-disable-

Fig. 98: Possible actions for WAPT packages

Hardware inventory tab

Information displayed by default in the *Hardware inventory* tab is:

- information on the host's hardware components;
- some information about the host;
- some information on the status of WAPT;

A *Filter* box allows to search for hosts.

Hint: Filters work with regular expression.

To add a column in the grid, drag and drop a hardware property from the *Hardware inventory* grid to the main grid.

Example: in *hosts*, drag and drop *physical_memory* in the left panel, and the column *physical_memory* appears in the main grid.

Software inventory tab

Common information displayed in the *Sofware inventory* tab is:

- *maker*
- *software name*
- *software version*;
- *installation date*;
- *uninstall key*;
- *uninstall string*;

Windows update tab

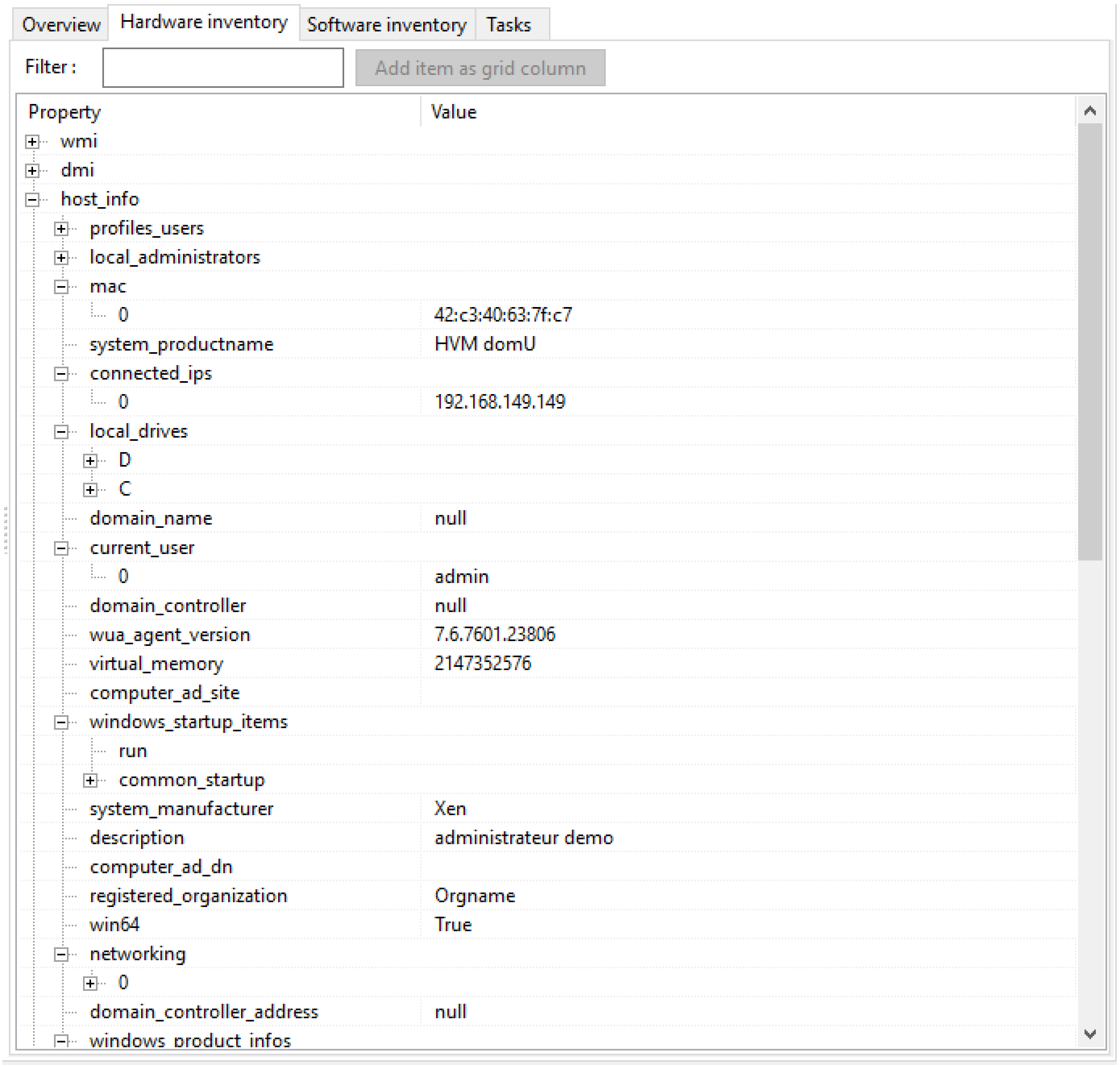
Information displayed in the *Hardware inventory* tab is:

- Windows update agent version;
- date of the last Windows update scan;
- duration of the last scan;
- WAPTWUA status;
- date of the last version of `wsusscn2.cab` processed by WAPT;
- status of WAPTWUA Enabled (True/ False);

  The grid then lists Windows cab files that have been installed or that are pending installation.

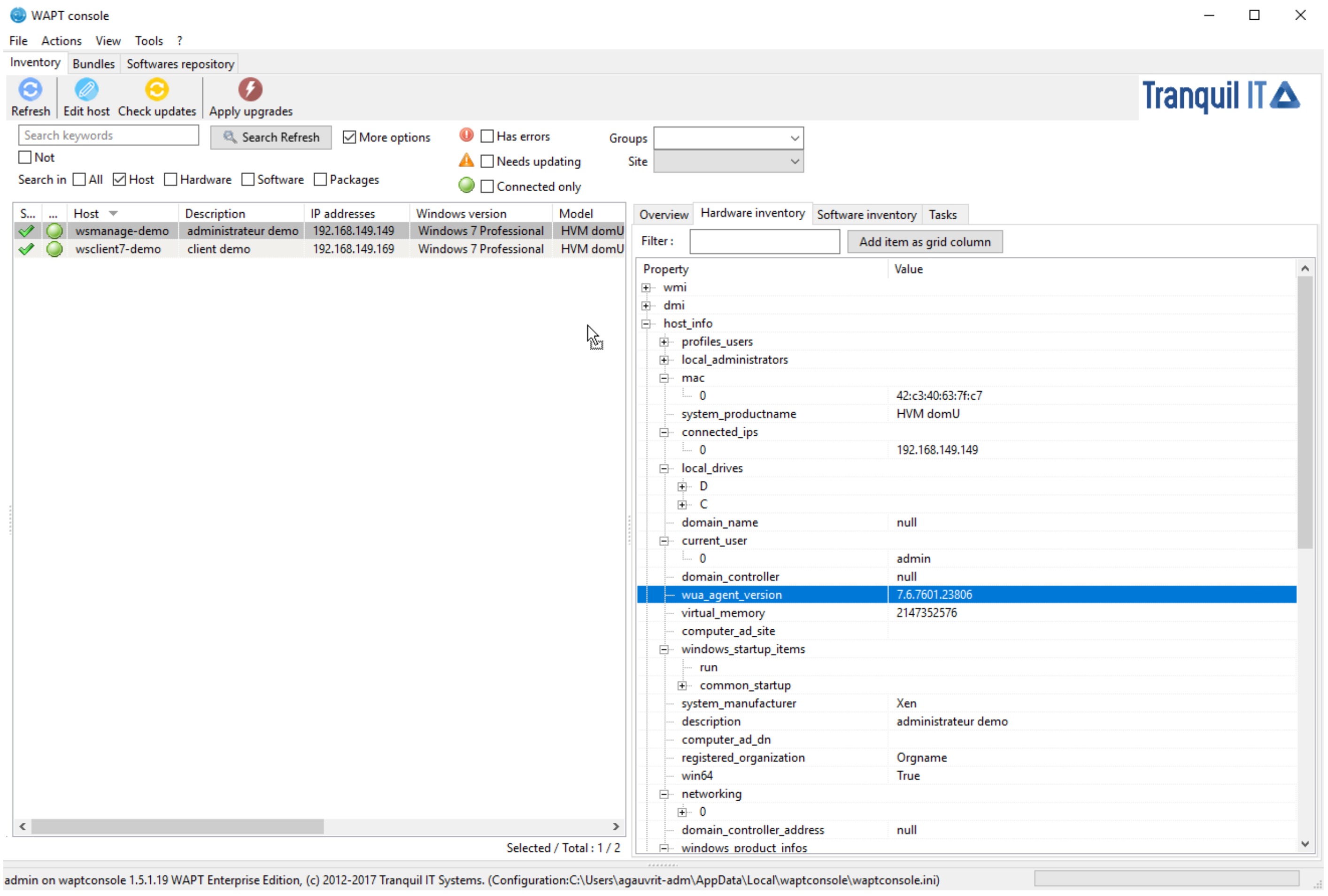
Information displayed in the *Windows Updates* tab are:

- *Status*;
- *Product*;
- *Update ID*;



Property	Value
wmi	
dmi	
host_info	
profiles_users	
local_administrators	
mac	
0	42:c3:40:63:7f:c7
system_productname	HVM domU
connected_ips	
0	192.168.149.149
local_drives	
D	
C	
domain_name	null
current_user	
0	admin
domain_controller	null
wua_agent_version	7.6.7601.23806
virtual_memory	2147352576
computer_ad_site	
windows_startup_items	
run	
common_startup	
system_manufacturer	Xen
description	administrateur demo
computer_ad_dn	
registered_organization	Orgname
win64	True
networking	
0	
domain_controller_address	null
windows_product_infos	

Fig. 99: Host hardware inventory



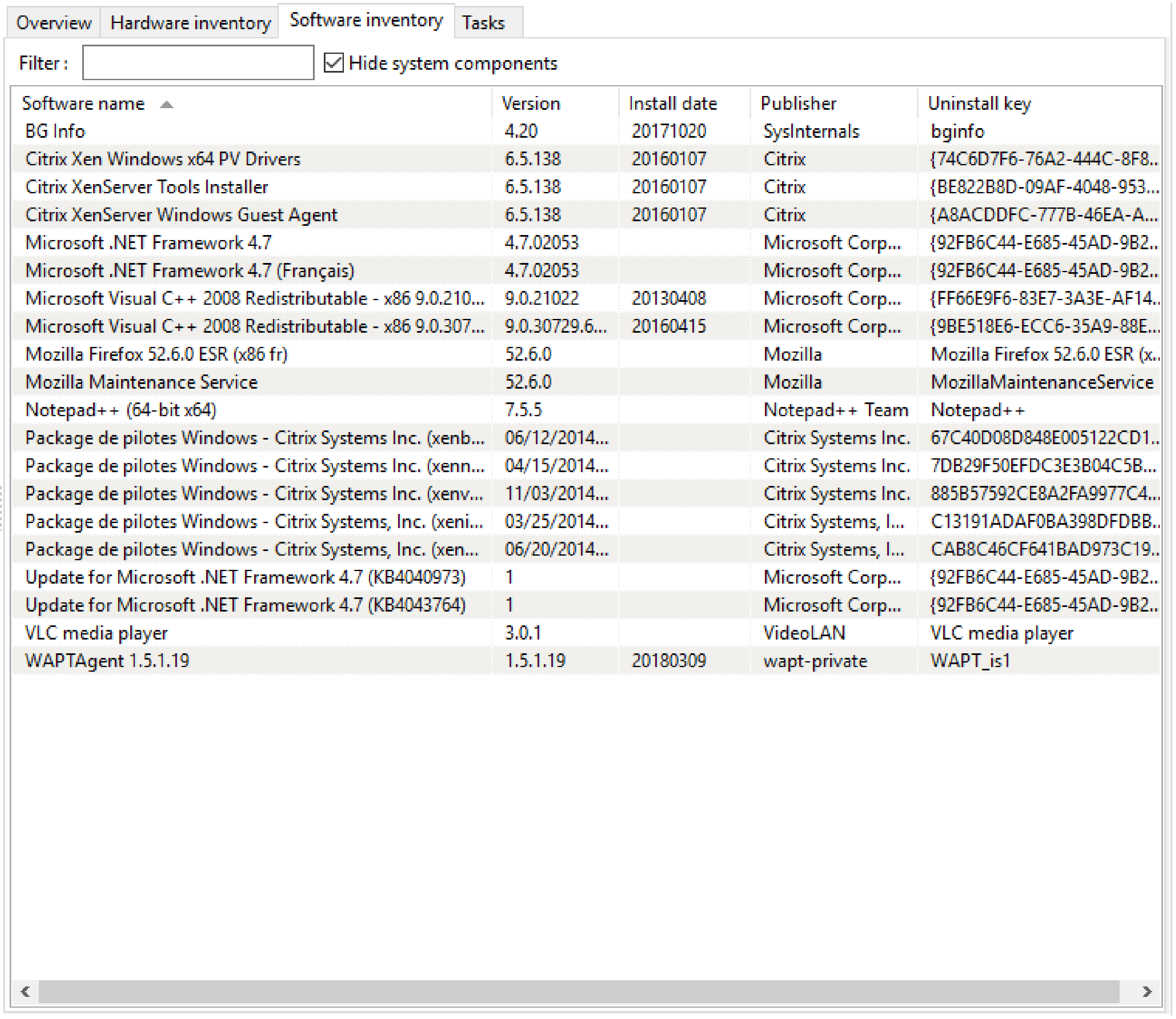
S...	...	Host	Description	IP addresses	Windows version	Model
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Professional	HVM domU
✓	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Professional	HVM domU

Selected / Total : 1 / 2

Property	Value
wmi	
dmi	
host_info	
profiles_users	
local_administrators	
mac	
0	42:c3:40:63:7f:c7
system_productname	HVM domU
connected_ips	
0	192.168.149.149
local_drives	
D	
C	
domain_name	null
current_user	
0	admin
domain_controller	null
wua_agent_version	7.6.7601.23806
virtual_memory	2147352576
computer_ad_site	
windows_startup_items	
run	
common_startup	
system_manufacturer	Xen
description	administrateur demo
computer_ad_dn	
registered_organization	Orgname
win64	True
networking	
0	
domain_controller_address	null
windows_product_infos	

admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\waptconsole\waptconsole.ini)

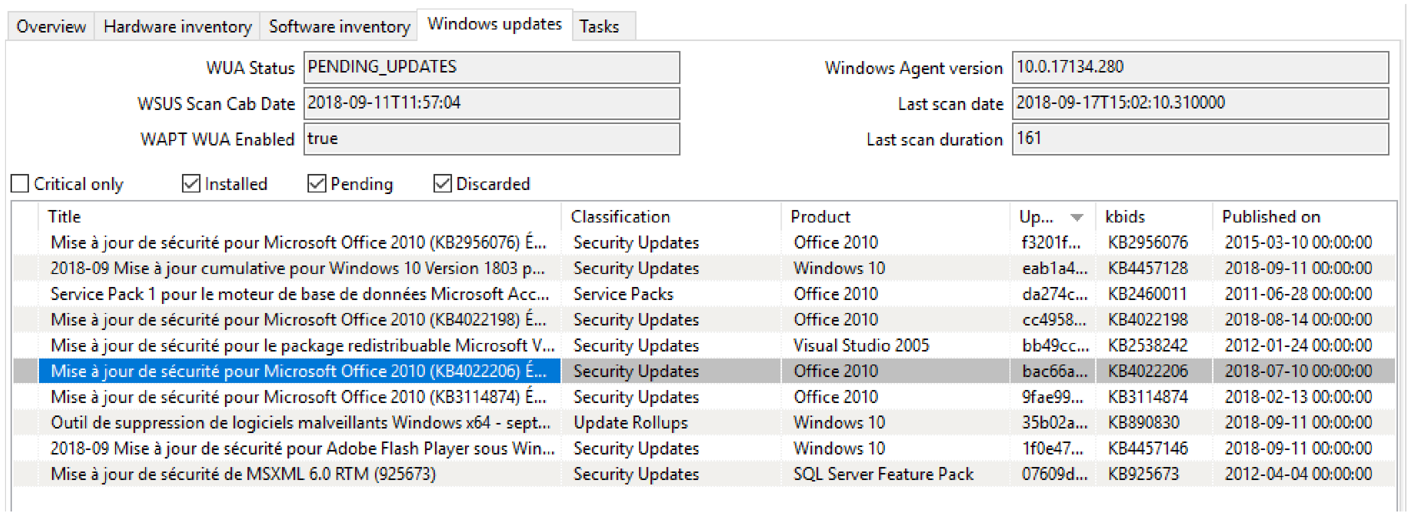
Fig. 100: Adding a criteria to the main grid of the WAPT console



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

Fig. 101: Software inventory as registered in the Windows registry of the host

- *Kbids*;
- *Published on*;
- *installation on*;
- *Severity on*;
- *Classification*;
- *Title*;
- *Download size*;



Overview | Hardware inventory | Software inventory | Windows updates | Tasks

WUA Status: PENDING_UPDATES
WSUS Scan Cab Date: 2018-09-11T11:57:04
WAPT WUA Enabled: true
Windows Agent version: 10.0.17134.280
Last scan date: 2018-09-17T15:02:10.310000
Last scan duration: 161

☐ Critical only ☑ Installed ☑ Pending ☑ Discarded

Title	Classification	Product	Up...	kbids	Published on
Mise à jour de sécurité pour Microsoft Office 2010 (KB2956076) É...	Security Updates	Office 2010	f3201f...	KB2956076	2015-03-10 00:00:00
2018-09 Mise à jour cumulative pour Windows 10 Version 1803 p...	Security Updates	Windows 10	eab1a4...	KB4457128	2018-09-11 00:00:00
Service Pack 1 pour le moteur de base de données Microsoft Acc...	Service Packs	Office 2010	da274c...	KB2460011	2011-06-28 00:00:00
Mise à jour de sécurité pour Microsoft Office 2010 (KB4022198) É...	Security Updates	Office 2010	cc4958...	KB4022198	2018-08-14 00:00:00
Mise à jour de sécurité pour le package redistribuable Microsoft V...	Security Updates	Visual Studio 2005	bb49cc...	KB2538242	2012-01-24 00:00:00
Mise à jour de sécurité pour Microsoft Office 2010 (KB4022206) É...	Security Updates	Office 2010	bac66a...	KB4022206	2018-07-10 00:00:00
Mise à jour de sécurité pour Microsoft Office 2010 (KB3114874) É...	Security Updates	Office 2010	9fae99...	KB3114874	2018-02-13 00:00:00
Outil de suppression de logiciels malveillants Windows x64 - sept...	Update Rollups	Windows 10	35b02a...	KB890830	2018-09-11 00:00:00
2018-09 Mise à jour de sécurité pour Adobe Flash Player sous Win...	Security Updates	Windows 10	1f0e47...	KB4457146	2018-09-11 00:00:00
Mise à jour de sécurité de MSXML 6.0 RTM (925673)	Security Updates	SQL Server Feature Pack	07609d...	KB925673	2012-04-04 00:00:00

Fig. 102: Inventory of Windows Updates

Task tab

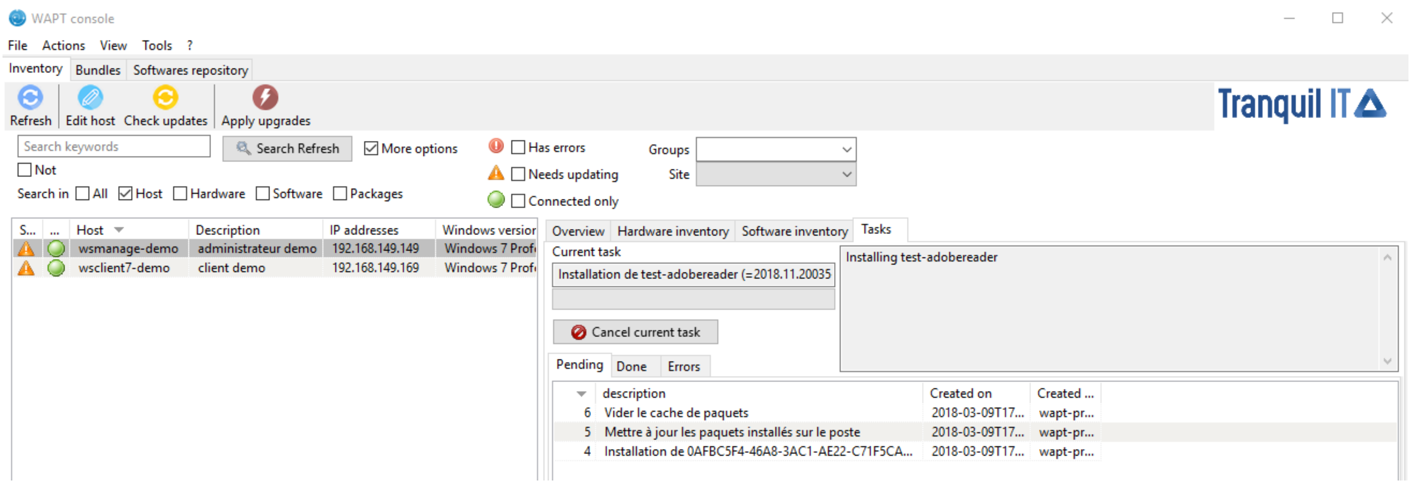
Information displayed by default in the *Tasks* tab is:

- pending tasks;
- completed tasks;
- tasks in error;

Perform a global search on all hosts

Performing global searches on all the criteria presented above is possible.

Choose the filters to check or uncheck.


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository
Refresh Edit host Check updates Apply upgrades
Tranquil IT
Search keywords
Search Refresh
Overview Hardware inventory Software inventory Tasks
Cancel current task
Pending Done Errors

[✓ More options] ☐ Not ☐ Has errors ☐ Needs updating ☐ Connected only

Groups | Site

Search in ☐ All [✓ Host] ☐ Hardware ☐ Software ☐ Packages

S...	...	Host	Description	IP addresses	Windows version
⚠	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Profe
⚠	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Profe

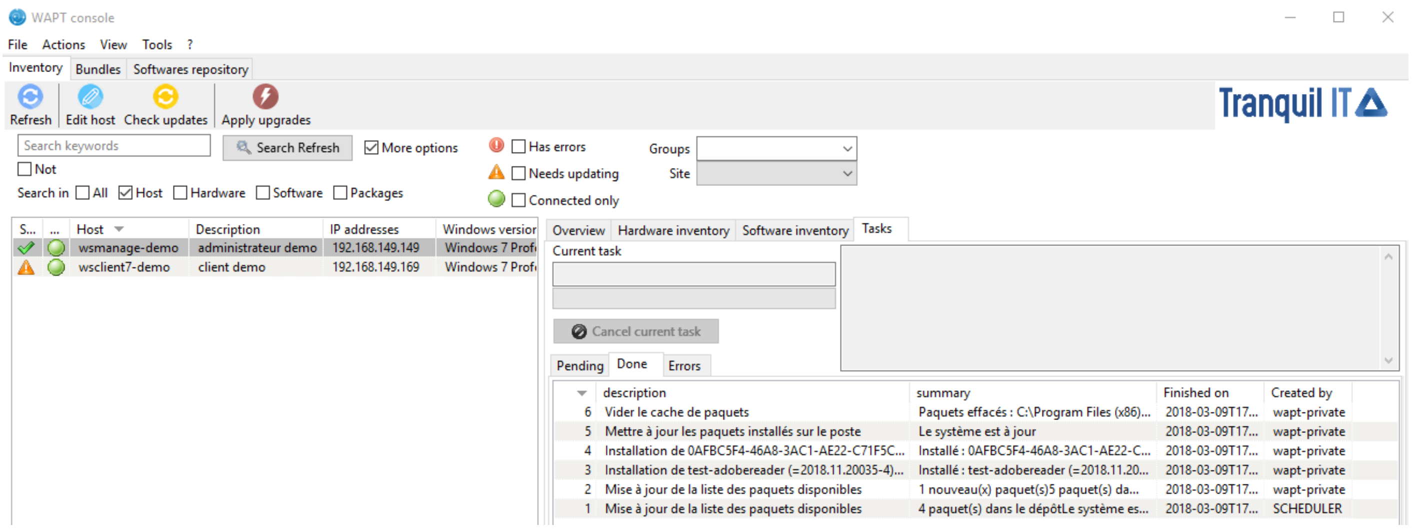
Current task

Installation de test-adobereader (=2018.11.20035

Installing test-adobereader

	description	Created on	Created ...
6	Vider le cache de paquets	2018-03-09T17...	wapt-pr...
5	Mettre à jour les paquets installés sur le poste	2018-03-09T17...	wapt-pr...
4	Installation de 0AFBC5F4-46A8-3AC1-AE22-C71F5CA...	2018-03-09T17...	wapt-pr...

Fig. 103: Details of pending tasks on the host


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository
Refresh Edit host Check updates Apply upgrades
Tranquil IT
Search keywords
Search Refresh
Overview Hardware inventory Software inventory Tasks
Cancel current task
Pending Done Errors

[✓ More options] ☐ Not ☐ Has errors ☐ Needs updating ☐ Connected only

Groups | Site

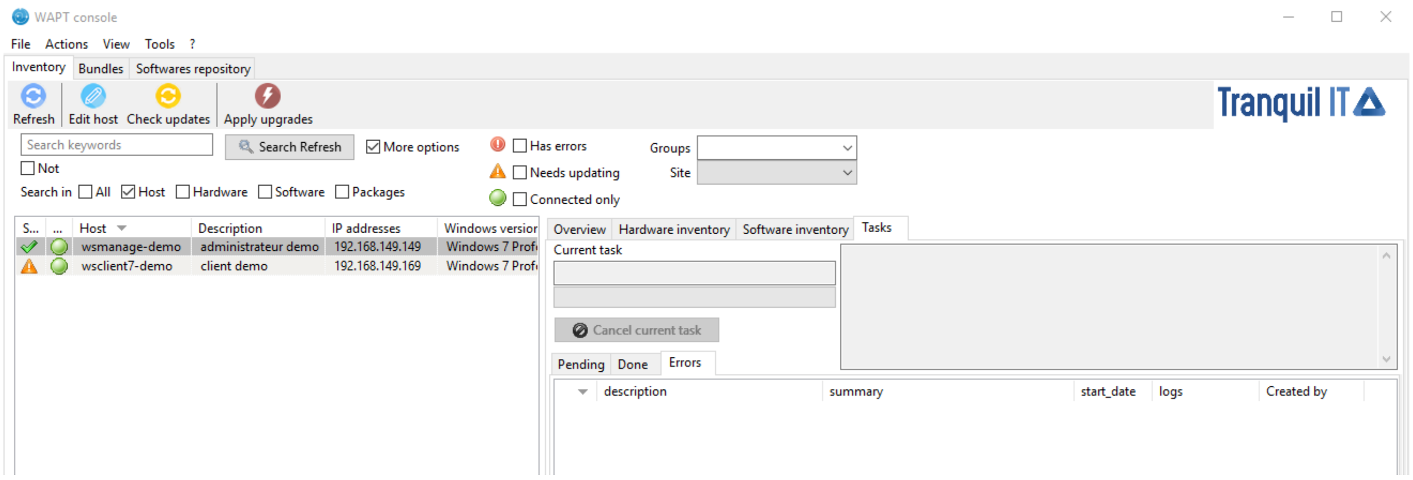
Search in ☐ All [✓ Host] ☐ Hardware ☐ Software ☐ Packages

S...	...	Host	Description	IP addresses	Windows version
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Profe
⚠	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Profe

Current task

	description	summary	Finished on	Created by
6	Vider le cache de paquets	Paquets effacés : C:\Program Files (x86)...	2018-03-09T17...	wapt-private
5	Mettre à jour les paquets installés sur le poste	Le système est à jour	2018-03-09T17...	wapt-private
4	Installation de 0AFBC5F4-46A8-3AC1-AE22-C71F5C...	Installé : 0AFBC5F4-46A8-3AC1-AE22-C...	2018-03-09T17...	wapt-private
3	Installation de test-adobereader (=2018.11.20035-4)...	Installé : test-adobereader (=2018.11.20...	2018-03-09T17...	wapt-private
2	Mise à jour de la liste des paquets disponibles	1 nouveau(x) paquet(s)5 paquet(s) da...	2018-03-09T17...	wapt-private
1	Mise à jour de la liste des paquets disponibles	4 paquet(s) dans le dépôtLe système es...	2018-03-09T17...	SCHEDULER

Fig. 104: Details of completed tasks


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository
Refresh Edit host Check updates Apply upgrades
Tranquil IT
Search keywords
Search Refresh
More options
Not
Search in All Host Hardware Software Packages
Has errors
Needs updating
Connected only
Groups
Site
Overview Hardware inventory Software inventory Tasks
Current task
Cancel current task
Pending Done Errors

S...	...	Host	Description	IP addresses	Windows version
✓	●	wsmanage-demo	administrateur demo	192.168.149.149	Windows 7 Profe
⚠	●	wsclient7-demo	client demo	192.168.149.169	Windows 7 Profe

description	summary	start_date	logs	Created by

Fig. 105: Details of tasks in error

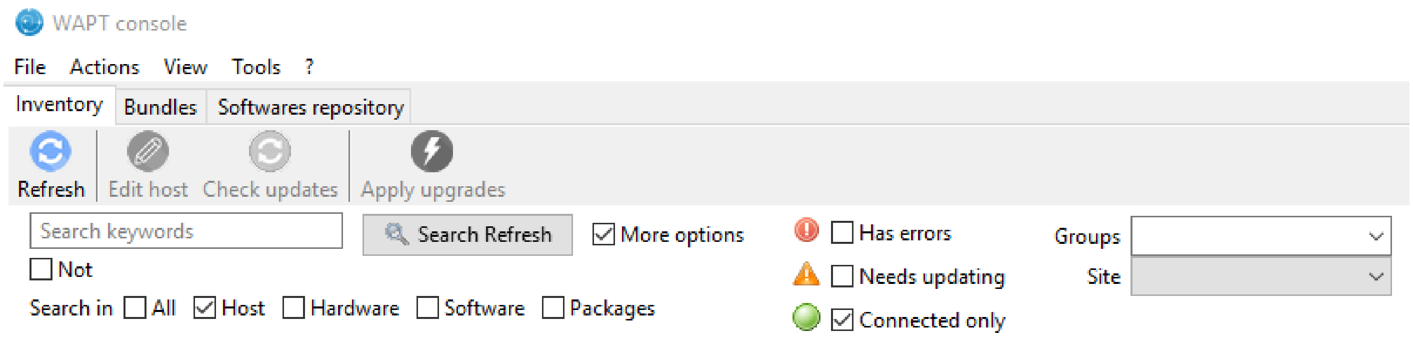

WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository
Refresh Edit host Check updates Apply upgrades
Search keywords
Search Refresh
More options
Not
Search in All Host Hardware Software Packages
Has errors
Needs updating
Connected only
Groups
Site

Fig. 106: Advanced search functionalities in the WAPT console

Table 19: Choice of filters

Possible options	Description
Host	*Host* section in the *Hardware inventory* tab when a host is selected
Hardware	*DMI* section in the *Hardware inventory* tab when a host is selected
Software	*Software inventory* section when a host is selected
Package	List of packages installed on the selected hosts
Have errors	Search only for hosts for which a task has not finished correctly
Needing upgrades	Search only for hosts needing upgrades
Group selection	Filter hosts based on their membership/ dependency to a group package

Hint: Filters work with regular expression.

Do a search based on a WAPT package

In the *Private repository*, select the package and then click on *Show Hosts*.

The grid will display the hosts on which the package is installed. Note that the filter is only active on the *Package* attribute of the selected package.

The different columns display information about the packages installed on the machine (e.g. *package version*, *package status*, *audit status*, *installation date*, *architecture*).

You can also add the columns *Log install* and *Last Audit Output* to display at a glance the installation and audit logs.

Creating a group package

Group packages allows you to create a package containing other packages to be affected as a dependency to a host.

To create a group of packages, go to the *Bundles* tab:

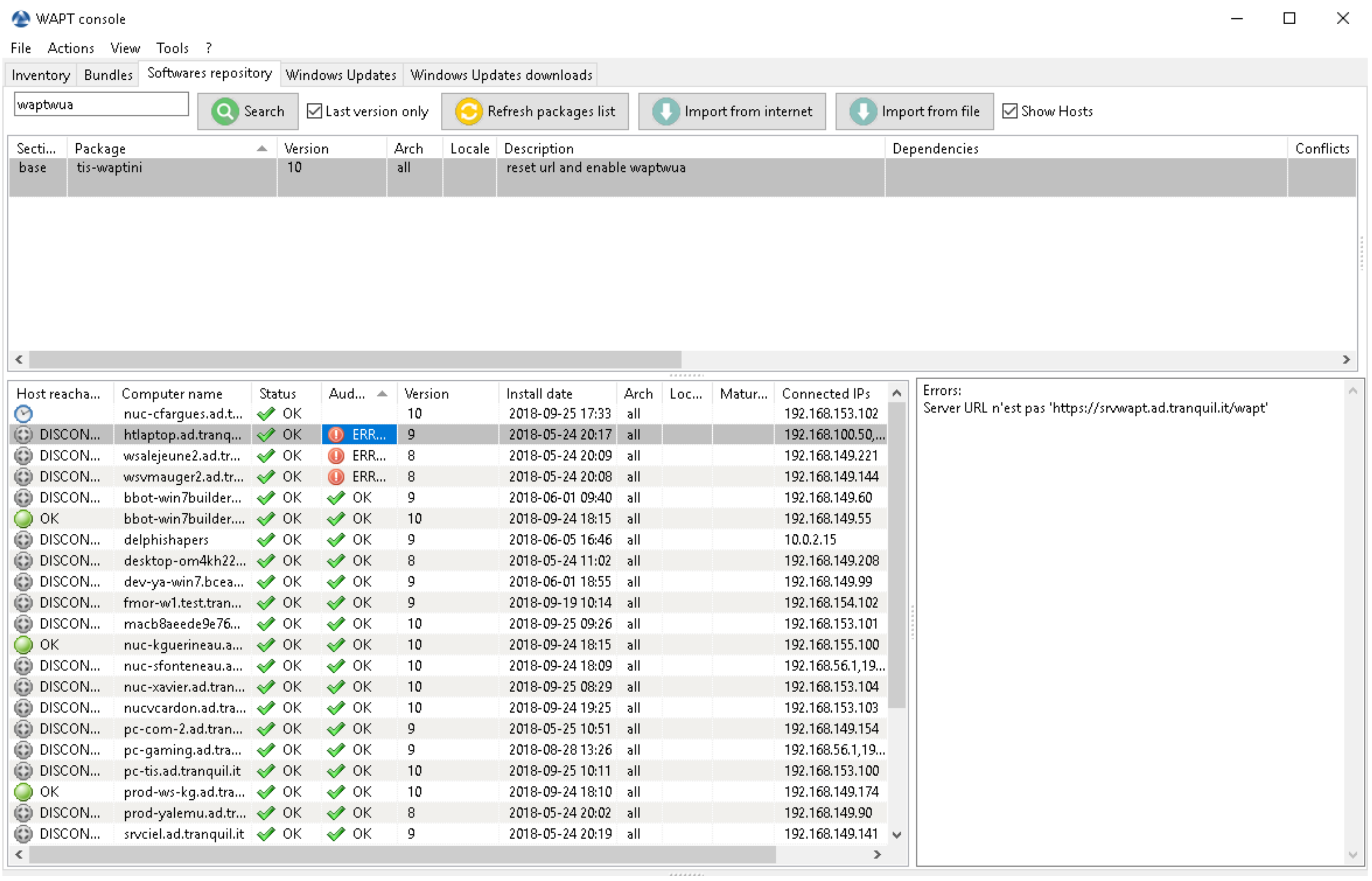
- click on *New bundle*;
- give a name to the *group* package;

Hint: If you name a group package with the same name as an Active Directory security group (Microsoft or Samba-AD), a member of the Active Directory group can be automatically added to the WAPT group package.

- fill in the description, add packages to the group package by dragging and dropping them or by Right-clicking on the package name, and adding it to the bundle;
- click on *Save* to save the bundle;

Hint: To uninstall a package, it is possible to add banned packages to a bundle.

In the *Software Repository* tab, the list of packages currently available in the WAPT repository appears. By default, the console will only show the latest version of packages.

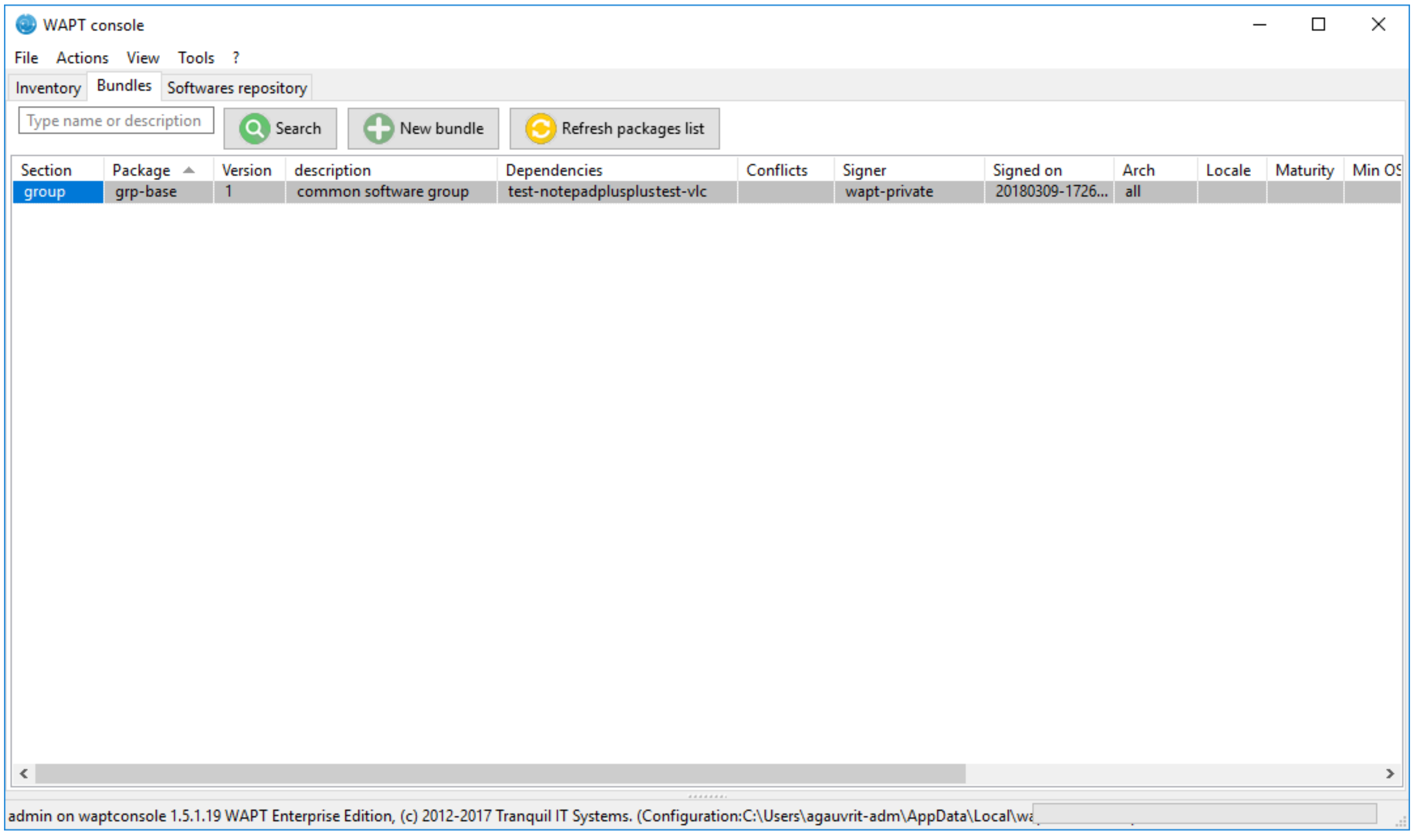


Secti...	Package	Version	Arch	Locale	Description	Dependencies	Conflicts
base	tis-waptini	10	all		reset url and enable waptwua		

Host reacha...	Computer name	Status	Aud...	Version	Install date	Arch	Loc...	Matur...	Connected IPs
	nuc-cfargues.ad.t...	OK		10	2018-09-25 17:33	all			192.168.153.102
DISCON...	htlaptop.ad.tranq...	OK	ERR...	9	2018-05-24 20:17	all			192.168.100.50,...
DISCON...	wsalejeune2.ad.tr...	OK	ERR...	8	2018-05-24 20:09	all			192.168.149.221
DISCON...	wsvmauger2.ad.tr...	OK	ERR...	8	2018-05-24 20:08	all			192.168.149.144
DISCON...	bbot-win7builder...	OK	OK	9	2018-06-01 09:40	all			192.168.149.60
OK	bbot-win7builder....	OK	OK	10	2018-09-24 18:15	all			192.168.149.55
DISCON...	delphishapers	OK	OK	9	2018-06-05 16:46	all			10.0.2.15
DISCON...	desktop-om4kh22...	OK	OK	8	2018-05-24 11:02	all			192.168.149.208
DISCON...	dev-ya-win7.bcea...	OK	OK	9	2018-06-01 18:55	all			192.168.149.99
DISCON...	fmor-w1.test.tran...	OK	OK	9	2018-09-19 10:14	all			192.168.154.102
DISCON...	macb8aeede9e76...	OK	OK	10	2018-09-25 09:26	all			192.168.153.101
OK	nuc-kguerineau.a...	OK	OK	10	2018-09-24 18:15	all			192.168.155.100
DISCON...	nuc-sfonteneau.a...	OK	OK	10	2018-09-24 18:09	all			192.168.56.1,19...
DISCON...	nuc-xavier.ad.tran...	OK	OK	10	2018-09-25 08:29	all			192.168.153.104
DISCON...	nucvcardon.ad.tra...	OK	OK	10	2018-09-24 19:25	all			192.168.153.103
DISCON...	pc-com-2.ad.tran...	OK	OK	9	2018-05-25 10:51	all			192.168.149.154
DISCON...	pc-gaming.ad.tra...	OK	OK	9	2018-08-28 13:26	all			192.168.56.1,19...
DISCON...	pc-tis.ad.tranquil.it	OK	OK	10	2018-09-25 10:11	all			192.168.153.100
OK	prod-ws-kg.ad.tra...	OK	OK	10	2018-09-24 18:10	all			192.168.149.174
DISCON...	prod-yalemu.ad.tr...	OK	OK	8	2018-05-24 20:02	all			192.168.149.90
DISCON...	srvciel.ad.tranquil.it	OK	OK	9	2018-05-24 20:19	all			192.168.149.141

Errors:
Server URL n'est pas 'https://srvwapt.ad.tranquil.it/wapt'

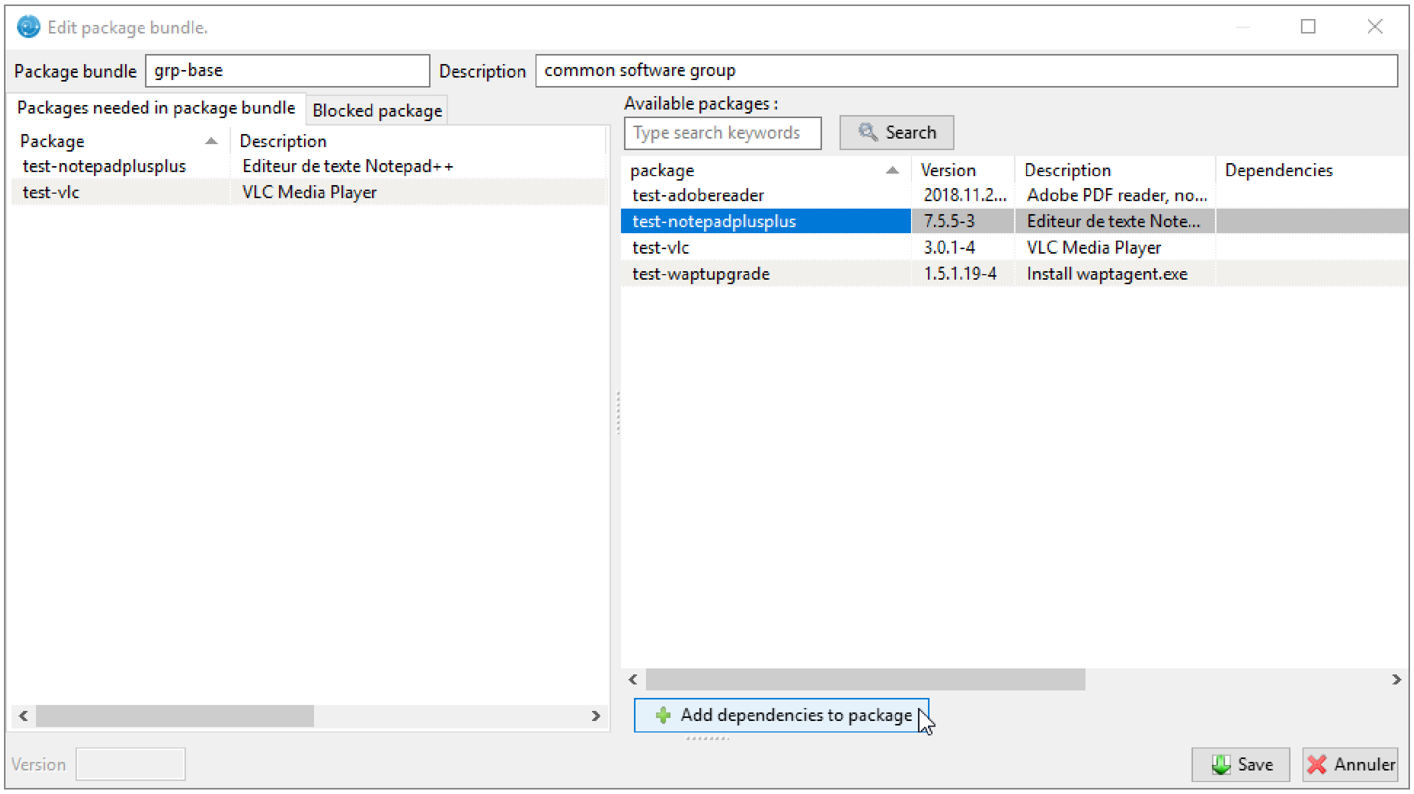
Fig. 107: Filter by package



Section	Package	Version	description	Dependencies	Conflicts	Signer	Signed on	Arch	Locale	Maturity	Min OS
group	grp-base	1	common software group	test-notepadplusplustest-vlc		wapt-private	20180309-1726...	all			

admin on waptconsole 1.5.1.19 WAPT Enterprise Edition, (c) 2012-2017 Tranquil IT Systems. (Configuration:C:\Users\agauvrit-adm\AppData\Local\wa

Fig. 108: Package group grid



Edit package bundle.

Package bundle: grp-base

Description: common software group

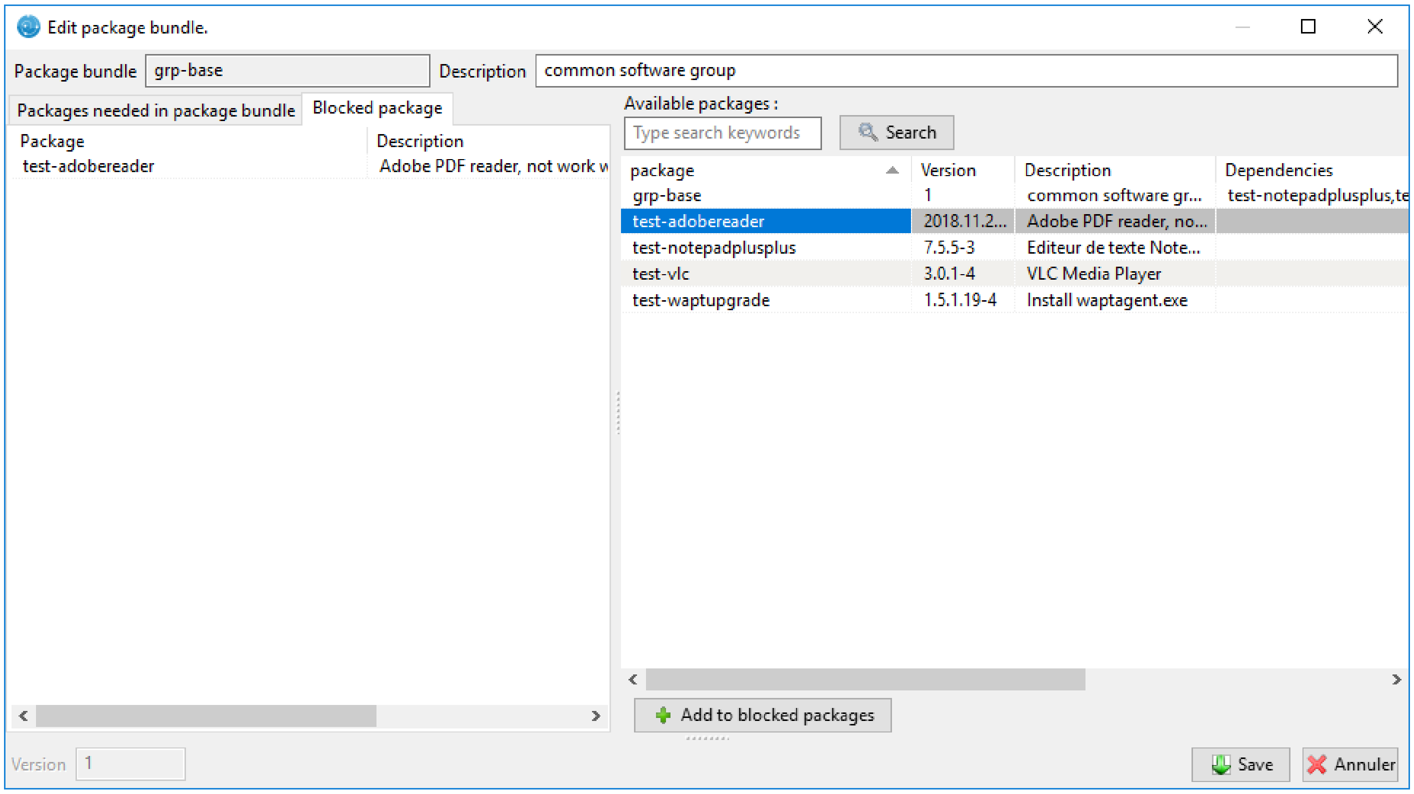
Packages needed in package bundle | Blocked package

Package	Description
test-notepadplusplus	Editeur de texte Notepad++
test-vlc	VLC Media Player

Available packages :

package	Version	Description	Dependencies
test-adobereader	2018.11.2...	Adobe PDF reader, no...	
test-notepadplusplus	7.5.5-3	Editeur de texte Note...	
test-vlc	3.0.1-4	VLC Media Player	
test-waptupgrade	1.5.1.19-4	Install waptagent.exe	

Fig. 109: Creating a group package


Edit package bundle.

Package bundle: grp-base
Description: common software group

Package	Description
test-adobereader	Adobe PDF reader, not work w

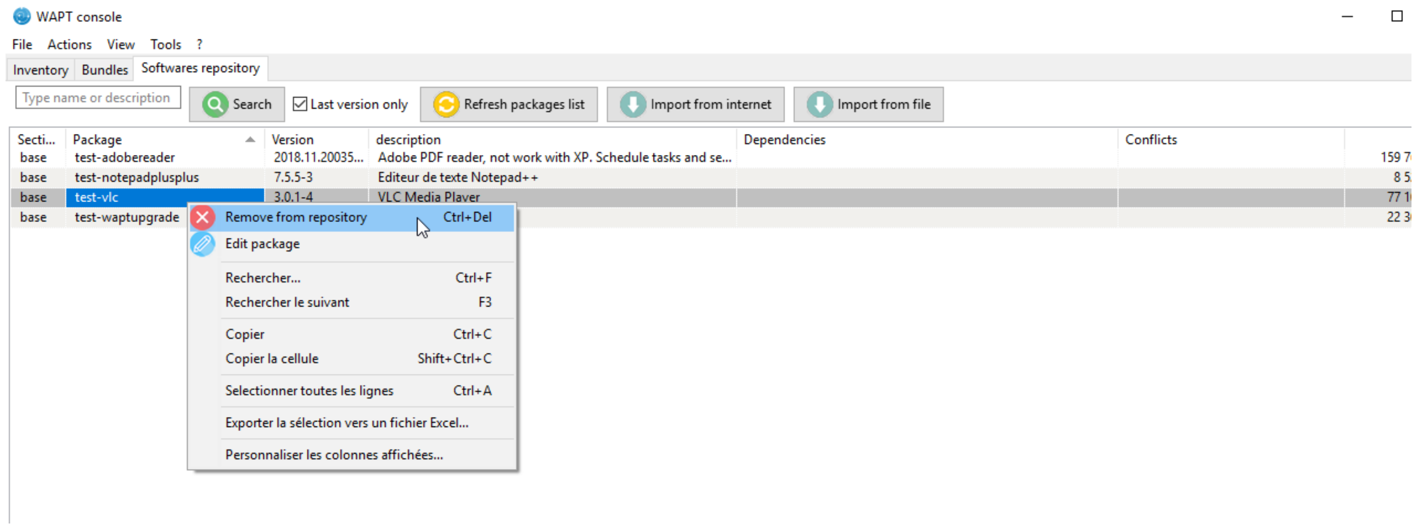
Available packages :

package	Version	Description	Dependencies
grp-base	1	common software gr...	test-notepadplusplus,te
test-adobereader	2018.11.2...	Adobe PDF reader, no...	
test-notepadplusplus	7.5.5-3	Editeur de texte Note...	
test-vlc	3.0.1-4	VLC Media Player	
test-waptupgrade	1.5.1.19-4	Install waptagent.exe	

Version: 1

Fig. 110: Forbid a package

To display all package versions, untick *Last version only*. To delete a package from the repository, *Right-click* → *Remove from repository*.


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository
Type name or description
Search
Last version only
Refresh packages list
Import from internet
Import from file
Remove from repository Ctrl+Del
Edit package
Rechercher... Ctrl+F
Rechercher le suivant F3
Copier Ctrl+C
Copier la cellule Shift+Ctrl+C
Selectionner toutes les lignes Ctrl+A
Exporter la sélection vers un fichier Excel...
Personnaliser les colonnes affichées...

Secti...	Package	Version	description	Dependencies	Conflicts	
base	test-adobereader	2018.11.20035...	Adobe PDF reader, not work with XP. Schedule tasks and se...			159 7
base	test-notepadplusplus	7.5.5-3	Editeur de texte Notepad++			8 5
base	test-vlc	3.0.1-4	VLC Media Player			77 1
base	test-waptupgrade					22 3

Fig. 111: Remove a package

To edit a package, *Right-click* → *Edit package*, the package will be downloaded locally in **the base package development directory** set in the console settings.

Make changes to the package as wanted, rebuild the package and upload it back to the repository. Once your package has uploaded, refresh the package list using the *Refresh package list* button or by pressing `F5` on your keyboard.

A search bar is also available to filter packages.

Hint: With the *section* drop-down menu, you can choose to create a profile package rather than a *group* package.

Cleaning the local cache from the WAPT console

When importing a package from the Internet, the WAPT console downloads the package in `%appdata%localwaptconsolecache`.

To clean the cache and free up disk space, click on *Tools* → *Clean local cache*.

Changing the password of the WAPT Server

To change the WAPT Server password, click on *Tools* → *Update password*, fill in the old password and add a new one.

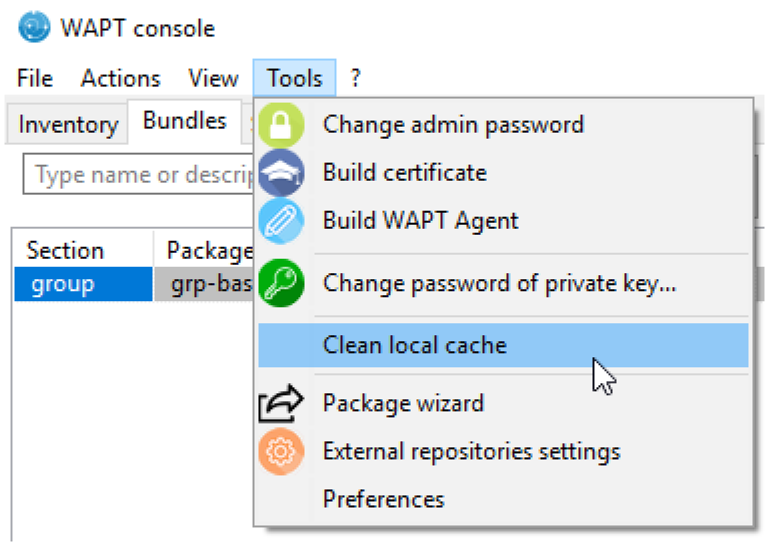


Fig. 112: Cleaning up the local cache

Making changes to the WAPT console preferences

To make changes to the console settings, go to *Tools* → *Preferences*.

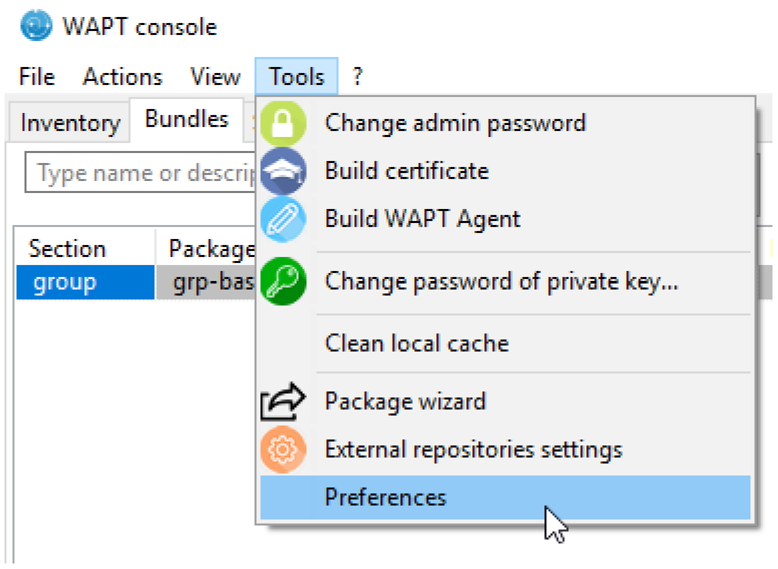
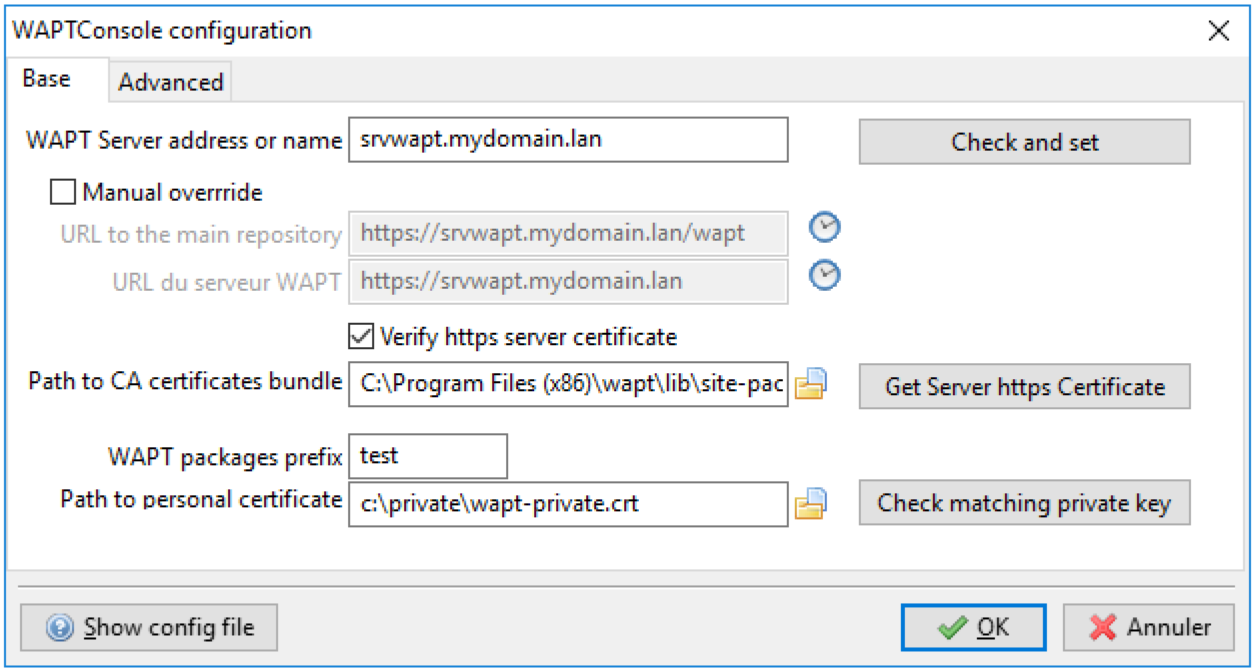


Fig. 113: Configuration options for the WAPT console

- *Basic* tab for basic options;


WAPTConsole configuration
Base
Advanced
Check and set
Get Server https Certificate
Check matching private key
Show config file
OK
Annuler

WAPT Server address or name: srvwapt.mydomain.lan

Manual override: ☐

URL to the main repository: https://srvwapt.mydomain.lan/wapt

URL du serveur WAPT: https://srvwapt.mydomain.lan

☑ Verify https server certificate

Path to CA certificates bundle: C:\Program Files (x86)\wapt\lib\site-pac

WAPT packages prefix: test

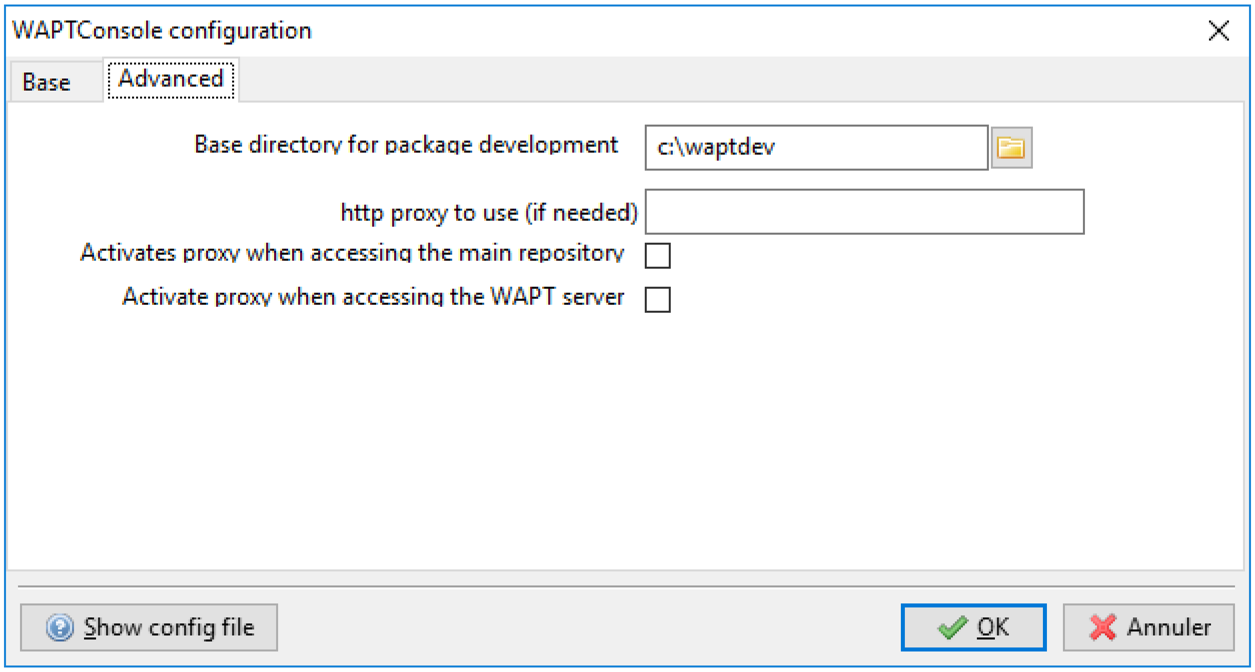
Path to personal certificate: c:\private\wapt-private.crt

Fig. 114: Configuration options for the WAPT console

Arguments	Description	Example
WAPT Server IP address	URL of the WAPT Server	*srvwapt.mydomain.lan*
URL of the main WAPT repository	URL of the main WAPT repository (only if *Specify manually* is checked)	http://srvwapt.mydomain.lan/wapt/
URL of the WAPT Server	URL of the WAPT Server (only if *Specify manually* is checked)	*https://srvwapt.mydomain.lan/*
Verifying the HTTPS certificate	Indicates whether the HTTPS certificate must be verified	yes
Path to the bundle of certificates	Path to the bundle of certificates that will allow certificates to be verified	Visit *the documentation on activating HTTPS verification*
Prefix to use when creating packages. Ex: *tis* or *demo*	Prefix that is given to packages during replication.	prefix
Path to the Administrator's personal certificate	Path to the certificate associated with the private key used to sign packages	`C:\private\mykey.crt`

Hint: The button *Get the server certificate* downloads the WAPT Server HTTPS certificate to `WAPT\ssl\server` and tells the WAPT console to verify HTTPS connections using that bundle of certificates. The method is called **Certificate pinning**. Before downloading the HTTPS certificate, you must be sure that you are connecting with the right server.

- *Advanced* tab for advanced options;



Base directory for package development: c:\waptdev
http proxy to use (if needed):
Activates proxy when accessing the main repository ☐
Activate proxy when accessing the WAPT server ☐

Fig. 115: Configuration options for the WAPT console

Table 20: :header-rows: 1

Arguments	Description	Example
Path to waptdev folder	Indicates the path to the directory for storing packages being developed	`C:\waptdev`
HTTP proxy to use	Indicates a proxy server to be used by the WAPT console when accessing the WAPT repository or the WAPT Server	*http://srvproxy.mydomain.local:8080*
Activating the proxy	Activate proxy settings for connecting to the WAPT repository or the WAPT Server	False

To make changes to console settings, go to *Tools* → *Preferences*.

Table 21: :header-rows: 1

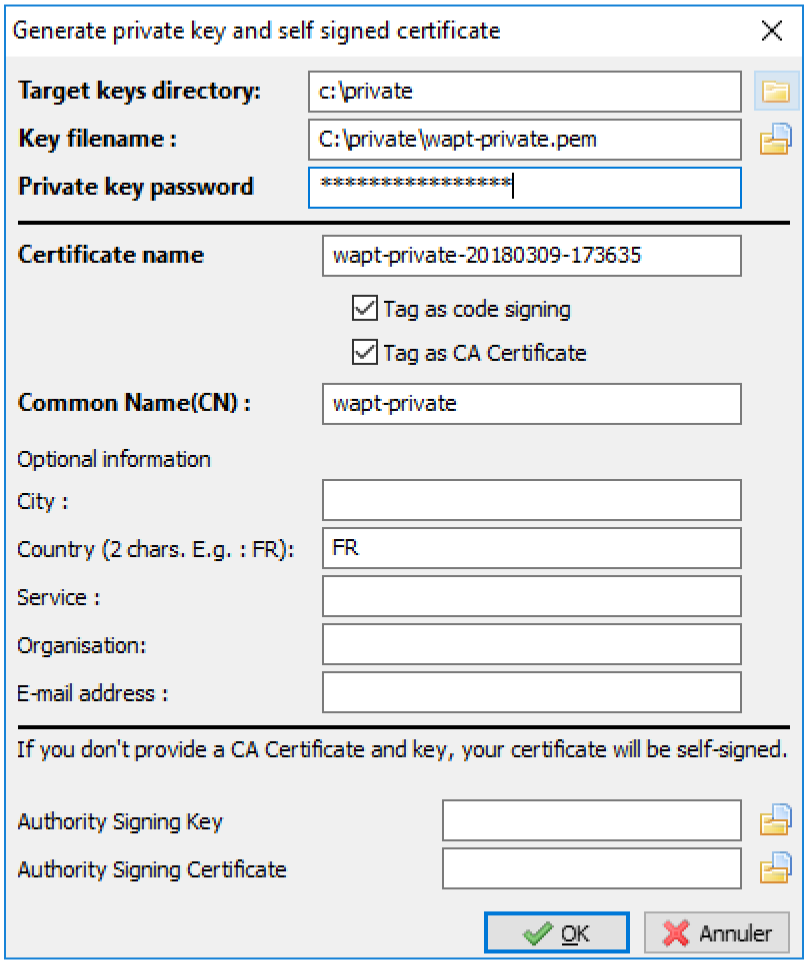
Arguments	Description	Example
Maximum number of hosts to be displayed in the console	Indicates the maximum number of hosts to be displayed in the WAPT console, so to optimize the behavior of the console.	2000
Language	Selects the language for the WAPT console	English
Showing debug information in the WAPT console	Shows debug information in the WAPT console	True
Allow third-party tools in the contextual menus of the hosts	TODO	True
Activate administration functionalities	TODO	True
Hide unavailable options	TODO	True

Generating a new public certificate

New in version 1.3.12.13.

Generating a new public certificate allows to actualize an existing public certificate without having to regenerate a public key/ certificate pair.

For that purpose, go to *Tools* → *Generate a new certificate*



Generate private key and self signed certificate

Target keys directory:	c:\private
Key filename :	C:\private\wapt-private.pem
Private key password	****************
Certificate name	wapt-private-20180309-173635
	✓ Tag as code signing
	✓ Tag as CA Certificate
Common Name(CN) :	wapt-private
Optional information	
City :	
Country (2 chars. E.g. : FR):	FR
Service :	
Organisation:	
E-mail address :	

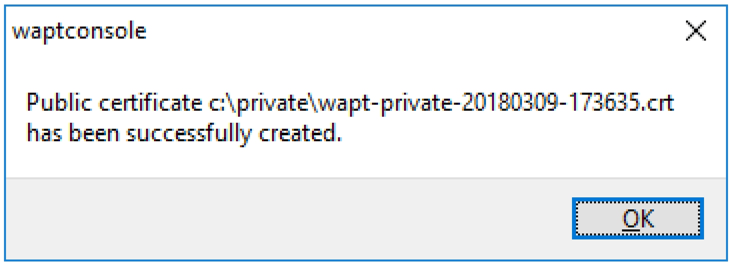
If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key	
Authority Signing Certificate	

Fig. 116: Generate a self-signed certificate

The private key is recovered from current settings, change the **Common Name** and regenerate a new certificate.

The old certificate will be invalidated.


waptconsole

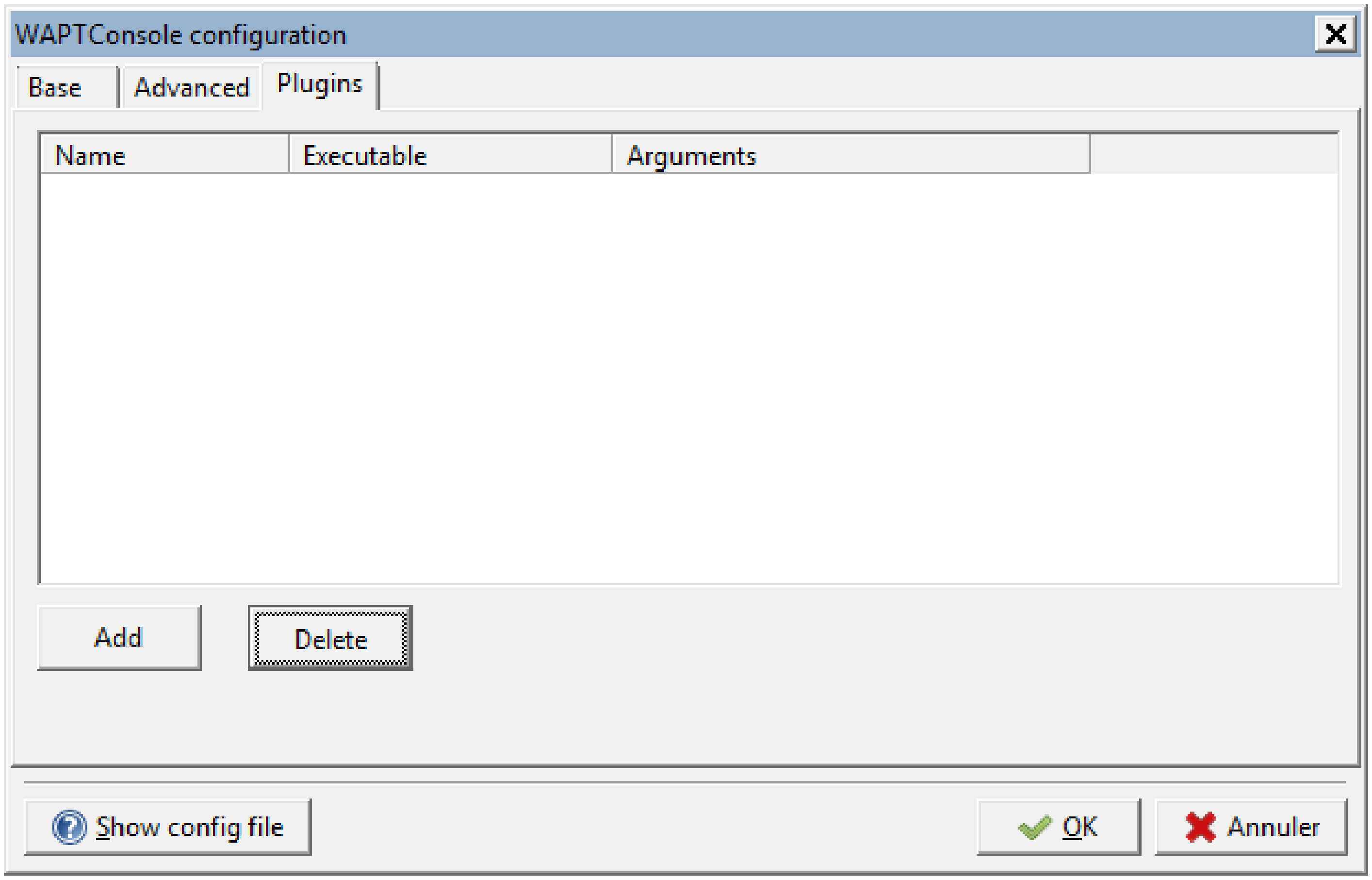
Public certificate c:\private\wapt-private-20180309-173635.crt has been successfully created.

Fig. 117: New public certificate has been created

Adding plugins in the Console

New in version 1.7.

To add plugins, go to *Tools* → *preference* → *plugins Tab*.

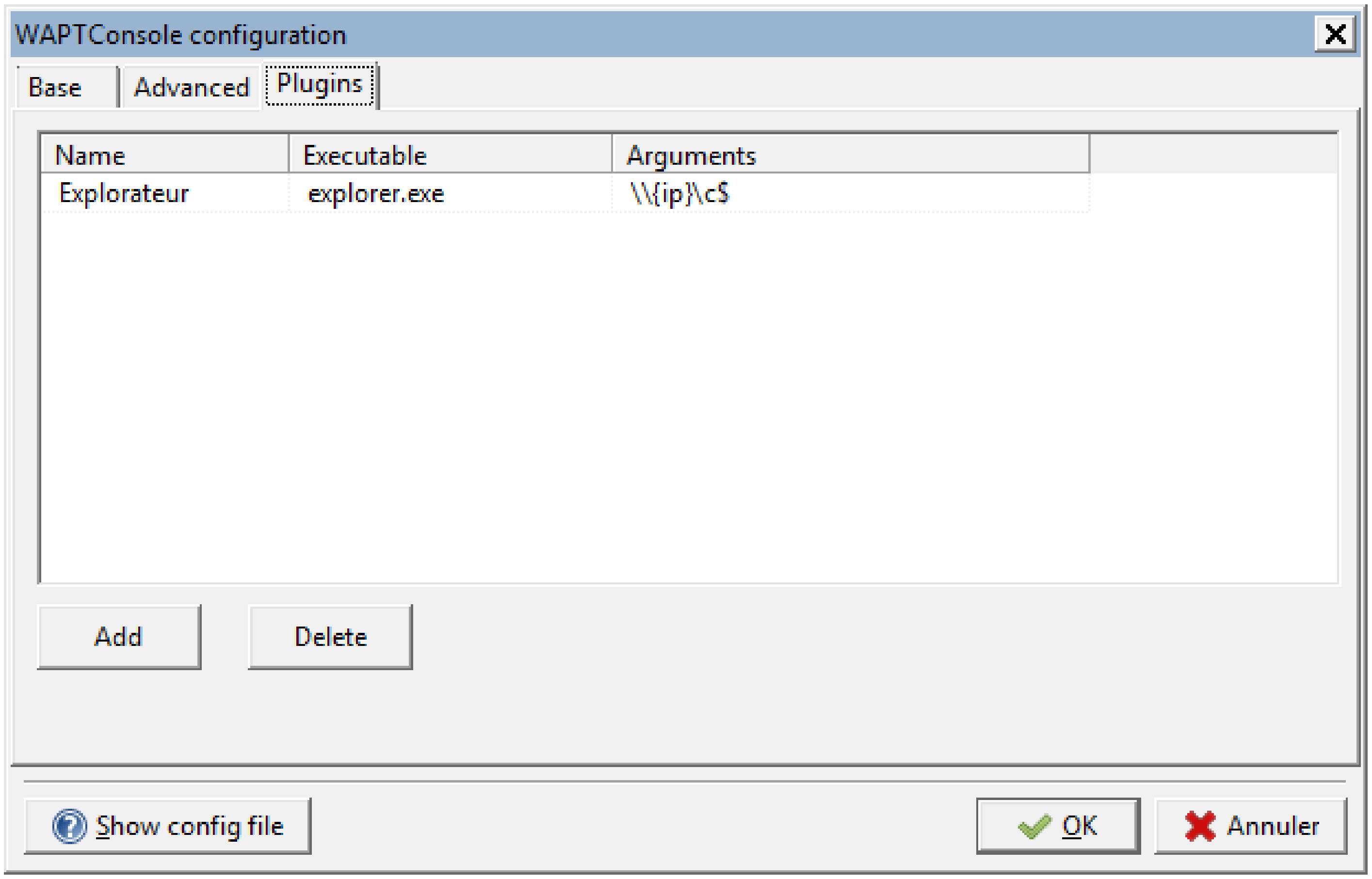


Click *Add* to add plugins, then edit the corresponding columns

Table 22: :header-rows: 1

Column	Description
Name	Name that will appear in the menu
Executable	Path of the executable that will be executed after the click
Arguments	Arguments passed to the executable. Some variables can be used like {ip}, {uuid} or {computer_fqdn}

Plugins will then appear in the menu:



Name	Executable	Arguments
Explorateur	explorer.exe	\\{ip}\c$

6.7.3 Using WAPTtray.

wapttray is a utility working in user context, it is located in the WAPT folder `C:\Program Files (x86)\wapt`.

wapttray launches at logon if the option has been ticked during installation. The icon will show up in the Windows tray toolbar.

We can also launch **wapttray** manually with a startup GPO pointing on `C:\Program Files (x86)\wapt\wapttray.exe`.

The tray icon is handy for autonomous users that want to choose the right moment to upgrade their packages.

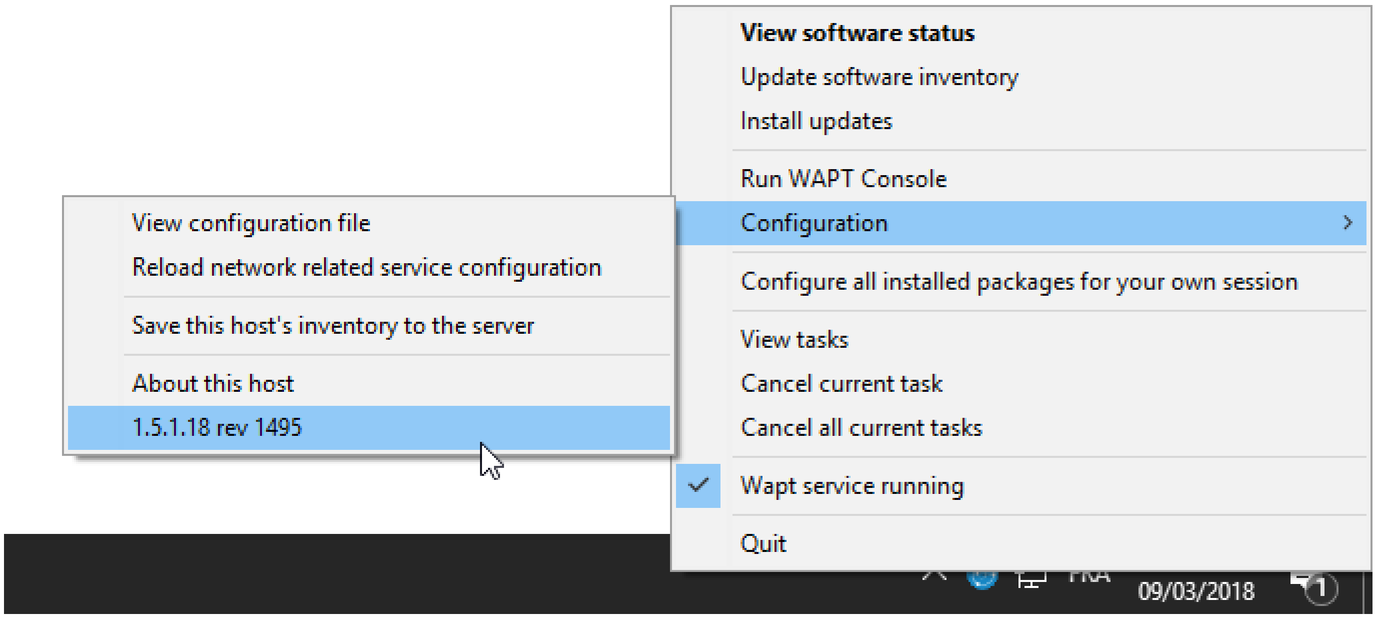


Fig. 118: WAPTtray in Windows notification tray

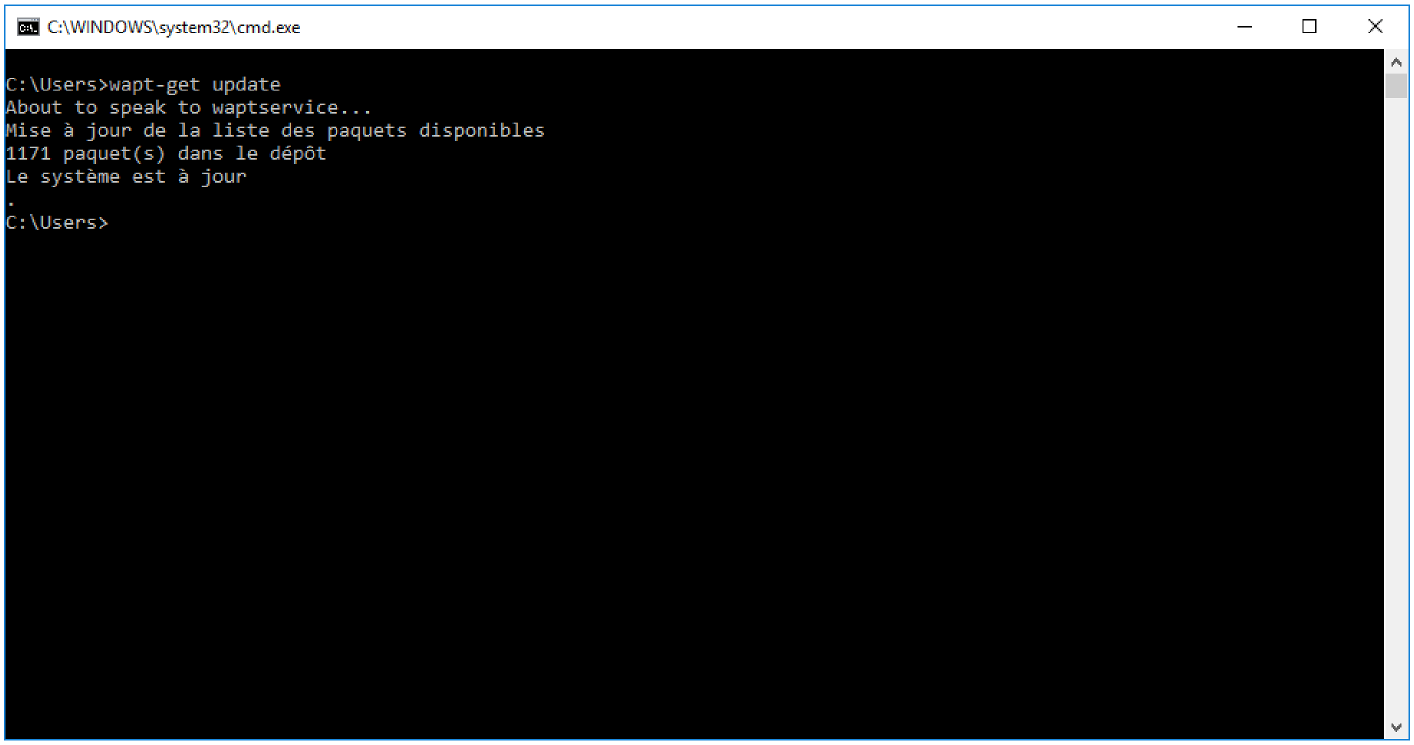
Functionalities of the WAPTtray

Table 23: List of functionalities of the WAPTtray

Action	Description
Showing the status of packages	launches the local web interface in a browser
Launching the installation of a up-date	launches the installation of pending upgrades
Refreshing the list of available	refreshes the list of available packages. Double-clicking on the tray icon brings about the same effect.
Launching the WAPT console	launches the WAPT console
Viewing the configuration file	opens the `C:\Program Files (x86)\wapt\wapt-get.ini` file with *Local Administrator* privileges (credentials may be asked)
Reloading network related service configuration	reloads the connection to the WAPT Server in the event of a network reconfiguration
Uploading the host's inventory to the WAPT Server	updates the host's inventory with the WAPT Server
Configuring all installed packages for the User	launches a **`session-setup`** to configure user environment for all packages installed on the host
Canceling WAPT tasks running on the host	shows running tasks, allows to cancel a running task, allows to cancel all running tasks
Stopping and starting the WAPT service	stops and reloads the *WAPTservice*
Exiting the WAPTtray	closes the tray *icon* without stopping the local *WAPTservice*

6.7.4 Using WAPT with the Command Line

The WAPT agent provides a command line interface utility **`wapt-get`**.



```
C:\Users>wapt-get update
About to speak to waptservice...
Mise à jour de la liste des paquets disponibles
1171 paquet(s) dans le dépôt
Le système est à jour
.
C:\Users>
```

Fig. 119: The Windows Command Line utility

Note:

- by default, command-line actions in WAPT are executed with the rights of the user who launched the **`cmd.exe`**;
- if the *User* is not a *Local Administrator* or if the **`cmd.exe`** has not been launched with *Local Administrator* privileges, the command will be passed on to the **`waptservice`**;
- for security reasons, some actions will require a login and a password;
- only *Local Administrators* and members of the *waptselfservice* Active Directory security group are allowed;
- to force using the WAPT service as a *Local Administrator*, simply add *-S* after **`wapt-get.exe`**;

Using the more common functions in WAPT with the command line

wapt-get update

The **update** command allow to update the list of available packages.

The local WAPT agent will download `Packages` file from the private repository and compare it to its local database.

- If new updates are available, the WAPT agent switches the packages status to **TO-UPGRADE**;
- If new software have been added on the repository, they become downloadable by the WAPT agent;

Note: The **update** command does not download packages, it only updates the local database of packages.

The command `wapt-get update` returns:

```
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

The command **wapt-get upgrade** allows to launch the installation of packages waiting to be upgraded or waiting to be installed.

The local WAPT agent downloads if necessary WAPT packages in its local cache then installs them.

Hint: It's strongly advised to launch a **wapt-get update** command before launching a **wapt-get upgrade** command;

Without previously launching a **update**, the WAPT agent will install nothing;

The command `wapt-get upgrade` returns:

```
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)

```

(continues on next page)

(continued from previous page)

```
=== additional packages ===
tis-mumble                    | tis-mumble (1.2.8-1)
```

wapt-get search

The **search** command allows to search for one or more package in the repositories.

The search command takes one argument to be looked up in package name and description.

The command `wapt-get search "Firefox"` returns:

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

The **install** command launches the installation of a package.

The command takes on argument. That argument is the package name with the repository prefix.

To install Mozilla Firefox, the command is `wapt-get install <prefix>-firefox`.

Note: If the package has not been downloaded to cache, **install** will first download the package to cache, then it will install it.

Attention: Installing a WAPT package with **install** does not add the package as a dependency to the host.

The package is installed on the machine, but if the computer is re-imaged, the package will not be reinstalled automatically.

The command `wapt-get install tis-firefox` returns:

```
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

The **`remove <package name>`** command removes a package.

The command takes on argument. That argument is the package name with the repository prefix.

To remove Mozilla Firefox, the command is **`wapt-get remove <prefix>-firefox`**.

Attention: Removing a WAPT package with **`remove`** does not remove the package dependency on the host.

The package will effectively be uninstalled from the machine, but it will automatically be reinstalled on the next :command:`upgrade`.

To completely remove a package from a host, do a **`remove`** for the targeted package, then edit the host configuration via the WAPT console to remove the package dependency on the host.

The command `wapt-get remove tis-firefox` returns:

```
Removing tis-firefox ...

=== Removed packages ===
  tis-firefox
```

wapt-get clean

The **`clean`** command removes packages from the `C:\Program Files (x86)\wapt\cache` folder.

The **`clean`** command is launched after each upgrade to save disk space.

The command `wapt-get clean` returns:

```
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
```

Using special Command Lines with WAPT

wapt-get register

The **`wapt-get register <description>`** command reports the computer hardware and software inventory to the WAPT inventory server.

Hint: You can pass a description as an argument to the **`register`**, that description will be displayed in the WAPT console in the column *description*.

You may benefit from WAPT to improve your IT management by affecting a username or a computer serial as descriptions for your hosts.

The command **`wapt-get register "John Doe PC`** returns nothing;

wapt-get download

The **wapt-get download <package name>** command downloads the WAPT package to the local cache located at `C:\Program Files\wapt\cache`.

The command **wapt-get download tis-7zip** returns:

```
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

The **wapt-get download-upgrade** command downloads packages to be upgraded to the local WAPT cache `C:\Program Files (x86)\wapt\cache`.

The command **wapt-get download-upgrade** returns:

```
=== downloaded packages ===
C:\Program Files (x86)\wapt\cache\tis-firebird_2.5.5.26952-1_all.wapt
```

wapt-get show

The **wapt-get show <package name>** command displays informations stored in the `Packages` index file.

If several versions of a package are available on the repository, every version of the package will be displayed.

The command **wapt-get show tis-firebird** returns:

```
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

The **`wapt-get list`** command lists WAPT packages that are installed on the computer.

The command **`wapt-get list`** returns:

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

The **`wapt-get upgradedb`** command upgrades the local WAPT database schema if necessary.

The command **`wapt-get upgradedb`** returns:

```
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

The **wapt-get setup-tasks** command adds **update** and **upgrade** scheduled tasks to local host.

Hint: This function is useful when it is desirable not to use the WAPT service, otherwise **waptservice** will take care of it.

To make it work, the following arguments must be configured in `wapt-get.ini`:

- *waptupdate_task_maxruntime*;
- *waptupgrade_task_maxruntime*;
- *waptupdate_task_period*;
- *waptupgrade_task_period*;

Then:

- the **wapt-get enable-tasks** command will enable scheduled tasks;
- the **wapt-get disable-tasks** command will disable scheduled tasks;

wapt-get add-upgrade-shutdown - wapt-get remove-upgrade-shutdown

- the **wapt-get add-upgrade-shutdown** command adds a **waptexit** local security policy object, enabling the execution of **waptexit** at system shutdown;
- the **wapt-get remove-upgrade-shutdown** command removes the **waptexit** local security policy object, disabling the execution of **waptexit** during system shutdown;

wapt-get inventory

The **wapt-get inventory** command displays all local inventory information in JSON format.

The command **wapt-get inventory** returns:

```
{
  "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

The command **wapt-get update-status** resends local status to the WAPT inventory server.

Note: If a hardware component has changed on the computer, **update-status** would not report that information back to the WAPT inventory server.

To do so, the command to be used is **inventory**.

The command **wapt-get update-status** returns:

```
Inventory correctly sent to server https://srvwapt.mydomain.lan.
```

wapt-get setlocalpassword

The **wapt-get setlocalpassword** command allows to define a local password for WAPT package installations.

The command **wapt-get setlocalpassword** returns:

```
Local password:
Confirm password:
Local auth password set successfully
```

wapt-get reset-uuid

The **wapt-get reset-uuid** command retrieves the host *UUID* from BIOS and resends it to the WAPT inventory server.

The command **wapt-get reset-uuid** returns:

```
New UUID: B0F23D44-86CB-CEFE-A8D6-FB8E3343FE7F
```

wapt-get generate-uuid

The **wapt-get generate-uuid** command creates a new host *UUID* and resends it to the WAPT inventory server.

Hint: Some batches of computers have their BIOS with identical *UUID*. It is a BIOS manufacturer setting problem because no two *UUID* should be the same.

The command **generate-uuid** exist to solve that problem.

The command **wapt-get generate-uuid** returns:

```
New UUID: 6640f174-de90-4b00-86f7-d7834ceb45bc
```

wapt-get get-server-certificate

The **wapt-get get-server-certificate** command downloads the SSL certificate from the WAPT Server to use HTTPS to communicate with the WAPT Server.

The downloaded certificate is stored in `C:\Program Files(x86)\waptssl\server`.

The command **wapt-get get-server-certificate** returns:

```
Server certificate written to C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt
```

wapt-get enable-check-certificate

The **wapt-get enable-check-certificate** command downloads the SSL certificate from the WAPT Server and enables secured communication with the server.

The command **wapt-get enable-check-certificate** returns:

```
Server certificate written to C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt
wapt config file updated
```

wapt-get session-setup

The **wapt-get session-setup** command launches user level customizations of installed WAPT packages.

Hint: The **session-setup** instruction sets are defined in WAPT package's `setup.py` file.

Every instruction set is stored in a SQLite local database.

The command **session-setup** is launched at every startup, the user environment customization script is executed only once per user per package version.

Note: The argument *ALL* will launch **session-setup** for all installed WAPT packages.

The command **wapt-get session-setup ALL** returns:

```
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
```

Using the Command Line to create WAPT packages

wapt-get make-template

The **wapt-get make-template <msi or exe file> <package name>** command allows to create a package template from a MSI or an EXE installer.

You will find the complete procedure for *creating WAPT packages*.

Hint:

- If you have previously installed *tis-waptdev* package on your development computer, **PyScripter** editor will launch automatically and open the package in development mode.

The command `wapt-get make-template C:\\Users\\User\\Downloads\\tightvnc-2.8.5-gpl-setup-64bit.msi tis-tightvnc` returns:

```
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

The **wapt-get make-host-template <host FQDN>** command creates an empty WAPT host package from a template.

The command `wapt-get make-host-template host01.mydomain.lan` returns:

```
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

The **wapt-get make-group-template <name of group>** command creates an empty WAPT group package from a template.

The command `wapt-get make-group-template accounting` returns:

```
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

The **wapt-get list-registry <keyword>** command lookups a keyword in software installed by WAPT on the computer.

The output of **list-registry** is a table listing *uninstall keys* for each software corresponding to the search term.

The command `wapt-get list-registry firefox` returns:

```
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

The **wapt-get sources <package name>** command downloads sources from a source code management platform like Git or SVN.

The command `wapt-get sources tis-firefox` returns nothing;

wapt-get build-package

The **wapt-get build-package <path to the package>** command builds a WAPT package and signs it with the private key of the *Administrator*.

Note: The path to the private key, the default prefix and the default development path must be properly set in the `wapt-get.ini` file.

The command `wapt-get sources tis-firefox` returns: :

```
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
```

(continues on next page)

(continued from previous page)

```
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

The **wapt-get sign-package <path to the package>** command signs a package with the private key of the *Administrator*.

Attention: **sign-package** does not rename the WAPT package with the chosen prefix of the *Organization*.

The command `wapt-get sign-package C:\\waptdev\\smp-7zip_16.4.0.0-1_all.wapt` returns:

```
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

The **wapt-get build-upload <path to the package>** command builds and uploads a WAPT package onto the main WAPT repository.

Hint: By passing the *-i* argument to **build-upload**, the WAPT packaging version number is incremented before upload, so to avoid having to modify manually the `control` file.

The command `wapt-get -i build-upload C:\\waptdev\\tis-tightvnc-wapt` returns:

```
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
wTVHiqpbXnZotiVIGrJDhgbaLwZ9CK6pfWiflC4126nx6PMF3T1i6w0R0NOE2wJpOSRYESk71DUz
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

The **wapt-get duplicate <package source> <package new_duplicate>** command duplicates a package downloaded from the repository and opens it as a **PyScripter** project.

The command `wapt-get duplicate tis-firefox tis-firefox-custom` returns:

```
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

The **wapt-get edit <package name>** command downloads and edits a WAPT package.

The command `wapt-get edit tis-firefox` returns:

```
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

The **wapt-get edit-host <host FQDN>** command edits a WAPT *host* package.

wapt-get upload-package

The **wapt-get upload-package <path to the package>** command uploads a package onto the main WAPT repository.

The command `wapt-get upload-package C:\\waptdev\\tis-tightvnc_2.8.5.0-1_all.wapt` returns:

```
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

The **wapt-get update-packages <path to folder>** command scans a local repository and creates the `Packages` index file.

The command `wapt-get update-packages D:\\Data\\WAPT` returns:

```
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
```

(continues on next page)

(continued from previous page)

```
  D:\Data\WAPT\tis-7zip_16.4.0.0-1_all.wapt
  D:\Data\WAPT\tis-mumble_3.14-3_all.wapt
  D:\Data\WAPT\tis-noforcereboot_1.0-1_all.wapt
Skipped packages:
```

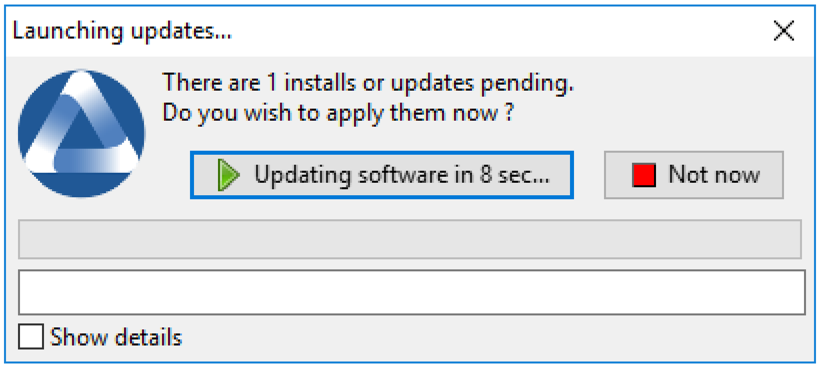
6.7.5 Using WAPTExit

waptexit allows to upgrade and install WAPT packages when a host is shutting down, at the user's request, or at a scheduled time.

The mechanism is simple. If packages are waiting to be upgraded, they'll be installed.

Hint: When to use WAPTexit?

The WAPTexit method is very effective in most situation because it does not require the intervention of the *User* or the *Administrator*.



There are 1 installs or updates pending.
Do you wish to apply them now ?

Fig. 120: WAPTexit window
WAPTexit

waptexit executes by default on shutdown; it is installed by default with the WAPT agent.

The behavior of **waptexit** is customizable in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Manually triggering the execution of WAPTexit

By creating a desktop shortcut, one can allow users to launch upgrades by themselves at a time that is convenient to them simply by clicking the *WAPTexit* icon.

The behavior of **waptexit** is customizable in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Triggering WAPTexit with a scheduled task

One can deploy a GPO or a WAPT package that will trigger WAPTexit at a pre-scheduled time.

Triggering WAPTexit with a scheduled task is best suited for servers that are not shutdown frequently.

You may adapt the procedure describing how to deploy the WAPT agent to *trigger the WAPTexit.exe script at the time of your choosing*.

Hint: You can use the following script for your scheduled task, adapted to your need (**Enterprise only**):

```
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')"
```

Warning: All running software that are upgraded may be killed with possible loss of data. WAPTexit may fail to upgrade a software program if a software that you are upgrading is in the `impacted_process` list of the `control` file of one of the software you are trying to upgrade. See *below* for more information.

The method of trigerring WAPTexit at a scheduled time is the least recommended method for desktops. It is better to let WAPTexit execute at shutdown or on user request.

Avoiding the cancellation of upgrades

To disable the interruption of the installation of updates you can run **waptexit** with the argument:

```
waptexit.exe -allow_cancel_upgrade = True
```

Otherwise **waptexit** will take the value indicated in `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**.

Increase the trigger time in waptexit

To specify the wait time before the automatic start of the installations you can start **waptexit** with the argument:

```
waptexit.exe -waptexit_countdown = 10000
```

Otherwise **waptexit** will take the value indicated in the 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**.

Do not interrupt user activity

To tell WAPT not to run an **upgrade** of running software on the machine (*impacted_process* attribute of the package), you can run **waptexit** with the argument:

```
waptexit.exe -only_if_not_process_running=True
```

Otherwise **waptexit** will take the value indicated in `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**.

Launching the installation of packages with a special level of priority

To tell WAPT to only upgrade high priority packages, you can run **waptexit** with the argument:

```
waptexit.exe -priorities = high
```

Otherwise **waptexit** will take the value indicated in `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).

Customizing WAPTexit

It is possible to customize waptexit by placing the image you want in `C:\Program Files (x86)\wapt\templates\waptexit-logo.png`.

Registering/ unregistering WAPTexit

To register or unregister **waptexit** in local shutdown group strategy scripts, use:

- to enable **waptexit** at host shutdown:

```
wapt-get add-upgrade-shutdown
```

- to disable **waptexit** at host shutdown:

```
wapt-get remove-upgrade-shutdown
```

6.8 How to use WAPT Enterprise

This section of the documentation covers the use of WAPT Enterprise features. New in version 1.7: Enterprise

6.8.1 Using Organizational Unit packages in WAPT

Hint: Feature only available with WAPT **Enterprise**.

Working principle

WAPT Enterprise offers organizational unit packages functionnality.

It automates software installations based on your Active Directory infrastructure.

The WAPT agent is aware of its position in the Active Directorytree structure, therefore it knows the hierarchy of Organizational Units that concerns it, for example:

```
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
```

If an Organizational Unit package is defined on each level, WAPT agent will automtically download packages and configurations that are attached to each level, by inheritance, and apply attached packages and their dependencies.

Filters and actions available with Organizational Units

Hint: You can see in the picture that **update** and **upgrade** actions can be performed through this menu, thus selecting hosts by their Organizational Unit.

In the **Enterprise** version, you may filter how hosts are displayed based on the Active Directory OU they belong to.

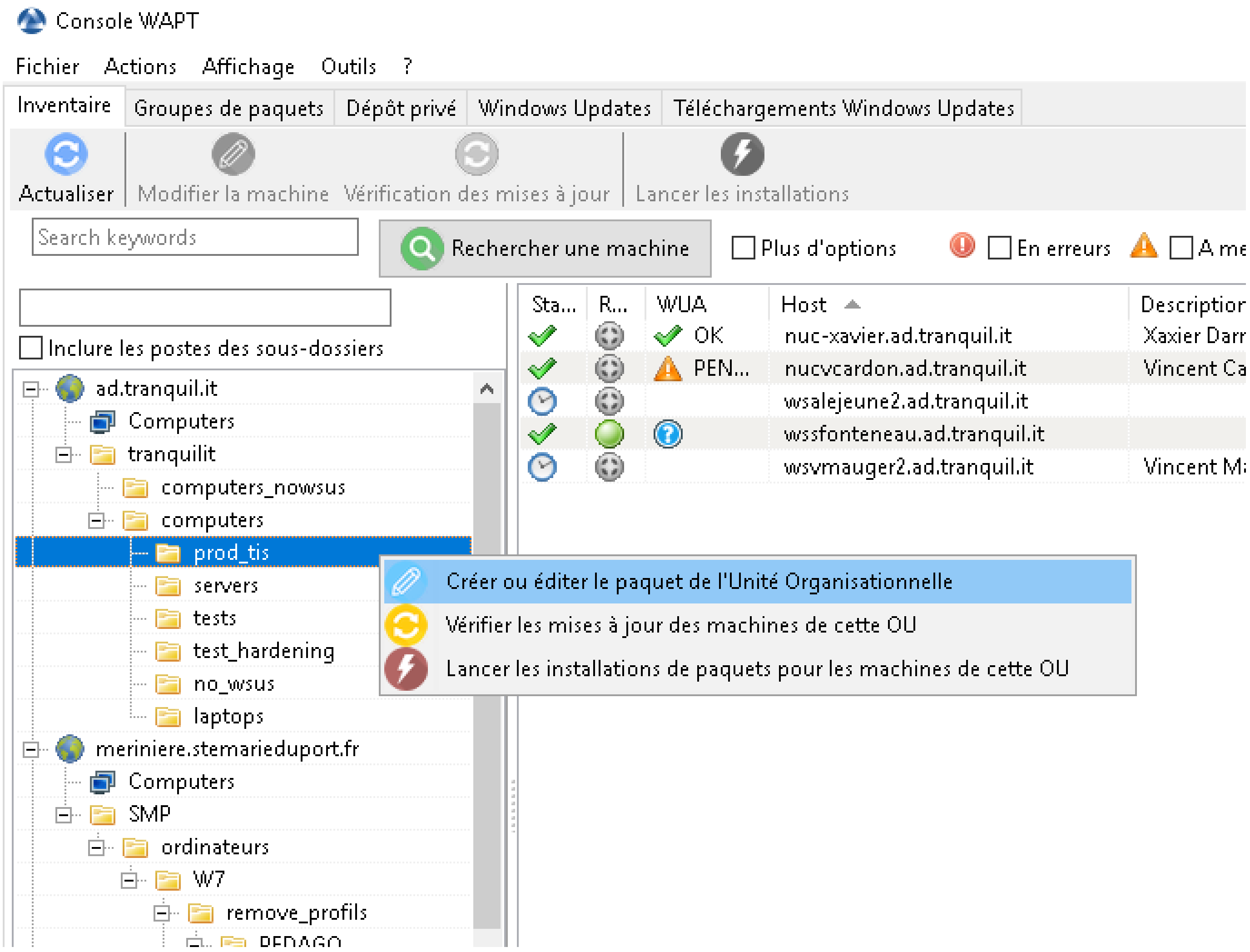
The checkbox *Include hosts in subfolders* allows to display hosts in subfolders.

Creating Organizational Unit packages in the WAPT console

You can create *unit* packages by *Right clicking on an OU* → *Create or edit the unit package*.

A window opens and you are prompted to choose which packages must be in **unit** bundle.

Save the package and it will be uploaded to the WAPT server.



Sta...	R...	WUA	Host	Descriptior
✓		✓ OK	nuc-xavier.ad.tranquil.it	Xaxier Darr
✓		⚠ PEN...	nucvcardon.ad.tranquil.it	Vincent Ca
			wsalejeune2.ad.tranquil.it	
✓	●	?	wssfonteneau.ad.tranquil.it	
			wsvmauger2.ad.tranquil.it	Vincent M

Fig. 121: WAPT console showing options applicable to OU

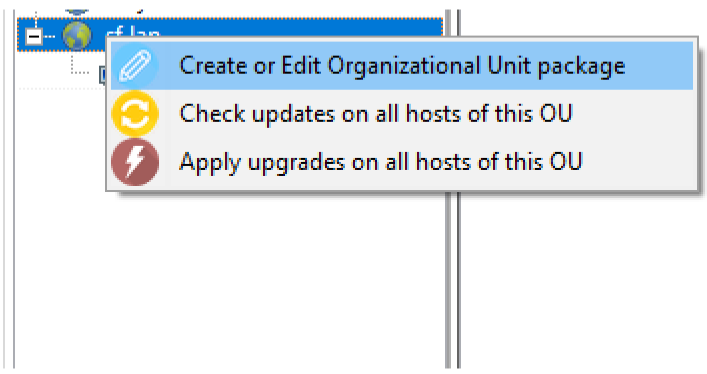
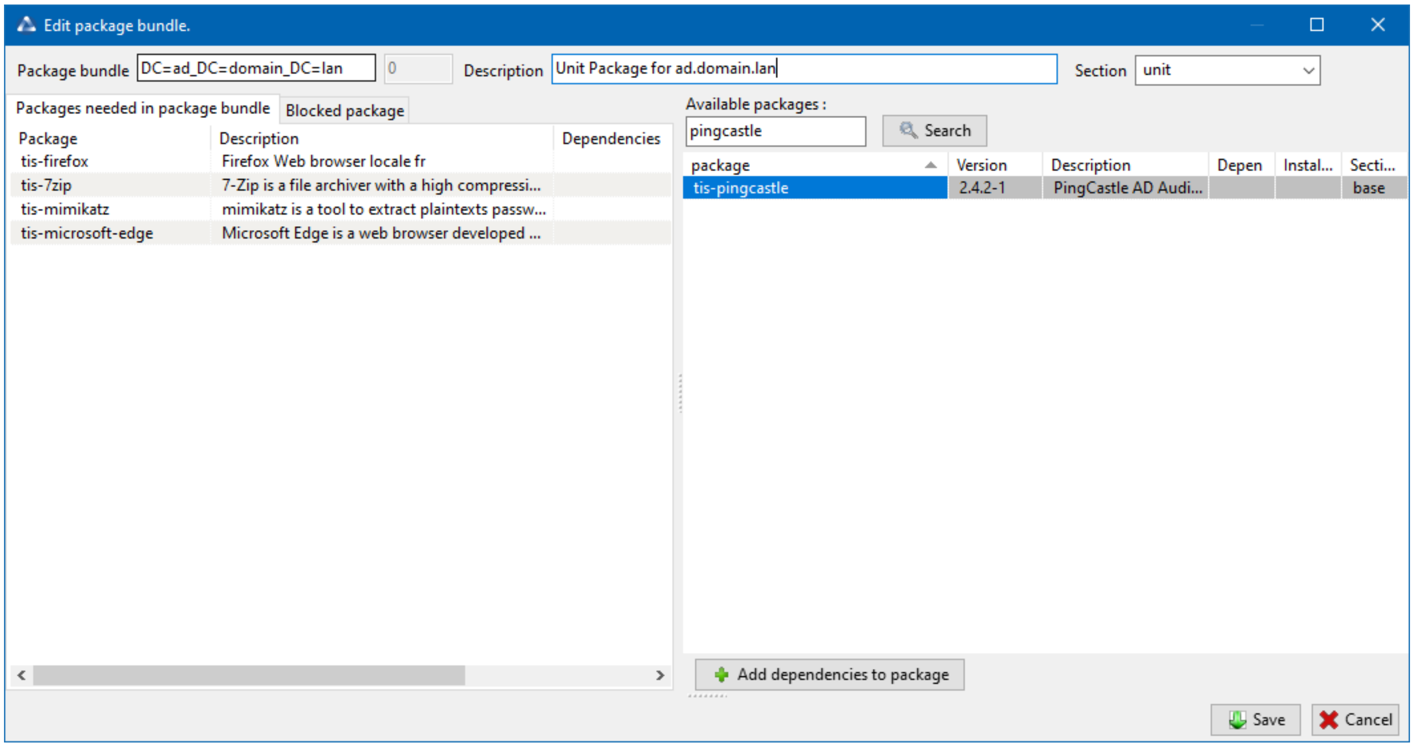

Create or Edit Organizational Unit package
Check updates on all hosts of this OU
Apply upgrades on all hosts of this OU

Fig. 122: Right-click on OU to create unit package.



Package	Description	Dependencies
tis-firefox	Firefox Web browser locale fr	
tis-7zip	7-Zip is a file archiver with a high compressi...	
tis-mimikatz	mimikatz is a tool to extract plaintexts passw...	
tis-microsoft-edge	Microsoft Edge is a web browser developed ...	

package	Version	Description	Depen	Instal...	Secti...
tis-pingcastle	2.4.2-1	PingCastle AD Audi...			base

Fig. 123: Adding package to unit bundle.

Faking organizational unit for WORKGROUP hosts

It can happen that some specific hosts cannot be joined to an Active Directory domain.

With that specificity, such hosts do not show up in your Active Directory Organizational Units in your WAPT Console.

To make all hosts show up in the console under the right Organizational Unit, whether they are joined to an AD domain or not, WAPT allows you to specify a *fake* Organizational Unit WAPT agent configuration file.

The benefits of this trick are:

- you can manage these hosts with WAPT as if they where joined to the AD;
- out-of-domain and workgroup hosts are now showing up in AD tree view;
- *unit* packages are usable on these hosts;

To setup a *fake* Organisational Unit on hosts, create an empty WAPT package;

```
wapt-get make-template demo-configure-fake-ou
```

Then use the following code:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
```

(continues on next page)

(continued from previous page)

```

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)
```

The `host_organizational_unit_dn` must be like below in `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.

New in version 1.7: Enterprise

6.8.2 Using profile bundles in WAPT

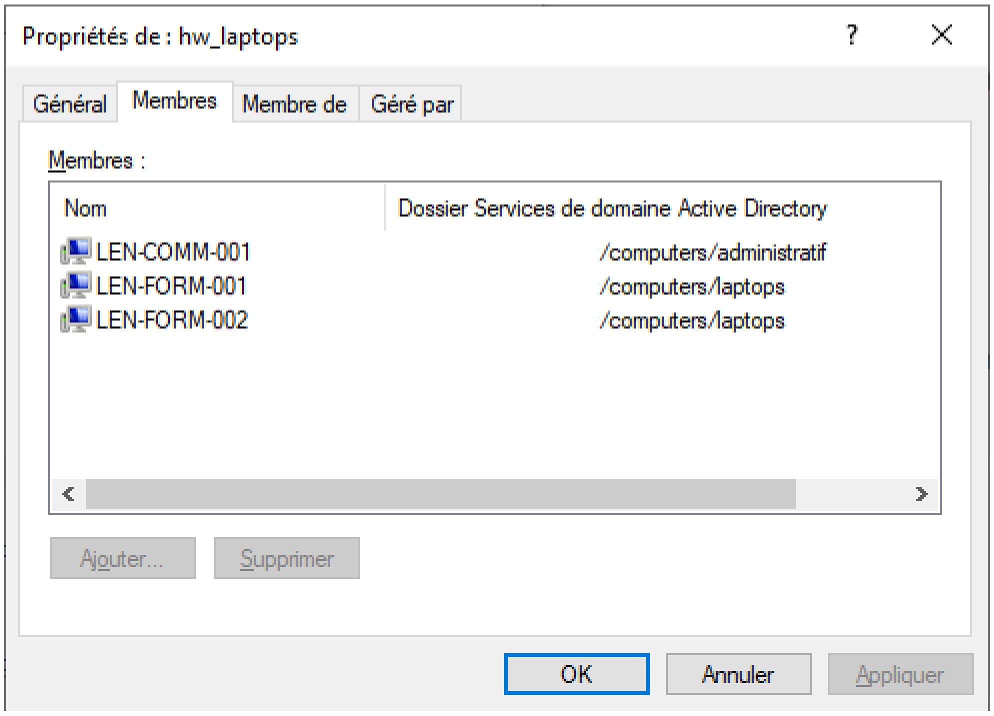
Hint: Feature only available with WAPT **Enterprise**.

Working principle

WAPT Enterprise offers Active Directory profile bundle functionnality.

It automates installation of WAPT software and configuration packages on hosts, based on their membership to Active Directory Computer Security Groups.

Important: Active Directory Computer's security groups contains Computers, not Users.



Nom	Dossier Services de domaine Active Directory
LEN-COMM-001	/computers/administratif
LEN-FORM-001	/computers/laptops
LEN-FORM-002	/computers/laptops

Fig. 124: Active Directory computer group

Automatically installing software and configurations based on user and user group membership is not implemented with WAPT. This use case is better served with the differenciated *self-service* feature that is also available with WAPT Enterprise.

Creating profile bundle packages in WAPT console

You can create *profile* bundle packages by clicking on *Bundles -> Create AD Profile*.

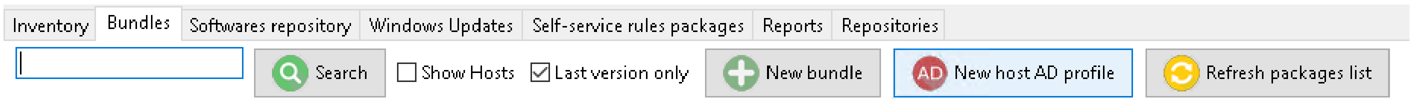


Fig. 125: Click on New host AD profile to create a *profile* bundle

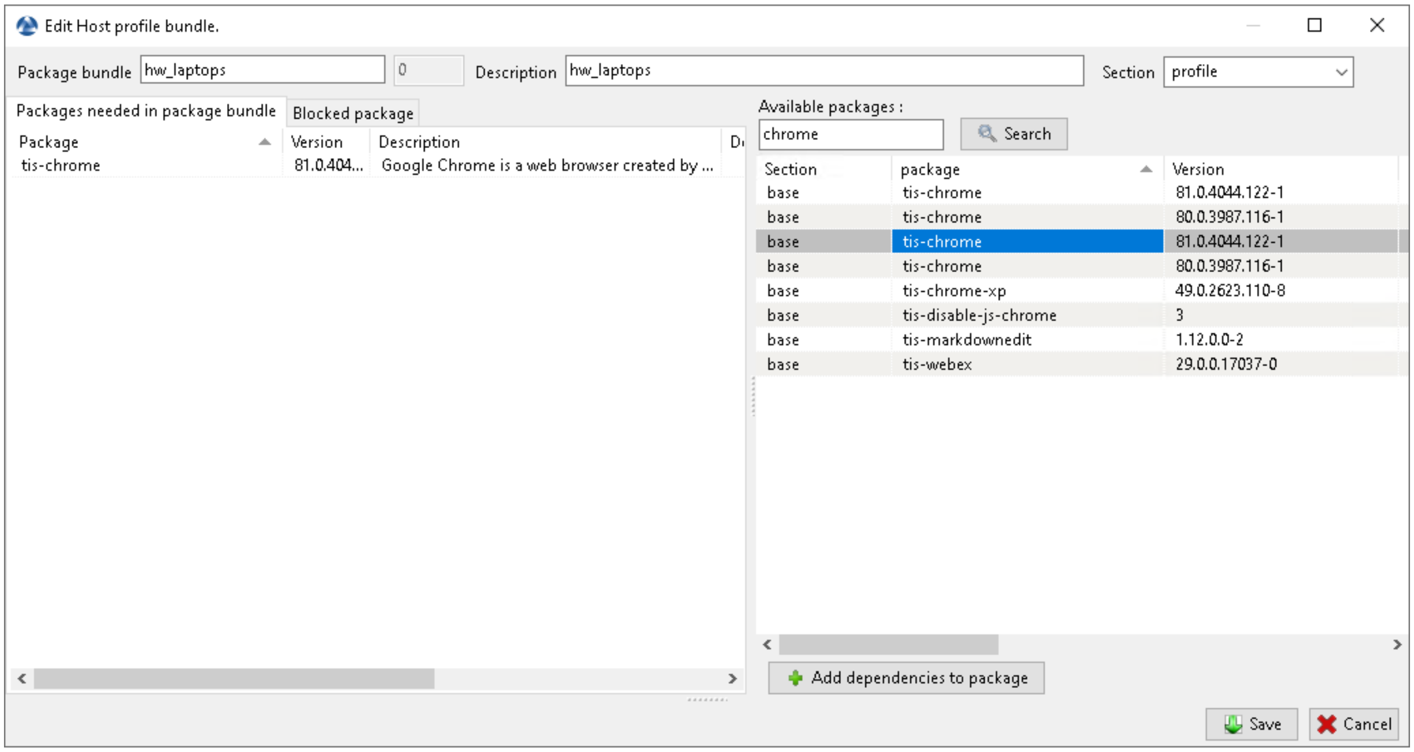
Important: Requirements:

- the *profile* package name must be **exactly** the same as the AD Security group name;
- the *profile* package name is case sensitive;

Example:

- AD Security group: `HW_laptops`;
- WAPT profile bundle: `HW_laptops`;

A window opens and you are prompted to choose which packages must be in the just created **profile** bundle.


Edit Host profile bundle.
Package bundle
Description
Section
Packages needed in package bundle
Blocked package
Available packages :
Search
Add dependencies to package
Save
Cancel

Package bundle: hw_laptops | 0 | Description: hw_laptops | Section: profile

Available packages: chrome

Package	Version	Description
tis-chrome	81.0.404...	Google Chrome is a web browser created by ...

Section	package	Version
base	tis-chrome	81.0.4044.122-1
base	tis-chrome	80.0.3987.116-1
base	tis-chrome	81.0.4044.122-1
base	tis-chrome	80.0.3987.116-1
base	tis-chrome-xp	49.0.2623.110-8
base	tis-disable-js-chrome	3
base	tis-markdownedit	1.12.0.0-2
base	tis-webex	29.0.0.17037-0

Fig. 126: Adding package to profile bundle

Save the *profile* bundle package and it will be uploaded to the WAPT server. New in version 1.7: Enterprise

6.8.3 Using WAPT Windows Update Agent (WAPTWUA)

Note: Since version 1.7, WAPT is able to manage Windows Updates on your endpoints.

- the internals of WAPTWUA is based on the WUA (Windows Update Agent) API,
- for more information: https://docs.microsoft.com/en-us/windows/win32/wua_sdk/using-the-windows-update-agent-api;

Working principle

Regularly, the WAPT server downloads an updated `wsusscn2.cab` file from Microsoft servers. By default, downloads happen once a day and no download is triggered if the `wsusscn2.cab` file has not changed since the last download.

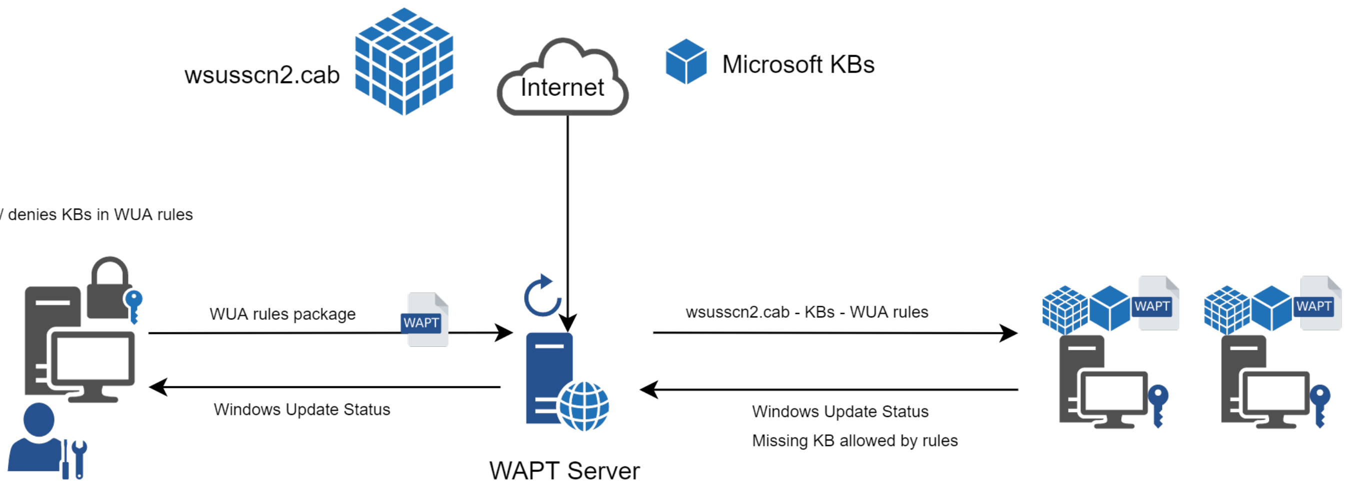


Fig. 127: WAPT Windows Update flow process.

Note: In some cases, you may wish to push new KBs before the next Patch Tuesday release.

To do so, you may follow *this documentation* on packaging *.msu* files for these *Out-of-band* updates.

The `wsusscn2.cab` file is then downloaded by the WAPT agent from its nearest repository and then passed on to the standard WUA Windows utility to crunch the update tree for the host.

Regularly, the host will analyze the available updates using the `wsusscn2.cab` file. The host will send its list of needed updates as determined by its WUA to the WAPT server.

If an update is pending on the host and if that update is not present on the WAPT server, the server will download the needed update from official Microsoft servers.

Hint: This mode of operation allows WAPT to download only the necessary updates on the computers, thus saving bandwidth, download time and disk space.

Note: Downloaded updates are stored:

- on Linux hosts in `/var/www/waptwua`;
- on Windows hosts in `C:\wapt\waptserver\repository\waptwua`;

The WAPT Windows Update Agent repository download URL is based on the `repo_url` parameter in `wapt-get.ini`:

- in case of repository replication, it is fully operational with WAPT Windows Update to reduce bandwidth use;
- do not forget to synchronize the `waptwua` folder if you are replicating your packages with distant repositories;

Note: If in your company, a proxy is needed to go out on the Internet, then be sure to *set the proxy server in the waptserver.ini file*.

Difference between WAPT Windows Updates and 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 only downloads updates that have been requested by at least one computer client. This helps to keep the local database small (a few 10s of Gigabytes) and it can be easily cleaned up if you want to recover space.

Major OS upgrades

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.

In the case of WAPT Windows Updates, you need to create a OS update package using a template package provided on https://store.wapt.fr.

Driver upgrades

Driver upgrades via WSUS are not recommanded since it is hard to properly handle side effects. In the case of WAPT Windows Updates, **DRIVERS ARE NOT DOWNLOADED** since they are not referenced in the `wsusscn2.cab` files provided by Microsoft.

It is recommanded to push driver updates via a custom WAPT package. If the driver patch is packaged as a *`msu`*, you may package it as a standard WAPT package.

Just select the `msu` file and click :menuselection:"create package" in the WAPT console to launch the wizard for simplified package creation.

If the driver update is packaged as a *`zip`* containing the `exe` file, you can create a WAPT package containing the necessary files and **`setup.exe`** binary with the correct silent flag.

Out of band KB

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.

If you want to deploy an OOB KB update, you can download it from the microsoft catalog 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.

You have to be carefull that OOB updates may break your system, be sure to read the prerequisites on the Microsoft bulletin corresponding to the update and thoroughly test the update.

Configuring WAPTWUA on the WAPT agent

WAPTWUA is configured in `wapt-get.ini`.

Add `[waptwua]` section.

You then have several options:

Table 24: Configuration options in the `[waptwua]` section in the `wapt-get.ini`

Options	Default Value	Description
`enabled`	False	Enable or disable WAPTWUA on this machine.
`allow_direct_download`	False	Allow direct download of updates from Microsoft servers if the WAPT server is not available
`default_allow`	False	Set if missing update is authorized or not by default
`filter`	Type='Software' or Type='Driver'	Define the filter to apply for the Windows update scan
`download_scheduling`	None	Set the Windows Update scan recurrence (Will not do anything if *waptwua* package rule or `wsusscn2.cab` file have not changed) (ex: 2h)
`install_scheduling`	None	Set the Windows Update install recurrence (Will do nothing if no update is pending) (ex: 2h)
`install_at_shutdown`	False	Install update when the machine will shutdown
`install_delay`	None	Set a deferred installation delay before publication in the repository (ex: 7d)
`allowed_severities`	None	Define a severity list that will be automatically accepted during a WAPT windows update scan. ex: *Important*, *Critical*, *Moderate*

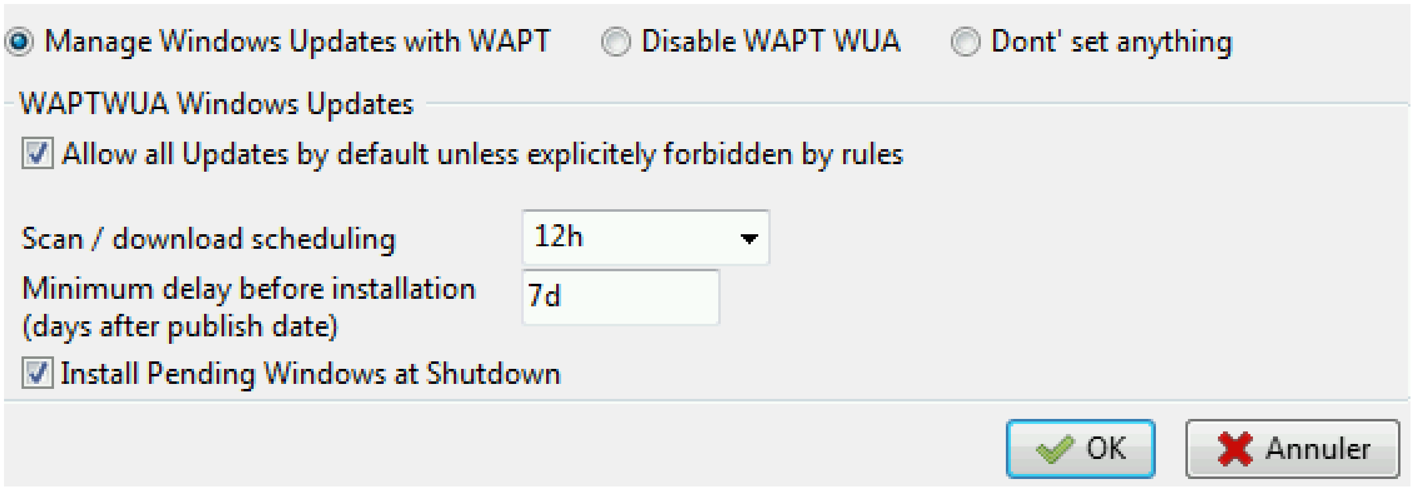
Hint: These options can be set when generating the agent.

Example `[waptwua]` section in `wapt-get.ini` file:

```
[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.

When you create the `waptagent.exe` from your console, these options are equivalent to this:


[✓ Manage Windows Updates with WAPT] Disable WAPT WUA Dont' set anything

WAPTWUA Windows Updates

☑ Allow all Updates by default unless explicitely forbidden by rules

Scan / download scheduling: 12h

Minimum delay before installation (days after publish date): 7d

☑ Install Pending Windows at Shutdown

Hint: if *default_allow* option is `True` and Wapt WUA is enabled too, clients will contact the WAPT Server and ask to download the missing updates. The clients will install missing updates on their own at time of upgrade.

Example package source code to modify `[waptwua]` settings:

```
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')
```

Using WAPTWUA from the console

The *WAPT Windows Update Agent* tab in the WAPT console comes with two sub-menus to manage WAPTWUA.

WAPTWUA Package

The *WAPTWUA Package* tab allows you to create *waptwua* rules packages.

- when this type of package is installed on a machine, it indicates to the WAPTWUA agent the authorized or forbidden KBs (Knowledge Base articles);
- when several *waptwua* packages are installed on a machine, the different rules will be merged;
- when a `cab` is neither mentioned as authorized, nor mentioned as prohibited, WAPT agents will then take the value of `default_allow` in `wapt-get.ini`;

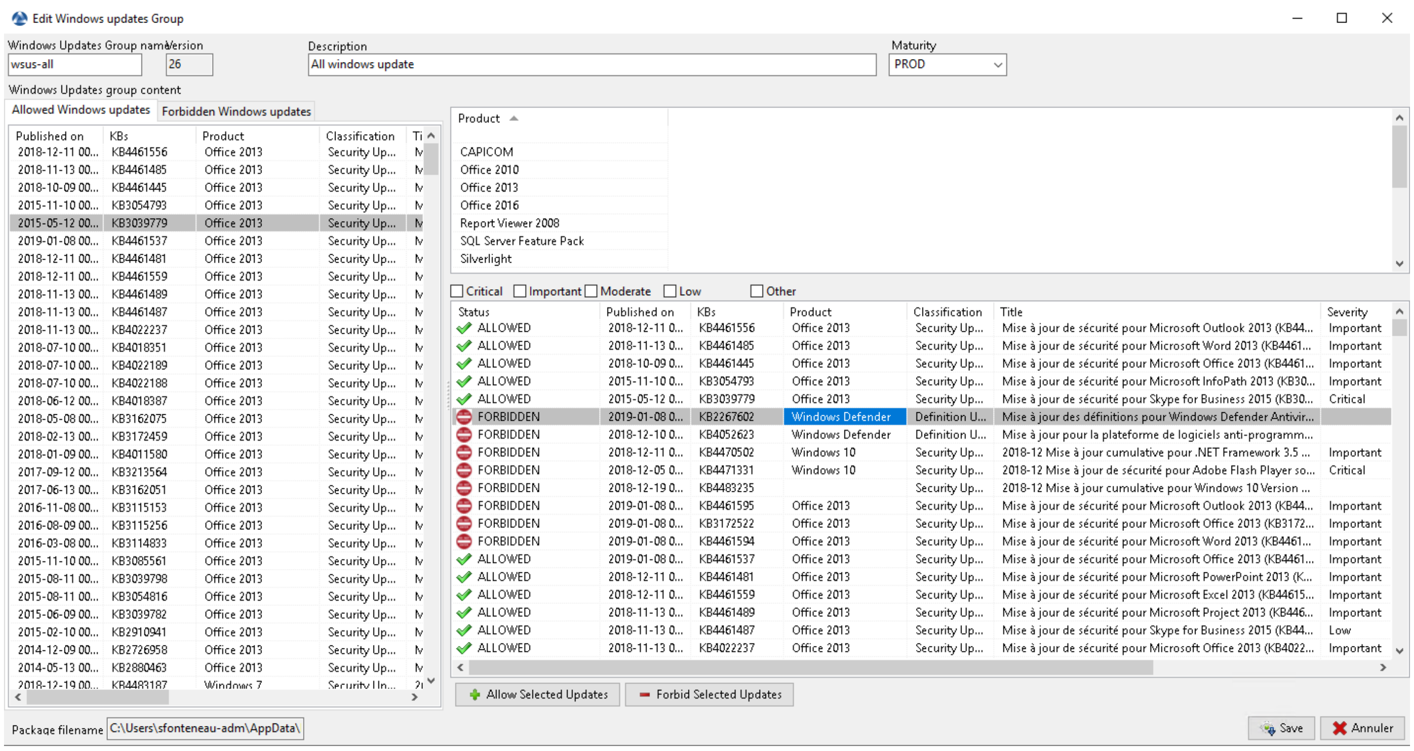
If a Windows update has not yet been downloaded to the WAPT server, then the WAPT agent will flag the update as *MISSING*.

Note:

- if the WAPTWUA agent configuration is set to `default_allow = True`, then it will be necessary to specify the forbidden `cab`;
- if the WAPTWUA agent configuration is set to `default_allow = False`, then it will be necessary to specify the authorized `cab`;

Hint:

- to test updates on a small set of computers, you can set WAPTWUA default value to `default_allow = False`;
- you can test updates on a small sample of hosts and if everything is good, you can release the updates to the entire fleet of computers;



Windows Updates Group name: wsus-all
Version: 26
Description: All windows update
Maturity: PROD

Windows Updates group content

Published on	KBs	Product	Classification	Ti
2018-12-11 00...	KB4461556	Office 2013	Security Up...	M
2018-11-13 00...	KB4461485	Office 2013	Security Up...	M
2018-10-09 00...	KB4461445	Office 2013	Security Up...	M
2015-11-10 00...	KB3054793	Office 2013	Security Up...	M
2015-05-12 00...	KB3039779	Office 2013	Security Up...	M
2019-01-08 00...	KB4461537	Office 2013	Security Up...	M
2018-12-11 00...	KB4461481	Office 2013	Security Up...	M
2018-12-11 00...	KB4461559	Office 2013	Security Up...	M
2018-11-13 00...	KB4461489	Office 2013	Security Up...	M
2018-11-13 00...	KB4461487	Office 2013	Security Up...	M
2018-11-13 00...	KB4022237	Office 2013	Security Up...	M
2018-07-10 00...	KB4018351	Office 2013	Security Up...	M
2018-07-10 00...	KB4022189	Office 2013	Security Up...	M
2018-07-10 00...	KB4022188	Office 2013	Security Up...	M
2018-06-12 00...	KB4018387	Office 2013	Security Up...	M
2018-05-08 00...	KB3162075	Office 2013	Security Up...	M
2018-02-13 00...	KB3172459	Office 2013	Security Up...	M
2018-01-09 00...	KB4011580	Office 2013	Security Up...	M
2017-09-12 00...	KB3213564	Office 2013	Security Up...	M
2017-06-13 00...	KB3162051	Office 2013	Security Up...	M
2016-11-08 00...	KB3115153	Office 2013	Security Up...	M
2016-08-09 00...	KB3115256	Office 2013	Security Up...	M
2016-03-08 00...	KB3114833	Office 2013	Security Up...	M
2015-11-10 00...	KB3085561	Office 2013	Security Up...	M
2015-08-11 00...	KB3039798	Office 2013	Security Up...	M
2015-08-11 00...	KB3054816	Office 2013	Security Up...	M
2015-06-09 00...	KB3039782	Office 2013	Security Up...	M
2015-02-10 00...	KB2910941	Office 2013	Security Up...	M
2014-12-09 00...	KB2726958	Office 2013	Security Up...	M
2014-05-13 00...	KB2880463	Office 2013	Security Up...	M
2018-12-19 00...	KB4483187	Windows 7	Security Up...	2(

Product
CAPICOM
Office 2010
Office 2013
Office 2016
Report Viewer 2008
SQL Server Feature Pack
Silverlight

☐ Critical ☐ Important ☐ Moderate ☐ Low ☐ Other

Status	Published on	KBs	Product	Classification	Title	Severity
✓ ALLOWED	2018-12-11 0...	KB4461556	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Outlook 2013 (KB44...	Important
✓ ALLOWED	2018-11-13 0...	KB4461485	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Word 2013 (KB4461...	Important
✓ ALLOWED	2018-10-09 0...	KB4461445	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4461...	Important
✓ ALLOWED	2015-11-10 0...	KB3054793	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft InfoPath 2013 (KB30...	Important
✓ ALLOWED	2015-05-12 0...	KB3039779	Office 2013	Security Up...	Mise à jour de sécurité pour Skype for Business 2015 (KB30...	Critical
⛔ FORBIDDEN	2019-01-08 0...	KB2267602	Windows Defender	Definition U...	Mise à jour des définitions pour Windows Defender Antivir...	
⛔ FORBIDDEN	2018-12-10 0...	KB4052623	Windows Defender	Definition U...	Mise à jour pour la plateforme de logiciels anti-programm...	
⛔ FORBIDDEN	2018-12-11 0...	KB4470502	Windows 10	Security Up...	2018-12 Mise à jour cumulative pour .NET Framework 3.5 ...	Important
⛔ FORBIDDEN	2018-12-05 0...	KB4471331	Windows 10	Security Up...	2018-12 Mise à jour de sécurité pour Adobe Flash Player so...	Critical
⛔ FORBIDDEN	2018-12-19 0...	KB4483235		Security Up...	2018-12 Mise à jour cumulative pour Windows 10 Version ...	
⛔ FORBIDDEN	2019-01-08 0...	KB4461595	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Outlook 2013 (KB44...	Important
⛔ FORBIDDEN	2019-01-08 0...	KB3172522	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3172...	Important
⛔ FORBIDDEN	2019-01-08 0...	KB4461594	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Word 2013 (KB4461...	Important
✓ ALLOWED	2019-01-08 0...	KB4461537	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4461...	Important
✓ ALLOWED	2018-12-11 0...	KB4461481	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft PowerPoint 2013 (K...	Important
✓ ALLOWED	2018-12-11 0...	KB4461559	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Excel 2013 (KB44615...	Important
✓ ALLOWED	2018-11-13 0...	KB4461489	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Project 2013 (KB446...	Important
✓ ALLOWED	2018-11-13 0...	KB4461487	Office 2013	Security Up...	Mise à jour de sécurité pour Skype for Business 2015 (KB44...	Low
✓ ALLOWED	2018-11-13 0...	KB4022237	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4022...	Important

Package filename: C:\Users\sfonteneau-adm\AppData\

Fig. 128: Creating a *waptwua* Package

Windows Updates list tab

The *Windows Update List* tab lists all needed Windows Updates.

Important: The server does not scan the `wsussc2.cab` itself, it lets the WUA wapt agents do it. If an update seems to you as missing from the list, you must run a scan on one of the machines present in the console. If you run a WUA scan on a Windows 7 agent, the CAB and Windows 7 files will be displayed on the Windows Update List tab.

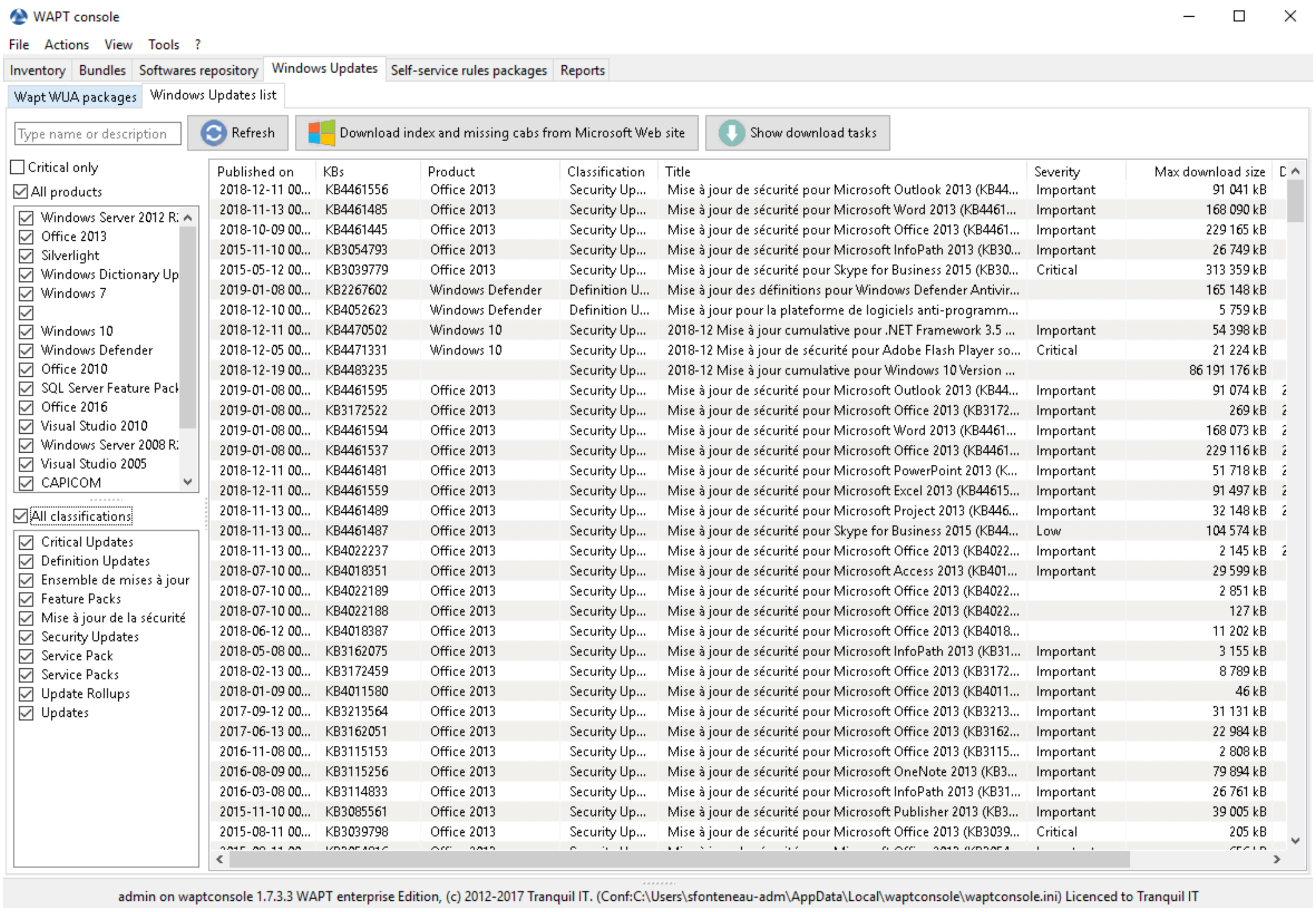
The left pane displays updates categories, allowing you to filter by:

- criticality;
- product;
- classification;

In the right panel grid, if the *Downloaded on* column is empty, it means that the update has not yet been downloaded by the WAPT server and is not present on the WAPT server (This update is not missing on any host).

- you can force the download of an update by *right-clicking* → *Download*;
- you can also force the download of the `wsusscn2.cab` file with the *Download WSUSScan cab from Microsoft Web Site* button;
- you can see the Windows Updates download on the server with the *Show download task* button;

Hint: To cleanup your `waptwua` folder, you can remove no longer needed Windows updates. WAPT server will only re-download deleted updates if one of the WAPT equipped hosts requests it;

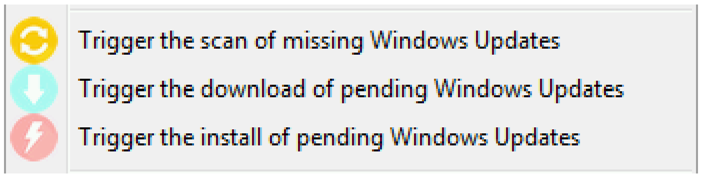


Published on	KBs	Product	Classification	Title	Severity	Max download size	D
2018-12-11 00...	KB4461556	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Outlook 2013 (KB44...	Important	91 041 kB	
2018-11-13 00...	KB4461485	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Word 2013 (KB4461...	Important	168 090 kB	
2018-10-09 00...	KB4461445	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4461...	Important	229 165 kB	
2015-11-10 00...	KB3054793	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft InfoPath 2013 (KB30...	Important	26 749 kB	
2015-05-12 00...	KB3039779	Office 2013	Security Up...	Mise à jour de sécurité pour Skype for Business 2015 (KB30...	Critical	313 359 kB	
2019-01-08 00...	KB2267602	Windows Defender	Definition U...	Mise à jour des définitions pour Windows Defender Antivir...		165 148 kB	
2018-12-10 00...	KB4052623	Windows Defender	Definition U...	Mise à jour pour la plateforme de logiciels anti-programm...		5 759 kB	
2018-12-11 00...	KB4470502	Windows 10	Security Up...	2018-12 Mise à jour cumulative pour .NET Framework 3.5 ...	Important	54 398 kB	
2018-12-05 00...	KB4471331	Windows 10	Security Up...	2018-12 Mise à jour de sécurité pour Adobe Flash Player so...	Critical	21 224 kB	
2018-12-19 00...	KB4483235		Security Up...	2018-12 Mise à jour cumulative pour Windows 10 Version ...		86 191 176 kB	
2019-01-08 00...	KB4461595	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Outlook 2013 (KB44...	Important	91 074 kB	2
2019-01-08 00...	KB3172522	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3172...	Important	269 kB	2
2019-01-08 00...	KB4461594	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Word 2013 (KB4461...	Important	168 073 kB	2
2019-01-08 00...	KB4461537	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4461...	Important	229 116 kB	2
2018-12-11 00...	KB4461481	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft PowerPoint 2013 (K...	Important	51 718 kB	2
2018-12-11 00...	KB4461559	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Excel 2013 (KB44615...	Important	91 497 kB	2
2018-11-13 00...	KB4461489	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Project 2013 (KB446...	Important	32 148 kB	2
2018-11-13 00...	KB4461487	Office 2013	Security Up...	Mise à jour de sécurité pour Skype for Business 2015 (KB44...	Low	104 574 kB	
2018-11-13 00...	KB4022237	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4022...	Important	2 145 kB	2
2018-07-10 00...	KB4018351	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Access 2013 (KB401...	Important	29 599 kB	
2018-07-10 00...	KB4022189	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4022...		2 851 kB	
2018-07-10 00...	KB4022188	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4022...		127 kB	
2018-06-12 00...	KB4018387	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4018...		11 202 kB	
2018-05-08 00...	KB3162075	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft InfoPath 2013 (KB31...	Important	3 155 kB	
2018-02-13 00...	KB3172459	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3172...	Important	8 789 kB	
2018-01-09 00...	KB4011580	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB4011...	Important	46 kB	
2017-09-12 00...	KB3213564	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3213...	Important	31 131 kB	
2017-06-13 00...	KB3162051	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3162...	Important	22 984 kB	
2016-11-08 00...	KB3115153	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3115...	Important	2 808 kB	
2016-08-09 00...	KB3115256	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft OneNote 2013 (KB3...	Important	79 894 kB	
2016-03-08 00...	KB3114833	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft InfoPath 2013 (KB31...	Important	26 761 kB	
2015-11-10 00...	KB3085561	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Publisher 2013 (KB3...	Important	39 005 kB	
2015-08-11 00...	KB3039798	Office 2013	Security Up...	Mise à jour de sécurité pour Microsoft Office 2013 (KB3039...	Critical	205 kB	

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

Launch WUA on clients

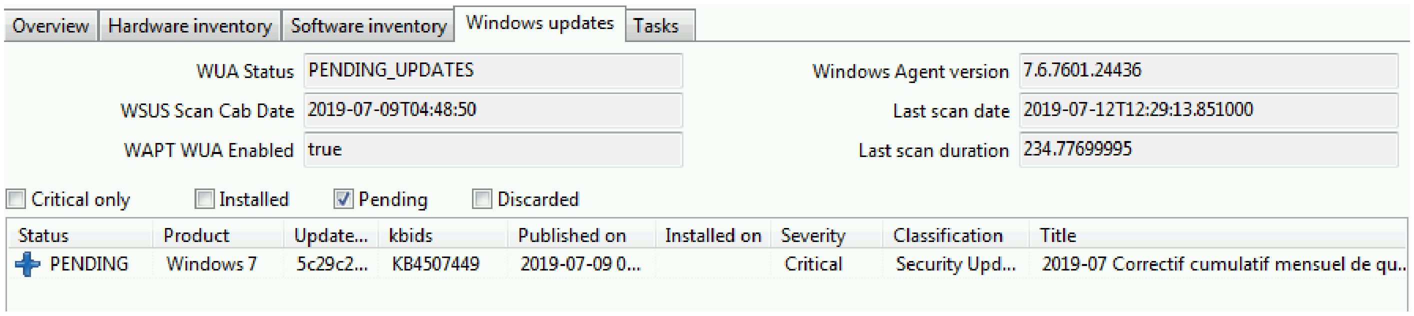
From the console you have three options.


Trigger the scan of missing Windows Updates
Trigger the download of pending Windows Updates
Trigger the install of pending Windows Updates

The *Trigger the scan of pending Windows Updates* button will launch the scan on the client and list all updates flagged for the OS. You can scan the client from the console like that or by using **`wapt-get waptwua-scan`** from the command-line.

Hint: Every 30 minutes, the WAPT Server will look for updates that have been requested at least once by WAPT Clients and that have not yet been downloaded and cached. If an update is pending, the WAPT Server will download it from official Microsoft servers.

You can force this scan with the *Download index and missing cabs from Microsoft Web site* button in tab *Windows Updates* → *Windows Updates list*



WUA Status	PENDING_UPDATES	Windows Agent version	7.6.7601.24436
WSUS Scan Cab Date	2019-07-09T04:48:50	Last scan date	2019-07-12T12:29:13.851000
WAPT WUA Enabled	true	Last scan duration	234.77699995

☐ Critical only ☐ Installed ☑ Pending ☐ Discarded

Status	Product	Update...	kbids	Published on	Installed on	Severity	Classification	Title
PENDING	Windows 7	5c29c2...	KB4507449	2019-07-09 0...		Critical	Security Upd...	2019-07 Correctif cumulatif mensuel de qu..

If you want to download from the console, use the *Trigger the download of pending Windows Updates* button.

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.

If you want to install the pending update(s), use **`wapt-get waptwua-install`** from the command-line prompt.

If you want to trigger the installation from the console, click on *Trigger the install of pending Windows Updates* button.

Hint: When you want to install the pending updates stored in cache, the WAPT Service triggers the WUA service.

The WAPT Service will enable and start the WUA Service temporarily to install the updates. When updates are installed, waptservice will stop and disable the WUA service until the next cycle.

Video demonstration

New in version 1.7: Enterprise

6.8.4 Using the reporting functions in WAPT

Hint: Feature only available with WAPT **Enterprise**.

Working principle

WAPT **Enterprise** offers advanced reporting capabilities.

Indeed, who better than you to know what you want in your report.

With WAPT we offer to write your own SQL queries to display the result in the wapt console.

WAPT query Designer

The query designer offers you the ability to edit your own queries on the WAPT PostgreSQL database.

To create a new report, click on *Reporting* → *Design Mode* → *New query*.

Hint:

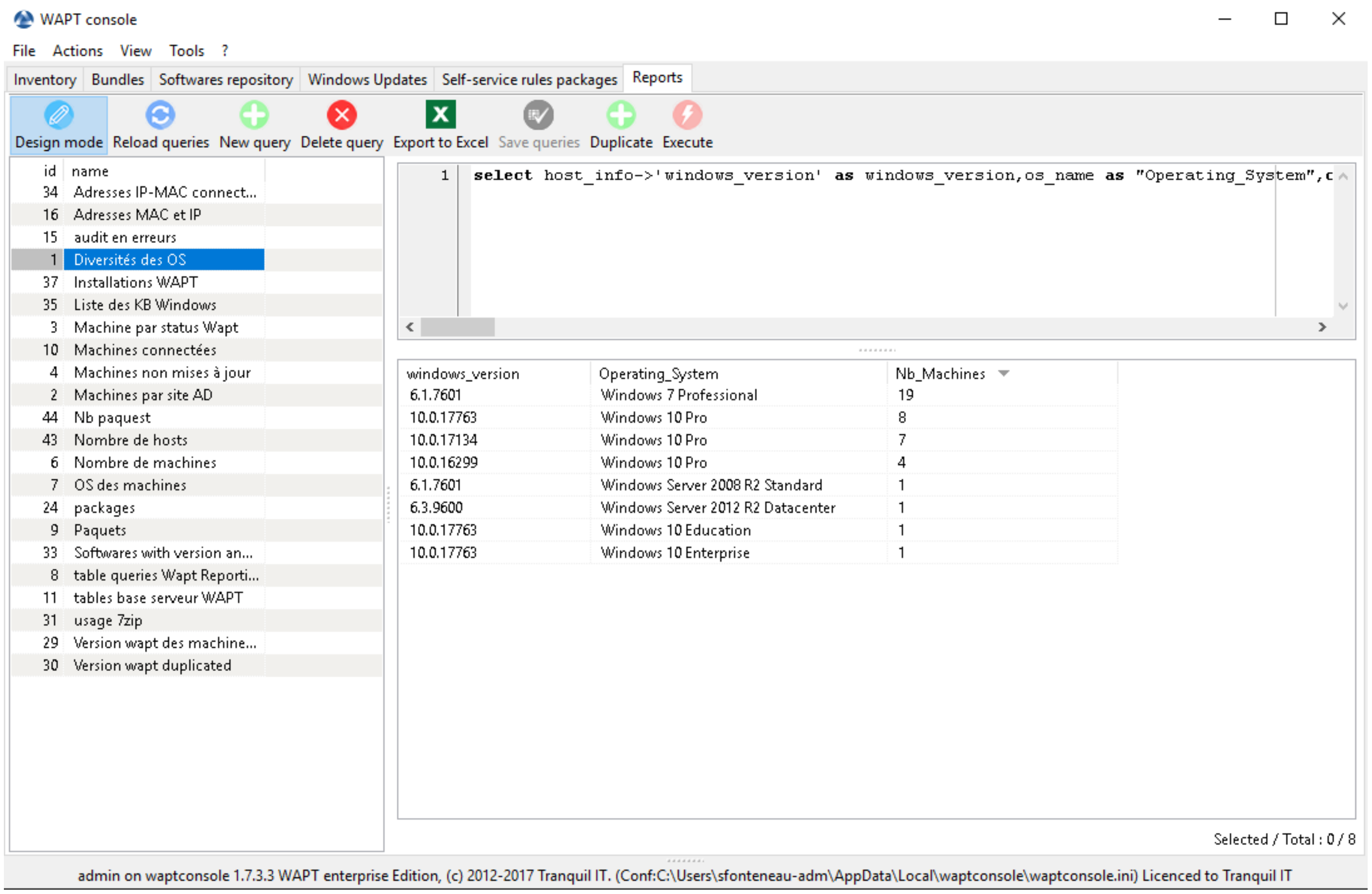
- to rename a query, press the `F2` key;
- in the top banner, you can write your SQL query;

To edit / modify / save your reports:

- the *Reload queries* button is used to reload queries saved on the server, for example, if a colleague has just edited a new query;
- the *New query* button will add a new blank query to the list;
- the *Delete query* button will delete the selected query from the WAPT server;
- the *Export to Excel* button will export the result of your query to a spreadsheet;
- the *Save queries* button will save your query to the WAPT server;
- the *Duplicate* button will duplicate an existing query to avoid writing a request from scratch;
- the *Execute* button executes the selected query;

Note:

- the queries are saved in the PostgreSQL WAPT database;
- the shortcut `CTRL+space` allows you to build your queries more effectively;


WAPT console
File Actions View Tools ?
Inventory Bundles Softwares repository Windows Updates Self-service rules packages Reports
Design mode Reload queries New query Delete query Export to Excel Save queries Duplicate Execute

id	name
34	Adresses IP-MAC connect...
16	Adresses MAC et IP
15	audit en erreurs
1	Diversités des OS
37	Installations WAPT
35	Liste des KB Windows
3	Machine par status Wapt
10	Machines connectées
4	Machines non mises à jour
2	Machines par site AD
44	Nb paquest
43	Nombre de hosts
6	Nombre de machines
7	OS des machines
24	packages
9	Paquets
33	Softwares with version an...
8	table queries Wapt Reporti...
11	tables base serveur WAPT
31	usage 7zip
29	Version wapt des machine...
30	Version wapt duplicated

```
select host_info->'windows_version' as windows_version,os_name as "Operating_System",c
```

windows_version	Operating_System	Nb_Machines
6.1.7601	Windows 7 Professional	19
10.0.17763	Windows 10 Pro	8
10.0.17134	Windows 10 Pro	7
10.0.16299	Windows 10 Pro	4
6.1.7601	Windows Server 2008 R2 Standard	1
6.3.9600	Windows Server 2012 R2 Datacenter	1
10.0.17763	Windows 10 Education	1
10.0.17763	Windows 10 Enterprise	1

Selected / Total : 0 / 8

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

Fig. 129: Designing a query in WAPT reporting

Query examples

Computers query

- Number of hosts

```
select count(*) as "Nb_Machines" from hosts
```

- Computers list:

```
select computer_name,os_name,os_version,os_architecture,serialnr from hosts order by 4,3,1
```

- Computers MAC addresses and IP:

```
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
```

- Windows versions:

```
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
```

- Lists OS diversity

```
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
```

- List hosts not seen in a while

```
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 hosts by Chassis types

```
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')
```

(continues on next page)

(continued from previous page)

```
end as type_chassis,
string_agg(distinct coalesce(manufacturer,'?') ||' '|| coalesce(productname,''),', '),
count(*) as "Nb_Machines" from hosts
group by 1
```

- List hosts with their Windows Serial Key

```
select computer_name,os_name,os_version,host_info->'windows_product_infos'->'product_key' as␣
↪windows_product_key from hosts order by 3,1
```

WAPT query

- List WAPT Packages in WAPT server repository

```
select package,version,architecture,description,section,package_uuid,count(*)
from packages
group by 1,2,3,4,5,6
```

- List hosts needing upgrade

```
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
```

Packages query

- List packages with their number of 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
```

Software query

- WAPT Community agents

```
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 %%')
```

- List hosts with their 7zip version associated

```
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
```

- List hosts with their softwares

```
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
```

- List normalized softwares

```
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
```

You can also find several more examples of queries on Tranquil IT's Forum.

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.

Normalizing software names

Sometimes, the version of the software or its architecture are an integral part of the software name. When they register with the WAPT Server inventory, they appear as different software whereas they are just one software for us humans.

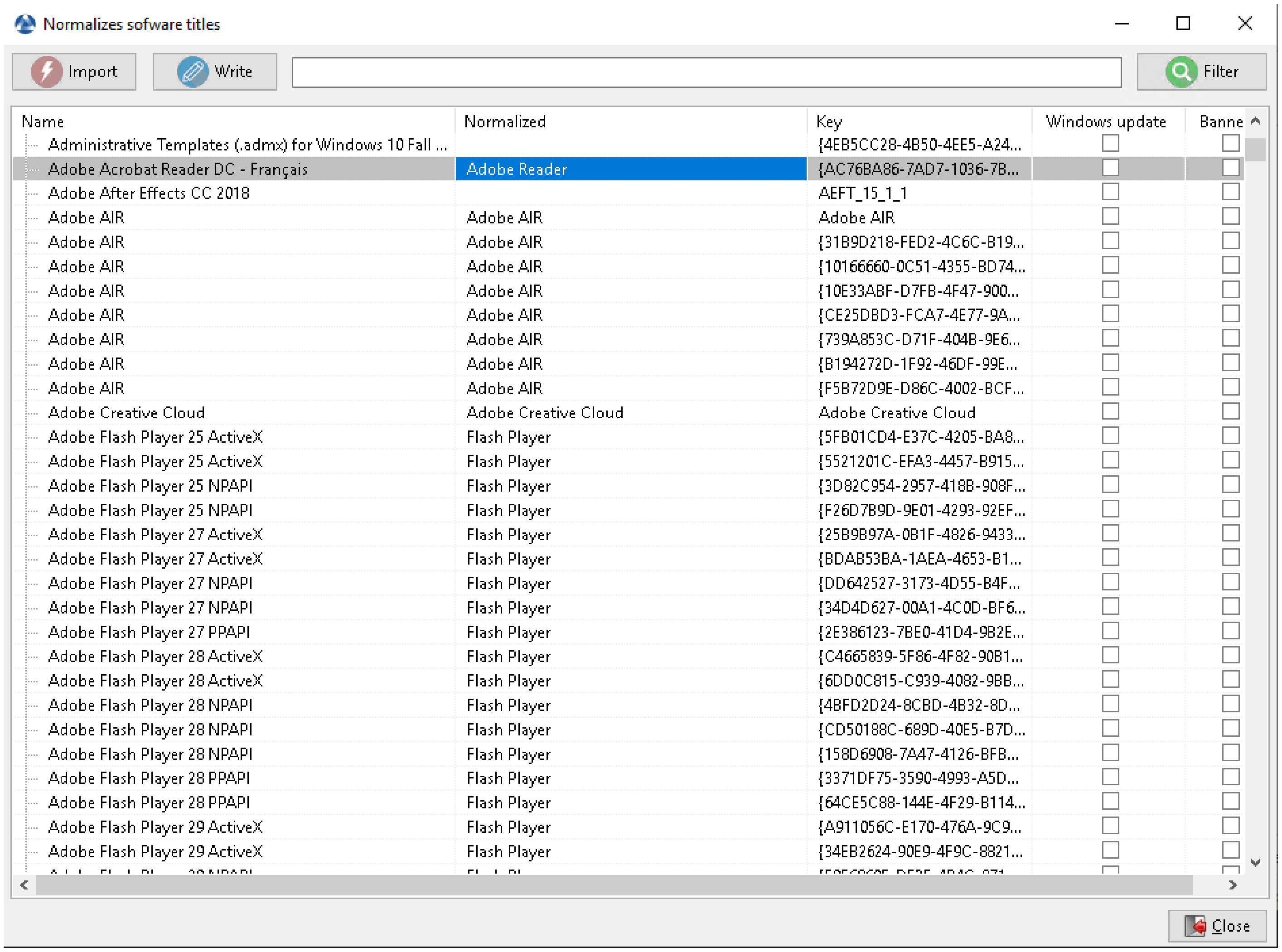
To solve this problem, we propose to standardize the name of the software with WAPT.

- click *Normalize Software Names* in the *Tools* menu;
- select the software to standardize, for example, all different version of Adobe Flash Player;
- on the column *normalized*, press `F2` to assign a standardized name to the selected software. Then press `Enter`;

Note:

- to select several programs, select them with the `shift-up/down` key combination;
- you can also indicate a software like *windows update* or *banned* (Press `spacebar` in the corresponding column);

- press on *Import* to load the changes from the server;
- press on *Write* to save your changes;



Name	Normalized	Key	Windows update	Banne
Administrative Templates (.admx) for Windows 10 Fall ...		{4EB5CC28-4B50-4EE5-A24...	☐	☐
Adobe Acrobat Reader DC - Français	Adobe Reader	{AC76BA86-7AD7-1036-7B...	☐	☐
Adobe After Effects CC 2018		AEFT_15_1_1	☐	☐
Adobe AIR	Adobe AIR	Adobe AIR	☐	☐
Adobe AIR	Adobe AIR	{31B9D218-FED2-4C6C-B19...	☐	☐
Adobe AIR	Adobe AIR	{10166660-0C51-4355-BD74...	☐	☐
Adobe AIR	Adobe AIR	{10E33ABF-D7FB-4F47-900...	☐	☐
Adobe AIR	Adobe AIR	{CE25DBD3-FCA7-4E77-9A...	☐	☐
Adobe AIR	Adobe AIR	{739A853C-D71F-404B-9E6...	☐	☐
Adobe AIR	Adobe AIR	{B194272D-1F92-46DF-99E...	☐	☐
Adobe AIR	Adobe AIR	{F5B72D9E-D86C-4002-BCF...	☐	☐
Adobe Creative Cloud	Adobe Creative Cloud	Adobe Creative Cloud	☐	☐
Adobe Flash Player 25 ActiveX	Flash Player	{5FB01CD4-E37C-4205-BA8...	☐	☐
Adobe Flash Player 25 ActiveX	Flash Player	{5521201C-EFA3-4457-B915...	☐	☐
Adobe Flash Player 25 NPAPI	Flash Player	{3D82C954-2957-418B-908F...	☐	☐
Adobe Flash Player 25 NPAPI	Flash Player	{F26D7B9D-9E01-4293-92EF...	☐	☐
Adobe Flash Player 27 ActiveX	Flash Player	{25B9B97A-0B1F-4826-9433...	☐	☐
Adobe Flash Player 27 ActiveX	Flash Player	{BDAB53BA-1AEA-4653-B1...	☐	☐
Adobe Flash Player 27 NPAPI	Flash Player	{DD642527-3173-4D55-B4F...	☐	☐
Adobe Flash Player 27 NPAPI	Flash Player	{34D4D627-00A1-4C0D-BF6...	☐	☐
Adobe Flash Player 27 PPAPI	Flash Player	{2E386123-7BE0-41D4-9B2E...	☐	☐
Adobe Flash Player 28 ActiveX	Flash Player	{C4665839-5F86-4F82-90B1...	☐	☐
Adobe Flash Player 28 ActiveX	Flash Player	{6DD0C815-C939-4082-9BB...	☐	☐
Adobe Flash Player 28 NPAPI	Flash Player	{4BFD2D24-8CBD-4B32-8D...	☐	☐
Adobe Flash Player 28 NPAPI	Flash Player	{CD50188C-689D-40E5-B7D...	☐	☐
Adobe Flash Player 28 NPAPI	Flash Player	{158D6908-7A47-4126-BFB...	☐	☐
Adobe Flash Player 28 PPAPI	Flash Player	{3371DF75-3590-4993-A5D...	☐	☐
Adobe Flash Player 28 PPAPI	Flash Player	{64CE5C88-144E-4F29-B114...	☐	☐
Adobe Flash Player 29 ActiveX	Flash Player	{A911056C-E170-476A-9C9...	☐	☐
Adobe Flash Player 29 ActiveX	Flash Player	{34EB2624-90E9-4F9C-8821...	☐	☐

Fig. 130: Normalizing the name of software

You can now run your queries on this standardized name.

Video demonstration

6.8.5 Using WAPT SelfService

New in version 1.7: Enterprise

Presentation

With WAPT 1.7 **Enterprise** you can now filter the list of self-service packages available for your users.

Your users will be able to install a selection of WAPT packages without having to be a *Local Administrator* on their desktop.

The *Users* gain in autonomy while deploying software and configurations that are trusted and authorized by the *Organization*. This is a time saving feature for the Organization's IT support Helpdesk.

How does it work?

With WAPT 1.7 **Enterprise**, a new type of WAPT package exists beside *base*, *group*, *host*, *profile* and *unit* packages: they are **selfservice** packages.

A *selfservice* package may now be deployed on hosts to list the different self-service rules that apply to the host.

How to use the selfservice feature?

Hint: The **selfservice** feature is only available with WAPT **Enterprise**.

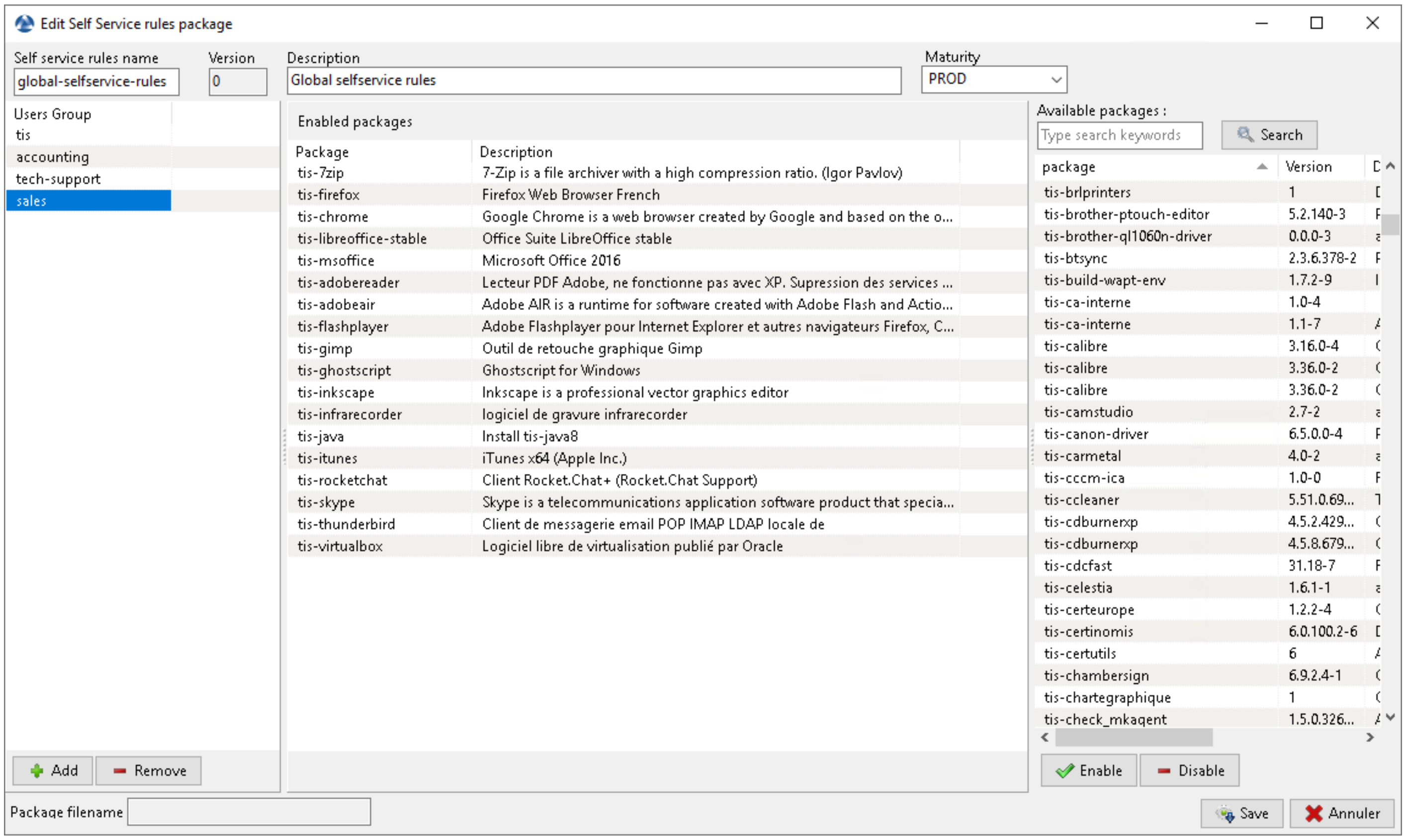
In the **Community** version, only Local Administrators and members of the *waptselfservice* group can access self-service on the agent.

In the **Community** version, it is not possible to filter the packages made accessible to the user.

In the console go to the tab *Self-service* rules.

You can now create your first *selfservice* rule package.

- give a name to your new *selfservice* package;
- click on *Add* to add an Active Directory group (at the bottom left);
- name the *selfservice* group (with `F2` or type directly into the cell);



Edit Self Service rules package

Self service rules name	global-selfservice-rules
Version	0
Description	Global selfservice rules
Maturity	PROD

Users Group
- tis
- accounting
- tech-support
- sales

Enabled packages

Package	Description
tis-7zip	7-Zip is a file archiver with a high compression ratio. (Igor Pavlov)
tis-firefox	Firefox Web Browser French
tis-chrome	Google Chrome is a web browser created by Google and based on the o...
tis-libreoffice-stable	Office Suite LibreOffice stable
tis-msoffice	Microsoft Office 2016
tis-adobereader	Lecteur PDF Adobe, ne fonctionne pas avec XP. Supression des services ...
tis-adobeair	Adobe AIR is a runtime for software created with Adobe Flash and Actio...
tis-flashplayer	Adobe Flashplayer pour Internet Explorer et autres navigateurs Firefox, C...
tis-gimp	Outil de retouche graphique Gimp
tis-ghostscript	Ghostscript for Windows
tis-inkscape	Inkscape is a professional vector graphics editor
tis-infrarecorder	logiciel de gravure infrarecorder
tis-java	Install tis-java8
tis-itunes	iTunes x64 (Apple Inc.)
tis-rocketchat	Client Rocket.Chat+ (Rocket.Chat Support)
tis-skype	Skype is a telecommunications application software product that specia...
tis-thunderbird	Client de messagerie email POP IMAP LDAP locale de
tis-virtualbox	Logiciel libre de virtualisation publié par Oracle

Available packages :

Type search keywords

package	Version	D
tis-brlprinters	1	[
tis-brother-ptouch-editor	5.2.140-3	F
tis-brother-ql1060n-driver	0.0.0-3	a
tis-btsync	2.3.6.378-2	F
tis-build-wapt-env	1.7.2-9	I
tis-ca-interne	1.0-4	
tis-ca-interne	1.1-7	A
tis-calibre	3.16.0-4	(
tis-calibre	3.36.0-2	(
tis-calibre	3.36.0-2	(
tis-camstudio	2.7-2	a
tis-canon-driver	6.5.0.0-4	F
tis-carmetal	4.0-2	a
tis-cccm-ica	1.0-0	F
tis-ccleaner	5.51.0.69...	T
tis-cdburnerxp	4.5.2.429...	(
tis-cdburnerxp	4.5.8.679...	(
tis-cdcfast	31.18-7	F
tis-celestia	1.6.1-1	a
tis-certeurope	1.2.2-4	(
tis-certinomis	6.0.100.2-6	[
tis-certutils	6	A
tis-chambersign	6.9.2.4-1	(
tis-chartegraphique	1	(
tis-check_mkagent	1.5.0.326...	A

Fig. 131: Create a *selfservice* package

- drag the allowed software and configuration packages for this *selfservice* group into the central column;
- add as many groups as you want in the package;
- save the package and deploy the package on your selection of hosts;
- once the package is deployed, only allowed packages listed in the *selfservice* group(s) of which the *User* is a member will be shown to the logged in *User*;

Note:

- if a group appears in multiple *selfservice* packages, then the rules are merged;
- the authentication used is system authentication, local users and groups, but if the machine is in a domain then authentication and groups will also work with users and groups in the domain;

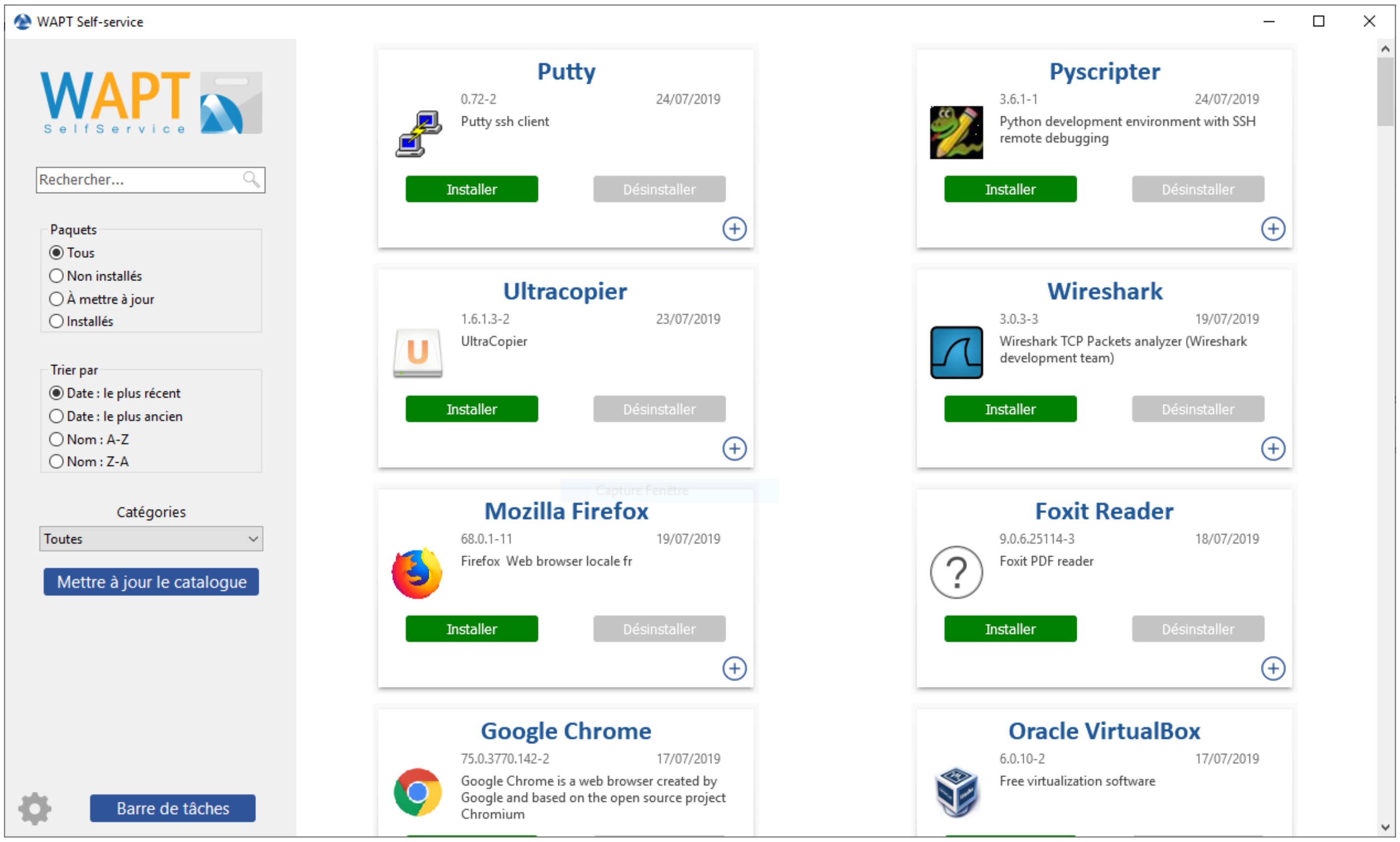
How to use the self-service on the user station?

The self-service is accessible to users in the start menu under the name *Self-Service software WAPT*.

It is also available directly in `<base>\waptself.exe`.

The login and password to enter when launching the self-service are the User's credentials (local or Active Directory credentials).

The self-service then displays a list of packages available for installation.



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

Package	Version	Date	Description
Putty	0.72-2	24/07/2019	Putty ssh client
Pyscripter	3.6.1-1	24/07/2019	Python development environment with SSH remote debugging
Ultracopier	1.6.1.3-2	23/07/2019	UltraCopier
Wireshark	3.0.3-3	19/07/2019	Wireshark TCP Packets analyzer (Wireshark development team)
Mozilla Firefox	68.0.1-11	19/07/2019	Firefox Web browser locale fr
Foxit Reader	9.0.6.25114-3	18/07/2019	Foxit PDF reader
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

- the user can have more details on each package with the + icon;
- different filters are available for the user on the left side panel;
- the *Update Catalog* button is used to force a **`wapt-get update`** on the WAPT agent;
- the list of package categories is displayed to the user. To add a category to the list, you must specify the category in the *categories* section of the `control` file of the relevant package;
- the current task list of the WAPT agent is available with the *task bar* button;
- it is possible to change the language of the interface with the *configuration* button at the bottom left.

Customizing the Self Service interface

Adding the Logo of your Organisation

In the **Enterprise version only of WAPT**, it is possible to change the logo that appears in the self-service interface and therefore improve the acceptation of the Self Service feature by your users.

To do this, simply place the logo you want in `<wapt>\templates\waptself-logo.png`

Note: It is highly recommended to use a *`.png`* file with a *200 x 150px* resolution.

Managing package categories

Default categories are:

- Internet;
- Utilities;
- Messaging;
- Security;
- System and network;
- Storage;
- Media;
- Development;
- Office;

You can create your own categories easily by filling the `control` file's `categories` section of any WAPT package and write a new category of your choice, WAPT will automatically show the package in the new category.

WAPT Agent Settings for WAPT Self-Service

WAPT Agent can be configured to force WAPT SelfService packages filtering to Local Administrators *Settings for WAPT Self-Service and Waptservice Authentification*.

Configuring a different authentication method for the selfservice

As mentioned above, authentication on WAPT service is configured by default in system mode.

This means that the WAPT service transmits the authentication directly to the operating system; it also recovers the groups by directly interrogating the operating system.

This behavior is defined with the value of `service_auth_type` in `wapt-get.ini`. The default value is *system*.

In this mode we assume that Local Administrators can see all the packages. To change this behavior, modify the value of `waptservice_admin_filter` in `wapt-get.ini`.

You may be interested in looking up this article describing the *settings for WAPT Self-Service and Waptservice Authentification* for more options.

Two additional modes are available starting with version 1.8.2:

- `waptserver-ldap`: this mode allows authentication to the WAPT server. The WAPT server will make a LDAP request to verify authentication and groups. **Warning** ! For this to work, you must have configured LDAP authentication on the WAPT server, (the configuration of the admin group will be ignored) See *this article on configuring authentication against Active Directory* for more information.
- `waptagent-ldap`, This mode allows authentication with an LDAP server identified in `wapt-get.ini`. The WAPT agent will make a LDAP request to verify authentication and groups.

  You may be interested in looking up this article describing the *settings for WAPT Self-Service and Waptservice Authentification* for more options.

Note: For the system authentication under to work correctly, be sure to correctly configure your pam authentication and your `nsswitch.conf`. The **`id username`** command must return the list of the groups the user is member of.

Video demonstration

New in version 1.5: Enterprise

6.8.6 Differentiating the role level in WAPT

Hint: Feature only available with WAPT Enterprise

Introduction

WAPT offers the possibility to differentiate administrator roles based on a PKI to sign packages and actions.

Hint: The following description of roles differentiation is temporary as it will evolve in the near future.

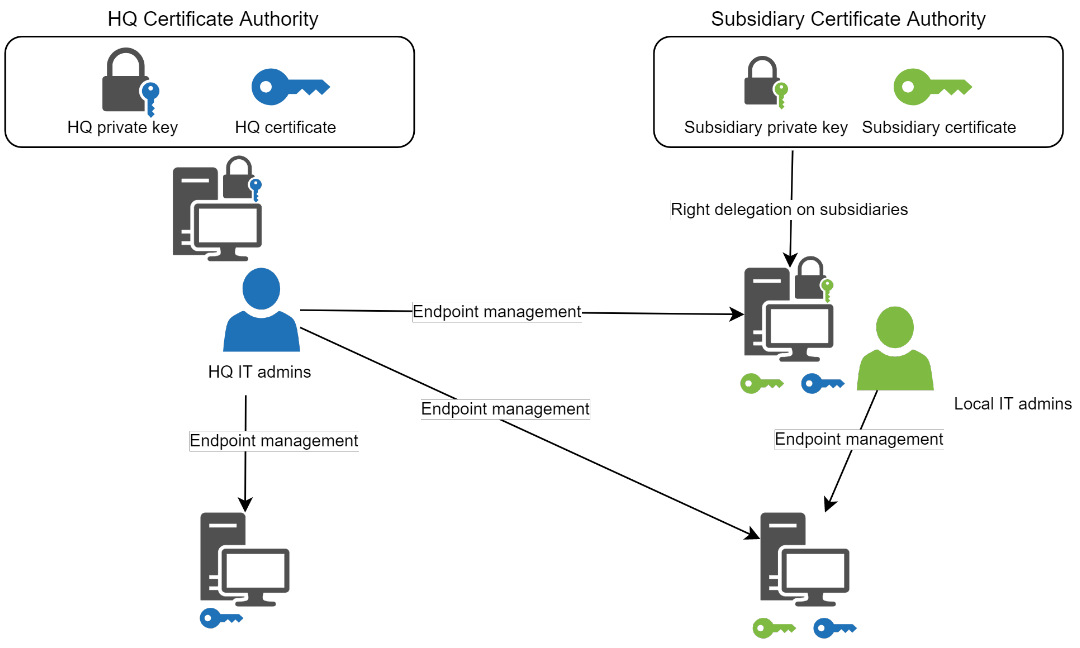


Fig. 132: WAPT admin users roles differentiation

There are three cases:

Private key + certificate types	Key usages
Simple private key + certificate	Allows authentication on WAPT console + interactions with WAPT agents
Developer private key + certificate	Allows authentication on WAPT console + interactions with WAPT agents + package signing
Certificate Authority (CA) private key + certificate	Allows authentication + interactions + package signing + private key issuing

Common WAPT install will generate a CA private key by default, allowing private key issuing for developers and package signing.

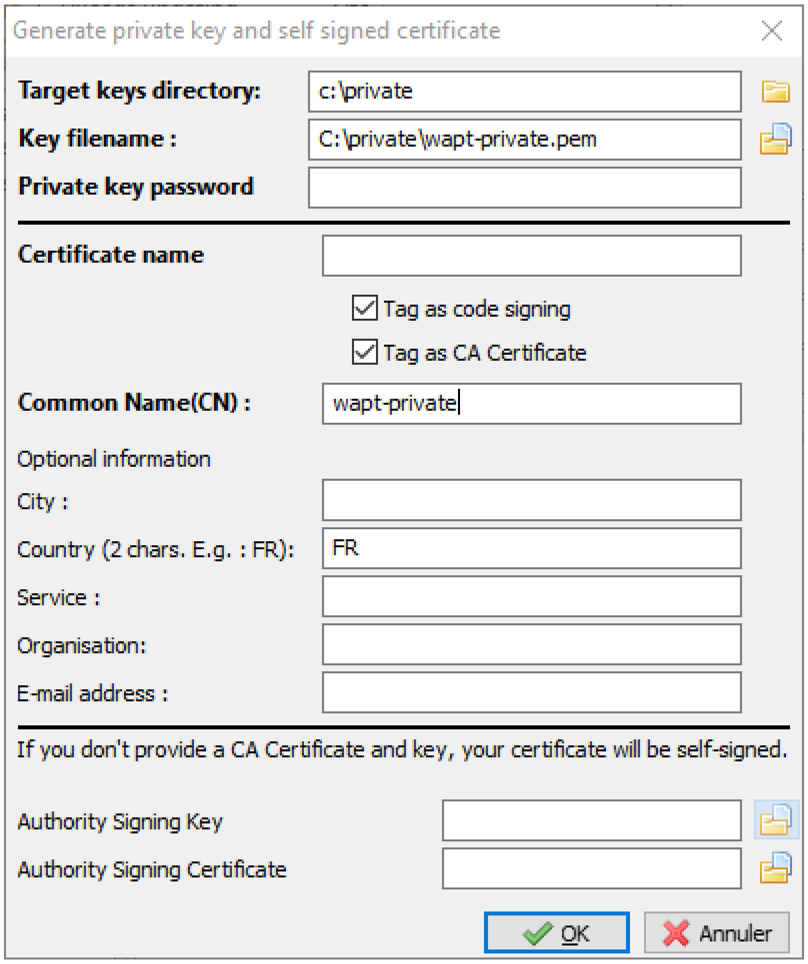
It is possible to emit a Certificate Authority for each subsidiaries. It is then possible to issue a personal private key and its corresponding certificate to each IT admins.

By looking at the above schematics, we can deduce the following conclusion:

- WAPT agents in HQ can be managed by HQ IT team and cannot be managed by subsidiaries IT teams;
- WAPT agents in the subsidiary having both certificates, from HQ and subsidiary, can be managed by local IT team and by HQ IT team;

The usage of an existing PKI is possible, WAPT Console comes with a simple certificate generator.

Generating a new certificate



Generate private key and self signed certificate

Target keys directory:	c:\private
Key filename :	C:\private\wapt-private.pem
Private key password	
Certificate name	
	[✓] Tag as code signing
	[✓] Tag as CA Certificate
Common Name(CN) :	wapt-private

Optional information

City :	
Country (2 chars. E.g. : FR):	FR
Service :	
Organisation:	
E-mail address :	

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key	
Authority Signing Certificate	

Fig. 133: Generating a new self-signed certificate

Generating the Certificate Authority (CA)

When installing WAPT, you are asked to create a `.pem` / `.crt` pair by checking the boxes *Certificate CA* and *Code Signing*.

This crt/ pem pair will allow to sign WAPT packages and new certificates.

Generating a new certificate with the Certificate Authority

To create a new pem/ crt pair from the private key, click on *Create a certificate*.

Note: The new certificate will not be a self-signed certificate;

This new certificate will be signed by the AC (the key generated at the time of the first installation of WAPT);

You must then fill in the *AC's certificate* and the *AC's key*.

When generating the new pem/ crt pair, you have the option to choose whether or not the new certificate will be a **Code Signing** type.

Hint: For recall, a *Code Signing* certificate is reserved to individuals with the *Administrator* role in the context of WAPT and a simple SSL certificate without the `Code Signing` attribute is reserved to individuals with the role of *Package Deployer*.

Administrators will be authorized to sign packages that **CONTAIN** a `setup.py` executable file (i.e. *base* packages).

Individuals with the *Package Deployer* role will be authorized to sign packages that **DO NOT CONTAIN** `setup.py` executable file (i.e. *host*, *unit* and *group* packages).

Keys and certificates that are **Not Code Signing** may be distributed to individuals in charge of deploying packages on the installed base of WAPT equipped devices.

Another team with certificates having the **Code Signing** attribute will prepare the WAPT packages that contain applications that will need to be configured according to the *Organization*'s security guidelines and the user customizations desired by her.

Generating a new prm/ crt pair will also allow to formally identify the individual who has signed a package by looking up the WAPT package certificate's CN attribute.

Hint: The new certificates will not be *CA Certificates*, which means that they will not be authorized to sign other certificates.

As a general rule, there is only one **CA Certificate** pem / crt pair per *Organization*.

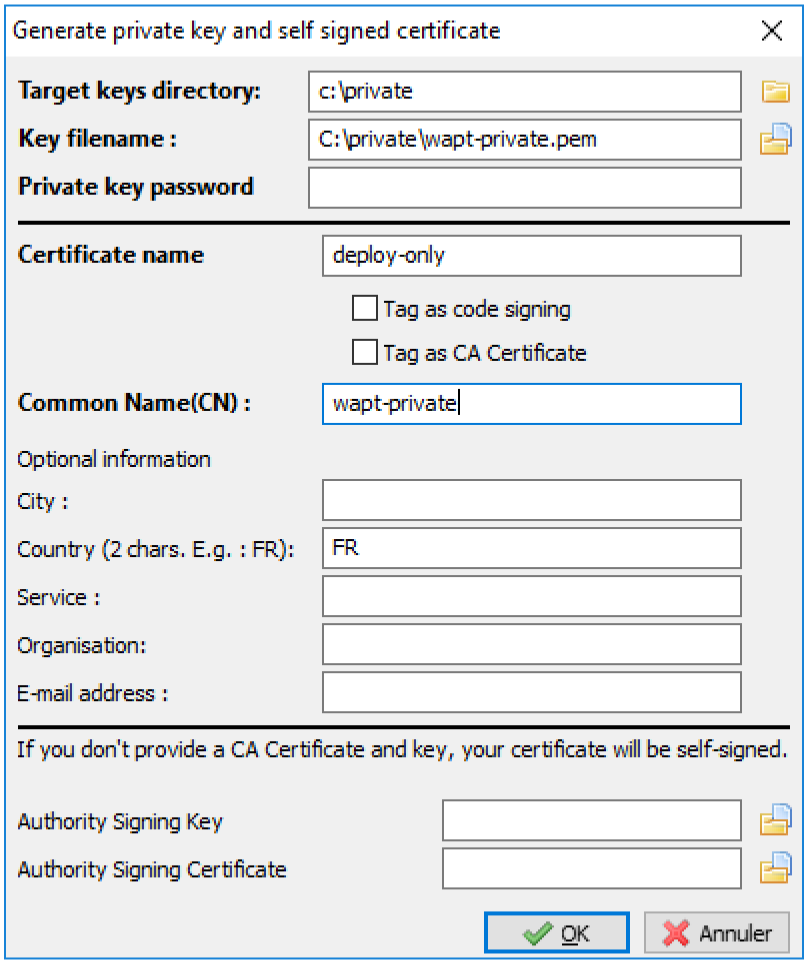
Deploying certificates of local IT admins on client

Some Organisations will choose to let local IT administrators perform actions on WAPT equipped devices by issuing them personnal certificates that will work on the set of devices for which the local IT admins are responsible.

The headquarter IT admins will deploy the certificates of local IT admins on the computers that local admins manage on their respective sites.

This way, local IT admins will not be able to manage computers located in headquarters, but on their own sites only.

You will need to copy the certificates of allowed local IT admins on client in `C:\program files(x86)\wapt\ssl`.


Generate private key and self signed certificate

Target keys directory:	c:\private
Key filename :	C:\private\wapt-private.pem
Private key password	
Certificate name	deploy-only
	☐ Tag as code signing
	☐ Tag as CA Certificate
Common Name(CN) :	wapt-private

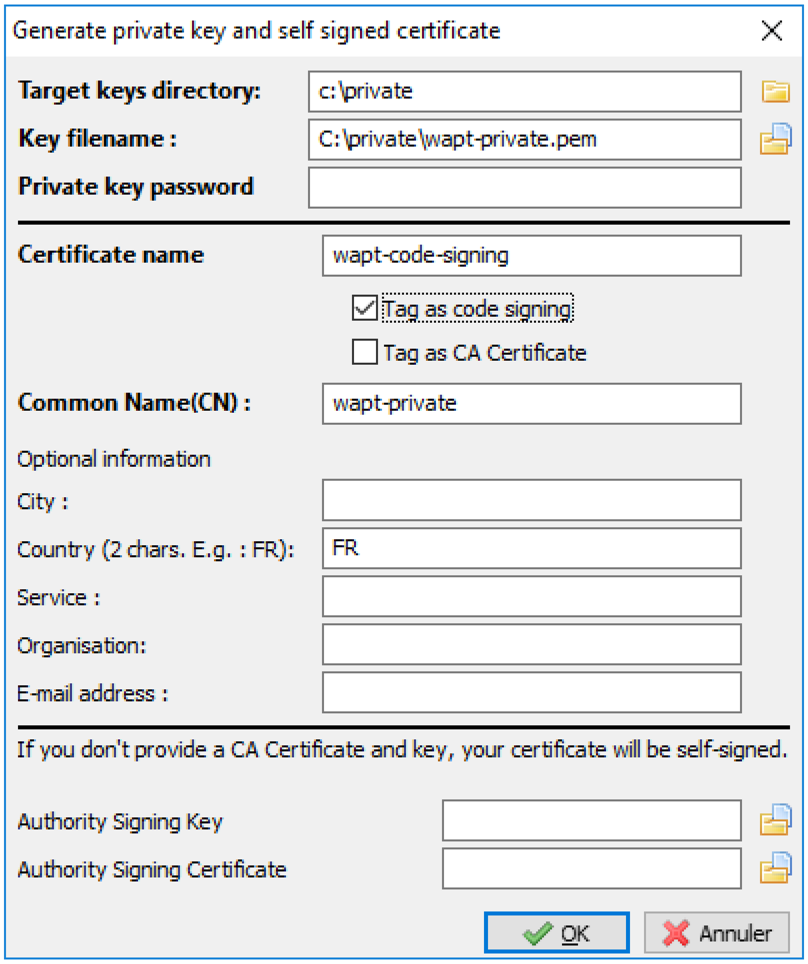
Optional information

City :	
Country (2 chars. E.g. : FR):	FR
Service :	
Organisation:	
E-mail address :	

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key	
Authority Signing Certificate	

Fig. 134: Generating a certificate without the *Code Signing* attribute



Generate private key and self signed certificate

Target keys directory:	c:\private
Key filename :	C:\private\wapt-private.pem
Private key password	
Certificate name	wapt-code-signing
	[✓] Tag as code signing
	[ ] Tag as CA Certificate
Common Name(CN) :	wapt-private
Optional information	
City :	
Country (2 chars. E.g. : FR):	FR
Service :	
Organisation:	
E-mail address :	

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Key	
Authority Signing Certificate	

Fig. 135: Generating a certificate with the *Code Signing* attribute

Hint: Do not forget to restart the WAPT service on clients for them to use their new certificate. Open a command line **cmd.exe** then:

```
net stop waptservice
net start waptservice
```

If you want to deploy the certificates using WAPT, below is an example of a package to deploy certificates on client computers.

```
# -*- 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 Upgrading the WAPT Server

Hint: If your WAPT Server is a virtual machine, take a snapshot of the VM. This way, you'll be able to go back easily in the rare case that the update fails.

Before upgrading WAPT Server, please refer to the following upgrading compatibility chart:

	To WAPT 1.5	To WAPT 1.6	To WAPT 1.7	To WAPT 1.8
From WAPT 1.3	Yes	Yes	No	No
From WAPT 1.5	•	Yes	Yes	No
From WAPT 1.6	•	•	Yes	No
From WAPT 1.7	•	•	•	Yes

6.9.1 Upgrading WAPT from 1.6/1.7 to 1.8

The upgrade process follows the process for a minor update:

- *minor update for Debian*;
- *minor update for CentOS*;
- *minor update for Windows*;

Attention:

- Debian Jessie is now deprecated. **WAPT 1.8 will not work with old Debian version**;
- consider migrating your existing WAPT installation to *Debian Buster or CentOS7*;

6.9.2 Upgrading WAPT from 1.5 to 1.6

The upgrade process follows the process for a minor update.

Note:

- if you are in Debian Stretch, it is recommended to upgrade to Debian Buster 64 bits: *Upgrading the Operating System*;
- this is **MANDATORY** for the **Enterprise** version with Windows Update support;
- when upgrading to Debian10, the PostgreSQL database must be upgraded;

6.9.3 Upgrading WAPT from 1.3 to 1.6

The upgrade from 1.3 to 1.6 is a major upgrade.

Changes between versions 1.3 and 1.6 and consequences

PostgreSQL has replaced MongoDB in WAPT

The database **`MongoDB`** has been replaced by **`PostgreSQL`** with JSON support.

The consequence is that **Administrators that have developed reports based on MongoDB will have to adapt their queries.**

Nginx is now the only supported web server in WAPT

Introducing Websockets in WAPT brings large benefits:

- status updates from WAPT agents are instantaneous in the WAPT console;
- when the *Administrator* launches immediate actions, it is no longer the WAPT Server that connects to the WAPT agent; the WAPT agent establishes and maintains a permanent connection with the WAPT Server;
- the Websockets avoid having to have a local listening port on the clients, improving the global security of WAPT equipped Organizations;

- since it is the WAPT agent that initiates and maintains a connection with the WAPT Server, connections travel through proxies and firewalls in a standard manner;

Nginx is the only web server capable of handling a very large quantity of Websockets. This technology is not something completely trivial to implement and maintain.

As a consequence, **support for IIS and Apache in WAPT is abandoned**.

Signature hashes are sha256 instead of sha1

Control sums for the list of files in the WAPT package and the signature of the attributes in the `control` file are now sha256 hashed instead of sha1 hashed to satisfy security requirements of cyberdefense agencies.

As a consequence, **all packages must be re-signed and the :file:`Packages` index files of all repositories must be regenerated**.

A script allows to massively re-sign all packages on the WAPT Server.

The host packages are now named after the UUID of the host

In versions <= 1.3.xx, host packages are named with the FQDN of the client host.

This poses some problems if the host is joined to another domain, or if the host is not joined to a domain, yet goes from a network to another with different DHCP settings.

In version >= 1.5, the *host* packages are named after the UUID of the client host, thus resolving the problems described above.

As a consequence, **host packages must be re-created and re-signed**.

A script on the WAPT Server allows to do this operation automatically.

Code Signing attribute for signing base packages

To differentiate between the roles of *Package Deployer* and *Package Developer*, the WAPT agent verifies the **Code Signing** attribute of the certificate that has signed the package.

If the package contains python executable code, i.e. a `setup.py` file, then the package must be signed with a certificate having the **Code Signing** attribute. **Otherwise, the package will not install**.

As a consequence, **it is necessary to generate and deploy a Code Signing Certificate on the WAPT equipped hosts**, and re-sign *base* packages (i.e. software packages) with a **Code Signing** certificate.

The certificate is deployed during the upgrade of the WAPT agent.

Upgrading from WAPT 1.3 to 1.6

Two methods are available:

- the *minimal upgrade* consists of just migrating the WAPT packages without the inventory. It is the simpler method;
- the *full upgrade* consists of migrating all data from the old WAPT Server;

Attention: Whether option 1 or 2 is chosen, there are two prerequisites:

- For the upgrade to work well, your Organization's WAPT **agents must all be in version 1.3.13 before upgrading** to 1.6. If not, the *waptupgrade* package will not work correctly. Otherwise, you may upgrade using GPOs;
- **all** your machines to be upgraded must have the *waptupgrade* package installed;

Hint: Tranquil IT offers fixed price contracts to help you upgrade from 1.3 to 1.6; you may contact us by calling +33(0)240 975 755.

Minimal upgrade

Note: This procedure also allows you to easily migrate your WAPT Server from Windows to Linux and vice versa.

This procedure is the simplest.

The minimal upgrade consists of migrating only WAPT base and group packages. Neither the inventory nor the host-packages will be kept.

- all informations stored in the WAPT database come from the WAPT agents, so no information will be lost;
- the inventory database will rebuild itself when the WAPT agent 1.6 equipped computers will re-register with the WAPT Server;

Migrating more easily from WAPT 1.3 to 1.6

This procedure is simpler than a complete migration, but some non essential data may be lost, breaking the chain of traceability.

- backup the WAPT files in:
  - Linux: `/var/www/wapt/` or `/var/www/html/wapt/`;
  - Windows: `C:\waptwaptserver\repository\wapt`;

Installing and re-importing the WAPT packages

If you are using a Windows based WAPT Server in 1.3.13, you must uninstall your 1.3.13 version before installing the 1.6 version.

You may now follow the same procedure to do a *normal installation of the WAPT Server*.

The WAPT Server must have the same DNS name as the previous WAPT Server (or the same IP if your WAPT agents find their WAPT Server using its IP address).

Hint: If you want the **`waptupgrade`** package to work, you will have to follow the procedure *to generate the necessary keys*.

Stop at the section on **re-signing the packages on the repository**.

Once the installation has finished, you may re-import all the WAPT packages that were saved using the *Import from file* button in the *Private repository* tab.

This operation will re-sign all the WAPT packages.

Installing the new WAPT agent 1.6 on your computers

- *recreate the WAPT agent*, which will automatically upload the **`waptupgrade`** package;
- then, *update the GPO for deploying the WAPT agent*;
- the **`waptupgrade`** package will then automatically install on the computers that already had the package and finally, the WAPT agent will automatically upgrade when the computer shuts down;

Attention:

- you will have to re-affect the packages to the hosts (no stress, it will not remove the software that are already installed);
  - you will just lose temporarily the list of your computers in the WAPT console;
  - this procedure does not allow to upgrade with the **`waptupgrade`** package;

Complete upgrade

This procedure upgrades completely your Organization's WAPT setup from 1.3 to 1.6 because it will migrate the database and WAPT base, group and host packages.

Upgrading the WAPT Server in version 1.3 to a more recent x64 Linux distribution

Under Linux, you may save your MongoDB inventory with the command `mongodump --db wapt --archive=/root/wapt.dump`.

```
mongodump
```

Hint: if the **`mongodump`** command is not available, you may install the utility with `apt install mongo-tools`.

Launching **`mongodump`** creates a `dump` file, save it.

You must also save your WAPT base and host packages. They are stored in `/var/www/wapt/` and `/var/www/wapt-host/`.

When you will install your new x64 based WAPT Server, you will have to reinstall WAPT 1.3.13 before upgrading to 1.6:

https://www.wapt.fr/en/doc-1.3/Installation/waptserver/linux/index.html

You will now restore the WAPT base and host packages previously saved in `/var/www/wapt/` and `/var/www/wapt-host/` of your new x64 based WAPT Server.

To restore the MongDB dump file, you may execute the command `mongorestore /root/dump`.

You may now follow the classic procedure to upgrade from 1.3.13 to 1.6:

- for *Debian Linux*;
- for *CentOS/ RedHat*;

Upgrading WAPT from 1.3 to 1.6 on Debian

Preamble

Note: We make the assumption that your WAPT Server is installed on a basic minimal install of Debian9 (x64). If this is not the case, you may follow the documentation to *upgrade your base server*.

This procedure aims to explain the migration of WAPT 1.3 to 1.6, only.

Element	WAPT 1.3	WAPT 1.6
Database	**`MongoDB`**	**`PostgreSQL`**
Web server	**`Apache2`**	**`Nginx`**
WAPT agent	agent listening on agent port 8088	agent initiating and maintaining a websocket with the server.
Signature	sha1 hashes	a *Code Signing* certificate is required, `control` file attributes are signed with sha256 hashes.

These changes require to follow scrupulously several operations for a smooth upgrade.

Install systemd and ca-certificates

- install systemd

```
apt install systemd
```

- install ca-certificates:

```
apt install ca-certificates
```

- restart the WAPT service:

```
reboot
```

Uninstalling WAPT 1.3 from the Debian server

```
apt remove tis-waptrepo tis-waptsetup tis-waptserver
systemctl stop apache2
systemctl disable apache2
```

Setting up the GNU/ Linux Debian server

```
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
```

Installing WAPT 1.6 on the Debian server

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

Note: The installation may ask you for the Kerberos realm. You may ignore it by pressing *Enter* to go on to the next step.

Launching the post-configuration script

Note:

- we advise that you launch the post-configuration steps after each server upgrade so that the server uses the latest configuration format;
- it is not required to reset a password for the WAPT console during the post-configuration step;

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

The post-configuration step will offer you to change the password or to move to the next step, you may choose to change the password if desired.

The post-configuration step will then detect that the current version is 1.3 and it will try to launch the process of migrating the MongoDB database to PostgreSQL.

The post-configuration step will next offer you to configure the **`Nginx`** web server. Validate this step.

Starting up WAPT on the Debian server

```
systemctl enable waptserver
systemctl start waptserver
```

Cleaning up the Debian server

At the end of the migration process, it is necessary to clean the WAPT Server.

WAPT will use from now on **Nginx** as its web server and **PostgreSQL** as its database server.

```
apt remove apache2 mongodb
apt autoremove
apt clean
```

Installing the new WAPT console

- download **waptsetup**: https://srvwapt.mydomain.lan/wapt/waptsetup-tis.exe;
- start the installation; the configuration of the WAPT repository and server URLs has not changed;
- open the **waptconsole** by selecting `C:\Program Files (x86)\wapt\waptconsole.exe` (default location) or `C:\wapt\waptconsole.exe` (older WAPT versions);
- check that the WAPT Server works correctly by clicking on the *wrench icons* and the button *Verify*!

You may now go to the next step to *generate the necessary keys*.

Upgrading WAPT from 1.3 to 1.6 on CentOS/ RedHat

Note: We make the assumption that your WAPT Server is installed on a basic minimal install of CentOS7 (x64). If this is not the case, you may follow the documentation to *upgrade your base server*.

This procedure aims to explain the migration of WAPT 1.3 to 1.6, only.

The main differences between these two versions of WAPT are:

Element	WAPT 1.3	WAPT 1.6
Database	**MongoDB**	**PostgreSQL**
Web server	**Apache2**	**Nginx**
WAPT agent	agent listening on agent port 8088	agent initiating and maintaining a websocket with the server.
Signature	sha1 hashes	a *Code Signing* certificate is required, `control` file attributes are signed with sha256 hashes.

These changes require to follow scrupulously several operations for a smooth upgrade.

Uninstalling WAPT 1.3 from the CentOS/ RedHat server

```
yum remove tis-waptrepo tis-waptsetup tis-waptserver
systemctl stop httpd
systemctl disable httpd
```

Configuring the CentOS/ RedHat server

```
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/
```

Updating the CentOS / RedHat server

```
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
```

Installing WAPT 1.6 on the CentOS / RedHat server

```
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: The installation may ask you for the Kerberos realm. You may ignore it by pressing *Enter* to go on to the next step.

Launching the post-configuration script

Note:

- we advise that you launch the post-configuration steps after each server upgrade so that the server uses the latest configuration format;
- it is not required to reset a password for the WAPT console during the post-configuration step;

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

The post-configuration step will offer you to change the password or to move to the next step, you may choose to change the password if desired.

The post-configuration step will then detect that the current version is 1.3 and it will try to launch the process of migrating the MongoDB database to PostgreSQL. Validate this step.

The post-configuration step will next offer you to configure the **Nginx** web server. Validate this step.

Starting up WAPT on the CentOS/ RedHat server

```
systemctl enable waptserver
systemctl start waptserver
```

Cleaning up the CentOS/ RedHat server

At the end of the migration process, it is necessary to clean the WAPT Server.

WAPT will use from now on **Nginx** as its web server and **PostgreSQL** as its database server.

```
yum remove httpd mongodb
```

Installing the new WAPT console

- download **waptsetup**: https://srvwapt.mydomain.lan/wapt/waptsetup-tis.exe;
- start the installation; the configuration of the WAPT repository and server URLs has not changed;
- open the **waptconsole** by selecting `C:\Program Files (x86)\wapt\waptconsole.exe` (default location) or `C:\wapt\waptconsole.exe` (older WAPT versions);
- check that the WAPT Server works correctly by clicking on the *wrench icons* and the button *Verify*!

You may now go to the next step to *generate the necessary keys*.

Upgrading WAPT from 1.3 to 1.6 on Windows

Attention: WAPT Server no longer installs on x86 versions of Windows.

- download the latest version of **`waptserversetup`** 1.6 from https://wapt.tranquil.it/wapt/releases/latest/;
- launch the installation on top of the existing 1.3 version by following *this documentation*;

Note: At the end of the post-configuration step, **`waptserver`** will detect that you are upgrading from WAPT 1.3.13 and will ask you to launch the migration of the database from MongoDB to PostgreSQL.

- click on *Yes*;
- launch the WAPT console

Note: If you come from a WAPT 1.3.13 version running with IIS, the WAPT listening ports must be changed. In that case, follow the documentation for *changing the listening port*.

You may now go to the next step to *generate the necessary keys*!!

Migrating WAPT 1.3 from a Windows OS to a Linux OS

The simplest method is to move over to a Linux based version of **`waptserver`**.

- download **`mongodb`** from https://www.mongodb.com/download-center/community;
- extract `mongodump.exe` from the archive;

Note: A dump folder should have been created in the same directory as the `mongodump.exe` file.

- backup the entire directory `C:\wapt` of the WAPT Server;
- backup the folder `C:\private`;
- install a fresh version **1.3.13** of WAPT on Linux (debian 8 x64) or CentOS7/ RedHat7 (x64);

Hint: To install a new Linux Debian 10 (Buster) on a physical or virtual machine without a graphical user interface, please visit the official documentation for Debian9.

- if the WAPT agents point to an IP address, then the new Debian based WAPT Server must have the same IP address as the old Windows based WAPT Server.
- if the WAPT agents point to a DNS CNAME, then you may point the *DNS* field *srvwapt* to the IP address of the new Linux server.
- update the download sources;

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

- install the WAPT Server;

Note: The utilities **`tis-waptserver`**, **`tis-waptsetup`** et **`tis-waptrepo`** are signed; it is therefore necessary to recover the GPG key below to avoid warning messages when installing them.

```
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
```

- launch the configuration script;

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

Note: The password requested in step 4 is used to access the WAPT console.

- configure the WAPT Server;
- start the WAPT Server;

```
systemctl start waptserver
```

- restore the WAPT packages on the Linux server;
  - upload the content of `C:\waptwaptserver\repository\wapt` in `/var/www/wapt/`;
  - upload the content of `C:\waptwaptserver\repository\wapt-host` in `/var/www/wapt-host/`;

  Hint: You may upload the files on the Linux Server using the **`WinSCP`** utility.

  - then change the owner of the files to wapt:

  ```
  chown wapt:www-data /var/www/wapt*
  ```

- restore the MongoDB database on the Linux server:
  - using **`WinSCP`**, upload the MongoDB dump folder in `/root/`;
  - restore the MongoDB dump on your Linux hosted MongoDB instance:

  ```
  mongorestore /root/dump
  ```

Your WAPT Server now works in 1.3.13 on Linux.

You may now install your **`waptagent`** on your *Administrator* management PC and restore the `C:\private` folder on your workstation.

Attention: You must not regenerate a private key, you must only point to your private key in the console. You must also refill the package prefix.

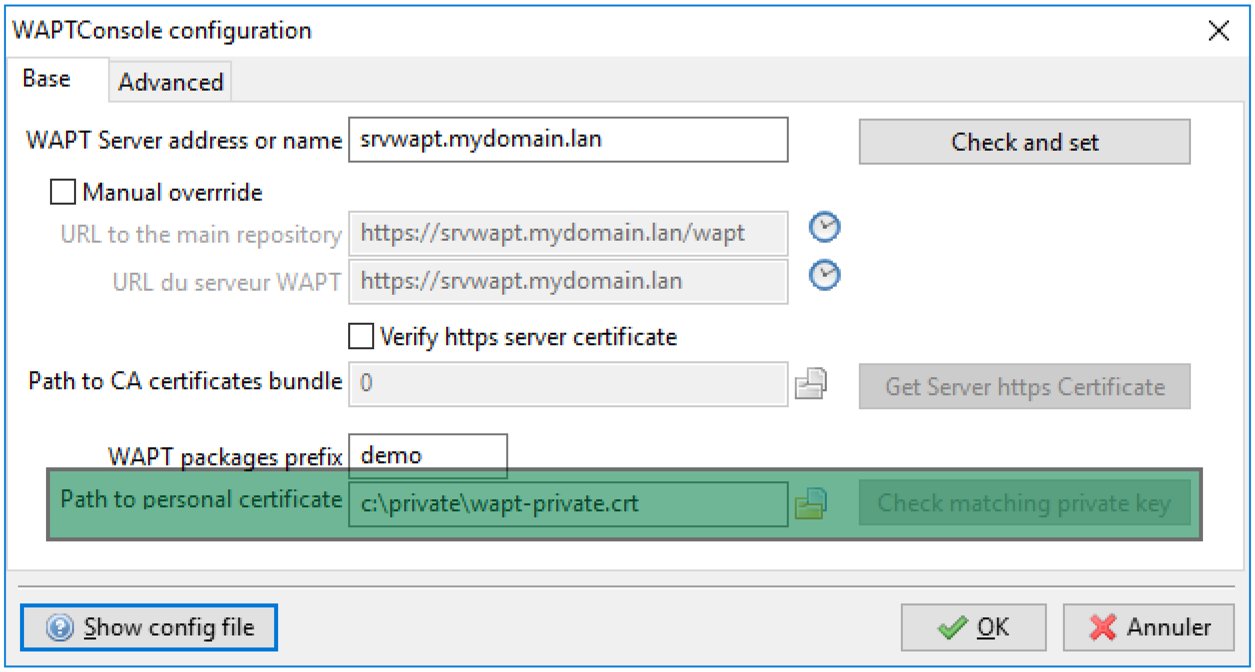
You may now follow the classic procedure to upgrade from 1.3.13 to 1.6!!

Upgrading the WAPT console and the WAPT agents

Defining a password for the Administrator's private key

The WAPT console has lost the path to the private key, it is normal because in 1.3 the WAPT console was pointing to the private key. Now in 1.5, the WAPT console points to the certificate associated with the private key.

Go to *Tools* → *Preferences* → *Path to personal certificate*, then fill in the path to the new certificate.


WAPTConsole configuration

WAPT Server address or name: srvwapt.mydomain.lan

Manual override: ☐

URL to the main repository: https://srvwapt.mydomain.lan/wapt

URL du serveur WAPT: https://srvwapt.mydomain.lan

Verify https server certificate: ☐

Path to CA certificates bundle: 0

WAPT packages prefix: demo

Path to personal certificate: c:\private\wapt-private.crt

Fig. 136: Select the private key certificate

WAPT 1.6 requires to protect the private key with a password to sign packages and actions.

In the WAPT console, go to *Tools* → *Change the password of the private key*, then select your certificate and enter a password for your private key.

Generating a *Code Signing* certificate

WAPT 1.6 differentiates **Code Signing** certificates from simple **SSL** certificates.

It is therefore necessary to regenerate a **Code Signing** certificate.

By default, WAPT 1.6 Community generates self-signed **Code Signing** certificates.

You must insure that the old certificate in `C:\private\mykey.crt` and the key `C:\private\mykey.pem` are present in `C:\private`.

Attention: For packages to be deployed on the *Organization*'s computers, all packages will have to be re-signed with a *Code Signing* certificate.

To create the *Code Signing* certificate, in the WAPT console go to *Tools* → *Generate a certificate*.

Note: Normally, you will not have to recreate a new key. Only the certificate will have to be changed. **You must enter the path to your key** (ex: `C:\private\mykey.pem`).

Attention: Do not make changes to the informations on the certificate. If any information is changed, then you will not be able to migrate from 1.3.13 to 1.6 using the **`waptupgrade`** procedure.

The WAPT console now asks you whether the new certificate should be added to the `ssl` folder of the local WAPT agent, you can accept.

Now that you have generated your new key you must now change the path to your personal certificate. The WAPT console must now point to this new certificate.

Go to *Tools* → *Preferences* → *Path to personal certificate*, then fill in the path to the new certificate.

You may now delete the old certificate.

Re-signing the packages on your WAPT repositories

All WAPT 1.3.13 must be re-signed because WAPT 1.6 packages must be signed with a *Code Signing* certificate and hashed with sha256.

Linux

You must temporarily copy your private key (`.pem`) and the *Code Signing* certificate (`.crt`) on your Linux based WAPT Server using **`WinSCP`** or an equivalent tool.

Then, connect with SSH to the Linux WAPT Server and re-sign all WAPT *base* packages with the new certificate:

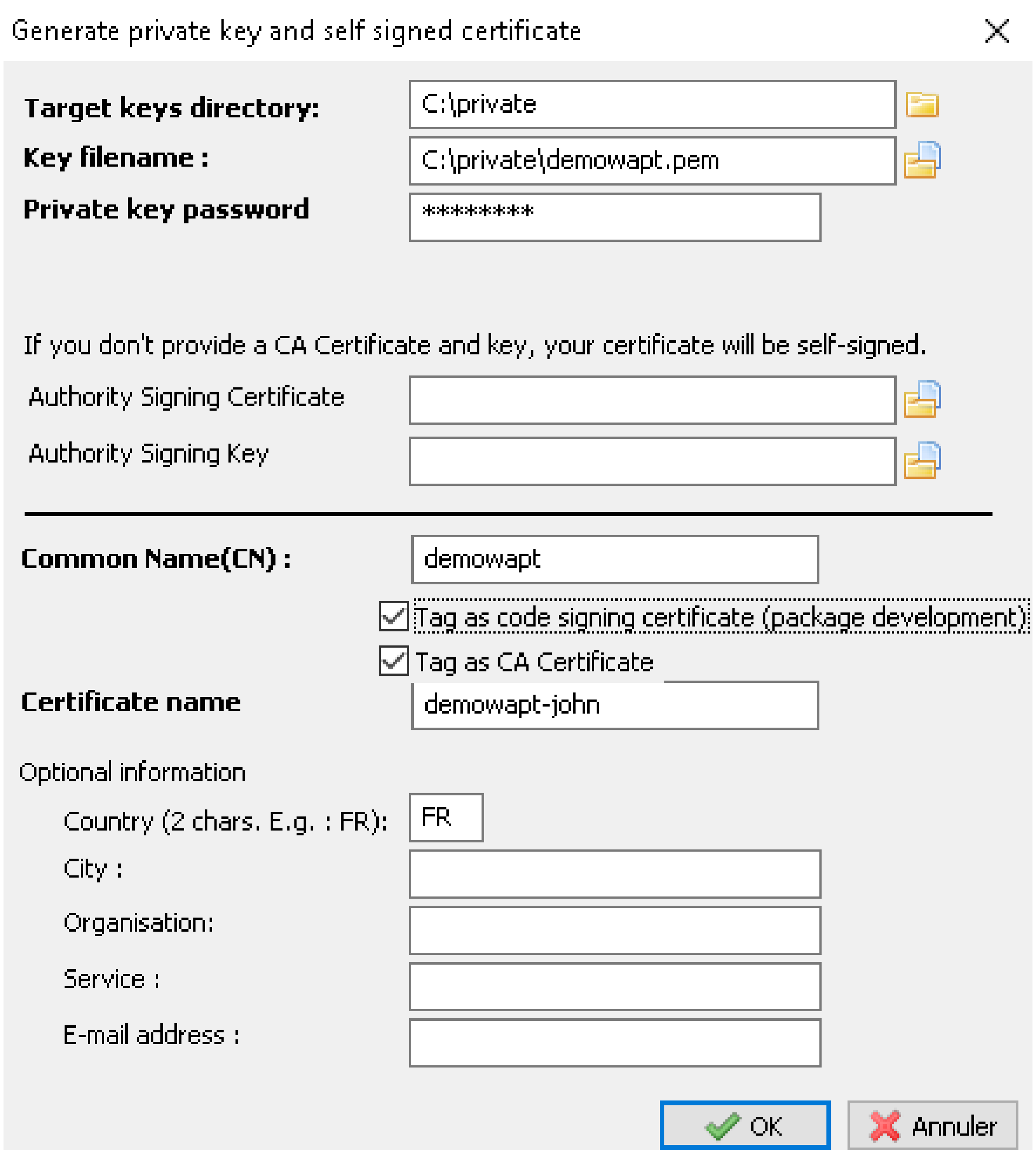
```
PYTHONPATH=/opt/wapt PYTHONHOME=/opt/wapt python /opt/wapt/wapt-signpackages.py -i -s --message-
↪digest=sha256,sha1 -c /root/wapt-private-20180312-1522.crt /var/www/wapt/*.wapt
```

Hint: The WAPT Server *SuperAdmin* password is requested to access the database so to do the matching between *FQDN* and their corresponding *UUID* .

Rename host packages with the UUID nomenclature:

```
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
```

Note: The private key now password protected, the password is requested for signing the packages.


Generate private key and self signed certificate

Target keys directory: C:\private

Key filename : C:\private\demowapt.pem

Private key password *********

If you don't provide a CA Certificate and key, your certificate will be self-signed.

Authority Signing Certificate

Authority Signing Key

Common Name(CN) : demowapt

[✓] Tag as code signing certificate (package development)

[✓] Tag as CA Certificate

Certificate name demowapt-john

Optional information

Country (2 chars. E.g. : FR): FR

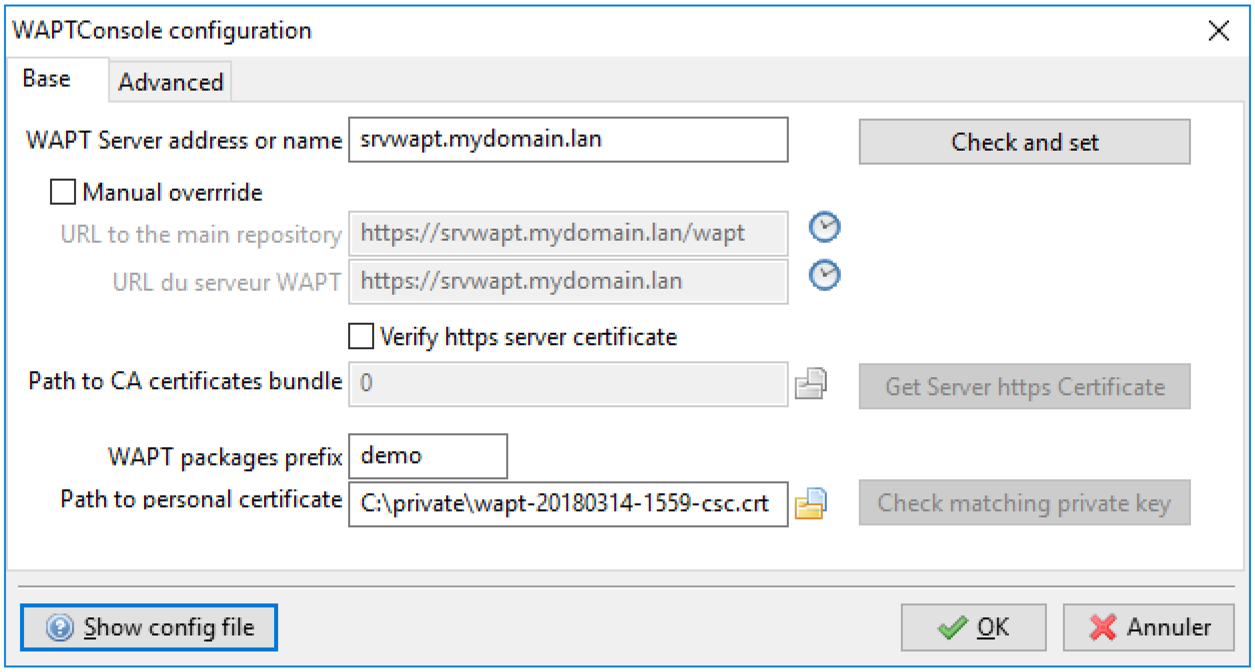
City :

Organisation:

Service :

E-mail address :

Fig. 137: Code Signing certificate


WAPTConsole configuration
Base
Advanced
Check and set
Get Server https Certificate
Check matching private key
Show config file
OK
Annuler

WAPT Server address or name: srvwapt.mydomain.lan

☐ Manual override

URL to the main repository: https://srvwapt.mydomain.lan/wapt

URL du serveur WAPT: https://srvwapt.mydomain.lan

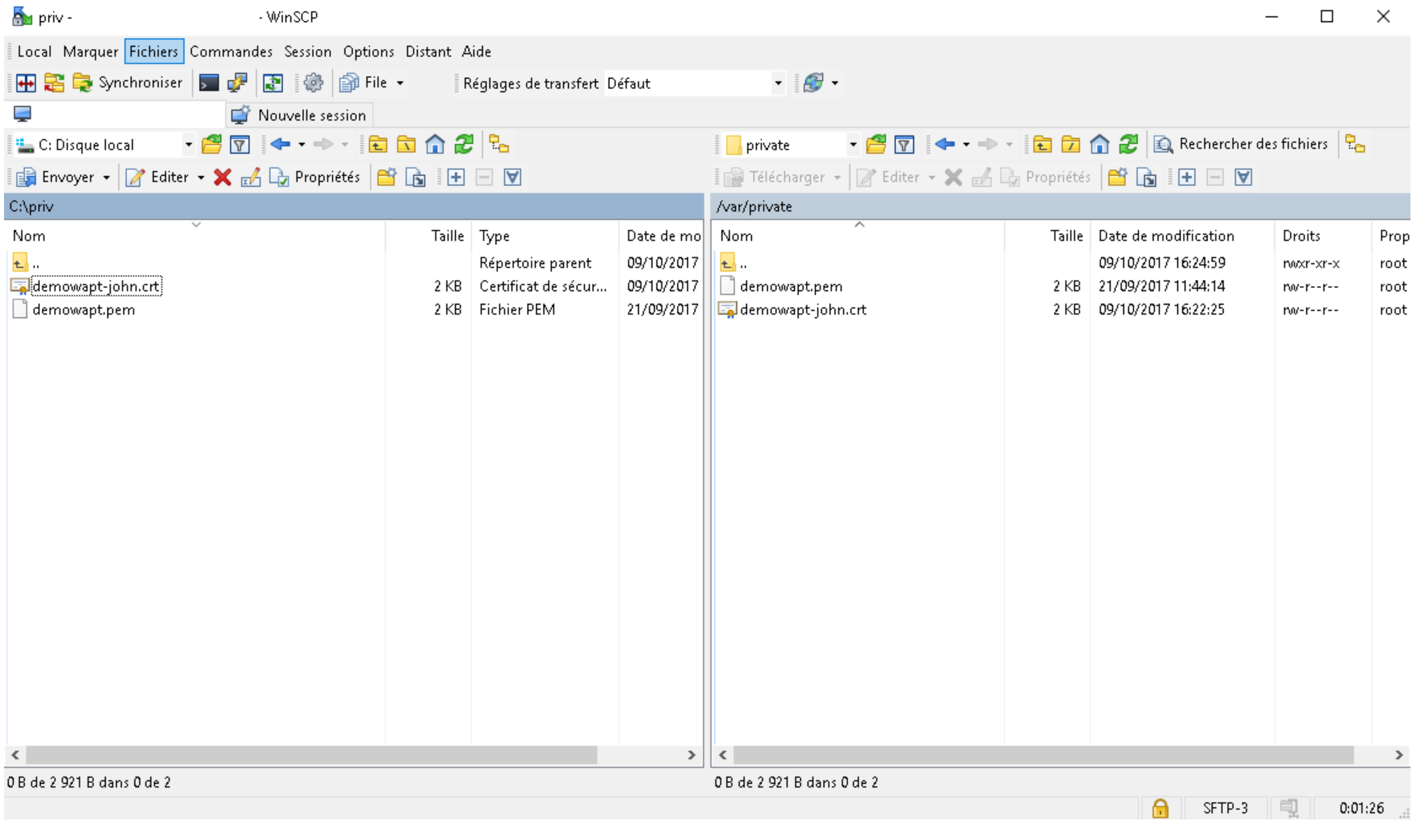
☐ Verify https server certificate

Path to CA certificates bundle: 0

WAPT packages prefix: demo

Path to personal certificate: C:\private\wapt-20180314-1559-csc.crt

Fig. 138: New Code Signing certificate



C:\priv

Nom	Taille	Type	Date de mo
..		Répertoire parent	09/10/2017
demowapt-john.crt	2 KB	Certificat de sécur...	09/10/2017
demowapt.pem	2 KB	Fichier PEM	21/09/2017

/var/private

Nom	Taille	Date de modification	Droits	Prop
..		09/10/2017 16:24:59	rwxr-xr-x	root
demowapt.pem	2 KB	21/09/2017 11:44:14	rw-r--r--	root
demowapt-john.crt	2 KB	09/10/2017 16:22:25	rw-r--r--	root

0 B de 2 921 B dans 0 de 2

0 B de 2 921 B dans 0 de 2

SFTP-3 0:01:26

Fig. 139: Temporarily upload your Code Signing certificate on the WAPT server

Attention: **DO NOT FORGET** to delete the private key (.pem) from the WAPT Server!

Windows

Then, open a session on the Windows machine hosting the WAPT Server and re-sign all WAPT base packages with the new certificate:

```
wapt-signpackages -i -s --message-digest=sha256,sha1 -c C:\private\wapt-private-20180312-1522.
↪crt C:\wapt\waptserver\repository\wapt\*.wapt
```

Hint: The WAPT Server *SuperAdmin* password is requested to access the database so to do the matching between FQDNs and their corresponding UUIDs.

Rename *host* packages with the UUID nomenclature:

```
"C:\wapt\waptserver\scripts\migrate-hosts.bat" -C C:\private\wapt-private-20180312-1522.crt -K␣
↪C:\private\wapt-private.pem
```

Note: The private key now password protected, the password is requested for signing the packages.

Attention: **DO NOT FORGET** to delete the private key (.pem) from the WAPT Server!

Generating the PostgreSQL database table for the group packages

In WAPT 1.3, group filtering was done by scanning the `Packages` index file in the `wapt-host` folder of the WAPT Server.

For performance reasons, the filtering is now done by querying a PostgreSQL database table. The table will fill in automatically as WAPT agents register with the WAPT Server 1.6.

To quickly regenerate the database table, the trick is to create a temporary group *mig-temp*.

You may then apply the *mig-temp* package to all WAPT clients from the WAPT console (`CTRL+A` in the inventory, then *Right-click* → *Add dependencies* and select the package *mig-temp*).

You may now revert the operation (`CTRL+A` in the inventory, then *Right-click* → *Remove dependencies* and select the package *mig-temp*).

The filtering of groups should be operational again.

Ugrading WAPT agents

You may now follow the procedure to *create the new WAPT agent*.

Attention: The version update implies a change in the way packages are signed.

If you update from 1.3.13, the WAPT package **waptupgrade** should install correctly if you check the box *sign waptupgrade with sha256 AND sha1* while generating the WAPT agent.

If your WAPT agents are in a version that is less than 1.3.13, the package **waptupgrade will not work**.

You may choose instead to use a **waptdeploy** GPO to *deploy the new WAPT agent on your installed base of PCs*.

Installing waptupgrade on the computers

With version 1.6, it is no longer possible to directly contact WAPT agents in an inferior version from the WAPT console.

To allow you to install the *waptupgrade* package on 1.3.13 computers, we have designed a small script.

Linux

You may thus launch the script:

```
/opt/wapt/waptserver/trigger_action.sh prefix-waptupgrade
```

Windows

You may thus launch the script:

```
"C:\wapt\waptserver\trigger_action.bat" prefix-waptupgrade
```

6.9.4 Minor Upgrades

This procedures concerns the classic method for upgrading the WAPT Server for minor versions, for exemple from 1.7.4 to 1.8.

The migration process includes:

- the upgrading of the WAPT Server;
- the upgrading of the WAPT console;
- the upgrading of the WAPT agent;
- the updating of the deployment GPO;

Note: For upgrading from 1.3 to 1.6, please refer to *upgrading from 1.3 to 1.6*.

Performing minor updates on a Debian based WAPT Server

Attention: Ports 80 and 443 are used by the WAPT Server and must be available.

- first of all, update the Debian underlying distribution:

```
apt update && apt upgrade -y && apt dist-upgrade
```

- add the package repository for Debian packages, import the GPG key from the repository and install the WAPT Server packages:

WAPT Enterprise

Hint: **To access WAPT Enterprise ressources**, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **deb** parameter to access WAPT Enterprise repository.

```
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

- launch the post-configuration step

  Note:

  - we advise that you launch the post-configuration steps after each server upgrade so that the server uses the latest configuration format;
  - it is not required to reset a password for the WAPT console during the post-configuration step;
  - if you have personalized the configuration of **`Nginx`**, do not answer *Yes* when the post-configuration ask you to configure **`Nginx`**;

Attention:

- with WAPT 1.8 post-configuration, WAPT WUA packages will be moved from their current storage location to waptwua root folder (`/var/www/waptwua`).
- if repository replication has been set, all KB/CAB packages will be re-synchronized on remote repositories

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

The password requested in step 4 is used to access the WAPT console.

- start the WAPT Server:

```
systemctl restart waptserver
```

- upgrade the WAPT console by following the same set of steps as *installing the WAPT console*;
- then *create the WAPT agent*:

  You will have to keep the same prefix for your packages and change nothing in relation to the private key/ public certificate pair!

  This will generate a **waptupgrade** package in the private repository.

Note: There are two methods for deploying the updates:

- using a GPO and **`waptdeploy`**;
- using a **`waptupgrade`** package and deploy it using WAPT;

- update the WAPT agents

  The steps to follow to update WAPT agents are the same as the ones to first install the WAPT agents.

  Download and install the latest version of the WAPT agent by visiting http://wapt.mydomain.lan/wapt/waptagent.exe.

  As mentioned above, this procedure may be made automatic with a GPO or a **waptupgrade** package.

Performing minor updates on a CentOS/ RedHat based WAPT Server

Attention: Ports 80 and 443 are used by the WAPT Server and must be available.

- first of all, update the CentOS/ RedHat underlying distribution:

```
yum update
```

WAPT Enterprise

Modify the repository address then launch the upgrade.

Hint: To access WAPT Enterprise ressources, you must use the username and password provided by our sales department.

Replace **user** and **password** in the **baseurl** parameter to access WAPT Enterprise repository.

```
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

- modify the repository address then launch the upgrade:

```
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

- launch the post-configuration step:

  Note:

  - we advise that you launch the post-configuration steps after each server upgrade so that the server uses the latest configuration format;
  - it is not required to reset a password for the WAPT console during the post-configuration step;
  - if you have personalized the configuration of **`Nginx`**, do not answer *Yes* when the post-configuration asks you to configure **`Nginx`**;

Attention:

- with WAPT 1.8 post-configuration, WAPT WUA packages will be moved from their current storage location to the waptwua root folder (`/var/www/waptwua`).
- if repository replication has been set, all KB/CAB packages will be re-synchronized on remote repositories.

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

- start the WAPT Server:

```
systemctl start waptserver
```

- upgrade the WAPT console by following the same set of steps as *installing the WAPT console*;
- then *create the WAPT agent*:

  You will have to keep the same prefix for your packages and change nothing in relation to the private key/ public certificate pair!

  This will generate a **waptupgrade** package in the private repository.

  Note: There are two methods for deploying the updates:

  - using a GPO and **`waptdeploy`**;
  - using a **`waptupgrade`** package and deploy it using WAPT;

- update the WAPT agents:

  The steps to follow to update WAPT agents are the same as the ones to first install the WAPT agents.

  Download and install the latest version of the WAPT agent by visiting http://wapt.mydomain.lan/wapt/waptagent.exe.

  As mentioned above, this procedure may be made automatic with a GPO or a **waptupgrade** package.

Performing minor updates on a Windows based WAPT Server

Attention: In the case you choose to use a GPO to upgrade the WAPT agent, the WAPT Server must be excluded from the OU where the GPO will apply.

Note: The WAPT Server may be installed on 64bit Windows 7 and Windows 10 clients, and 64 bit Windows Server 2008/R2, 2012/R2 and 2016.2, 2012/R2, 2016 and 2019.

- on the Windows machine hosting the WAPT Server, download the latest version of the installer from Tranquil IT's website WAPTServerSetup.exe and launch it as *Local Administrator*;
- install the update;

Note: The procedure to follow is the same as the one for *installing a WAPT Server on Windows* .

Attention: The prefix must NOT be changed and you must NOT generate a new key!

- on the workstation that you use to build your packages, manually download WAPTSetup from
- Community: waptserversetup.exe;
- Enterprise: waptserversetup.exe;
- then *create the WAPT agent*:

  You will have to keep the same prefix for your packages and change nothing in relation to the private key/ public certificate pair!

  This will generate a **waptupgrade** package in the private repository.

  Note: There are two methods for deploying the updates:

  - using a GPO and **`waptdeploy`**;
  - using a **`waptupgrade`** package and deploy it using WAPT;

- update the WAPT agents

  The steps to follow to update WAPT agents are the same as the ones to first install the WAPT agents.

  Download and install the latest version of the WAPT agent by visiting http://wapt.mydomain.lan/wapt/waptagent.exe.

  As mentioned above, this procedure may be made automatic with a GPO or a **waptupgrade** package.

Upgrading the Operating System

Upgrading from Debian 9 Stretch to Debian 10 Buster

In order to upgrade your WAPT server from Stretch to Buster you have to follow the standard procedure for Debian. You first modify the apt source files `/etc/apt/sources.list` and `/etc/apt/sources.list.d/wapt.list`, then start the upgrade.

By default the PostgreSQL is not upgraded to PostgreSQL 11. One needs to manually request the upgrade. After upgrading, it is possible to remove the old PostgreSQL 9.6 database.

```
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
pg_upgradecluster -v 11 9.6 main
apt remove postgresql*-9.6
apt autoremove
/opt/wapt/waptserver/scripts/postconf.sh
```

Upgrading from CentOS 7 to CentOS 8

WAPT does not yet support CentOS 8. This section will be updated accordingly when support will be added.

6.10 Backing up the WAPT Server

6.10.1 Backing up the WAPT Server on Linux

Attention: This procedure is valid for WAPT 1.5 and above.

- stop WAPT related services on the server;

```
systemctl stop nginx
systemctl stop waptserver
systemctl stop wapttasks
```

- backup these directories using a backup tool (ex: **rsync**, **WInSCP**, etc..);

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

- backup the PostgreSQL database using the **pg_dumpall** utility (adapt filename with your requirements);

```
sudo -u postgres pg_dumpall > /tmp/backup_wapt_$(date +%Y%m%d).sql
```

- restart WAPT related services on the server;

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

6.10.2 Restoring the WAPT Server on Linux

In case of a complete crash, restart a standard WAPT Server installation on a Linux server.

- stop WAPT related services on the server;

```
systemctl stop nginx
systemctl stop waptserver
systemctl stop wapttasks
```

- restore the following directories:

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
```

(continues on next page)

(continued from previous page)

```
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

- restore the database (adapt the name of your file). The first command **deletes** the WAPT database (if it exists). Make sure that your dump file is correct before deleting!

```
sudo -u postgres psql -c "drop database wapt"
sudo -u postgres psql < /tmp/backup_wapt_20180301.sql
```

- apply ownership rights to the restored folders:

```
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/
```

- scan package repositories;

```
wapt-scanpackages /var/www/wapt/
```

- restart WAPT related services on the server;

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

6.10.3 Backing up the WAPT Server on 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;

Backup PostgreSQL Database with pg_dump.exe:

```
"C:\wapt\waptserver\pgsql-9.6\bin\pg_dumpall.exe" -U postgres -f C:\backup_wapt.sql
```

- restart WAPT related services on the server;

6.10.4 Restoring the WAPT Server on Windows

- restore the following directories `C:\wapt\waptserver\repository` and `C:\wapt\waptserver\conf` and `C:\wapt\waptserver\nginx\ssl`
- Apply the total right to the folder `C:\wapt\waptserver\repository` for the "Network Service" group

Restore PostgreSQL Database with pg_restore.exe:

```
"C:\wapt\waptserver\pgsql-9.6\bin\psql.exe" -f c:\backup_wapt.sql -U postgres
```

6.11 Replicating repositories and working with multiple repositories

Large organizations with remote sites and subsidiaries sometimes require services to be replicated locally to avoid bandwidth congestion (*Edge Computing*).

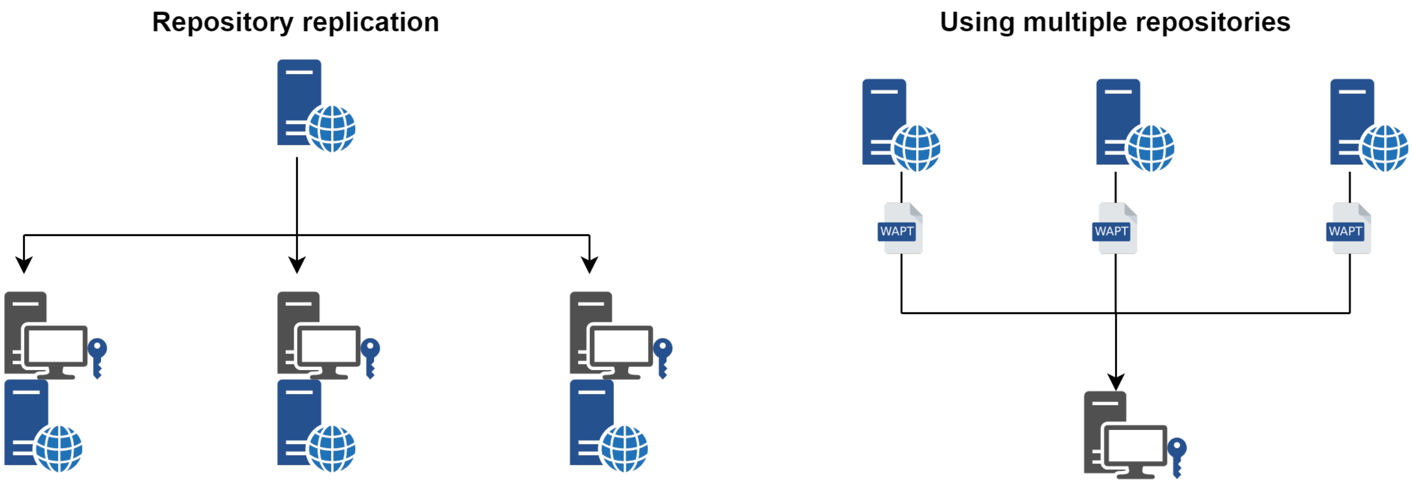


Fig. 140: Repository replication and multiple repositories

New in version WAPT: Enterprise 1.8

WAPT Enterprise offers the possibility to upgrade remote agents to serve as remote repositories that can be managed directly from the WAPT Console. All WAPT agents can then be centrally configured to automatically select the best repository based on a set of rules.

You'll find in this part of the documentation how to implement such architectures.

6.11.1 Replicating a repository to preserve the bandwidth on remote sites

When WAPT is used on bandwidth limited remote sites, it makes sense to have a local device that will replicate the main WAPT repository to reduce the network bandwidth consumed when deploying updates on your remote devices.

With remote repositories, WAPT remains a solution with a low operating cost because you don't have to implement high bandwidth fiber links to take advantage of WAPT.

It works as follows:

- a small form factor and no maintenance appliance with the role of secondary repository is deployed on the local network of each remote site; a workstation can also be used, although it may not be up and running if you want to connect to it;
- the remote repository replicates the packages from the main repository and other repositories;
- the WAPT clients connect in priority with the repository that is the closest to them, the local repository;

Hint: The former method used to sync repositories was Syncthing and that method is available both for the Community and the Enterprise versions of WAPT.

You can find the old documentation here: *(Deprecated) Replicating repositories with Syncthing*.

The method explained below is for the Enterprise version only.

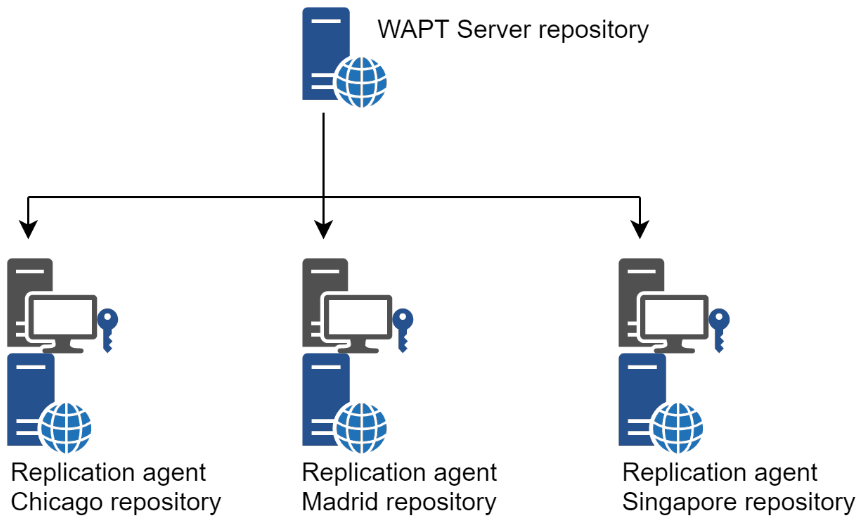


Fig. 141: Replicating WAPT repositories

WAPT Agent replication role

New in version WAPT: 1.8 Enterprise

Starting with WAPT 1.8, repository replication can be enabled using a WAPT agent installed on an existing machine, a dedicated appliance or Virtual Machine.

The replication role is deployed through a WAPT package that enables the **`Nginx web server`** and configures scheduling, packages types, packages sync, and much more.

This feature allows WAPT agents to find dynamically their closest available WAPT repository from a list of rules stored on the WAPT server.

Replication behavior

Repository replication in WAPT is now handled by WAPT agents natively (**Enterprise versions only**).

It is based on a `sync.json` file which indexes every files present in these folders:

- wapt;
- waptwua;
- wapt-host;

Enabling replication has the following effects:

- once `enable_remote_repo` is enabled on a WAPT agent, it will sync packages locally inside the `local_repo_path` folder;
- it adds the WAPT agent in the *Repositories* tab as a Remote repository, enabling new actions such as *Force Sync* or *Check files*;
- by default, only the `wapt` folder is synchronized, you can select which folder to sync by adding up elements in `remote_repo_dirs` parameters;
- synchronization period can be configured with `local_repo_time_for_sync_start` and `local_repo_time_for_sync_stop` parameters;

- bandwidth allocated to sync can be configured with `local_repo_limit_bandwidth`;

Every parameters of WAPT repository sync must be set in the `[repo-sync]` section of the WAPT agent's `wapt-get.ini` configuration file.

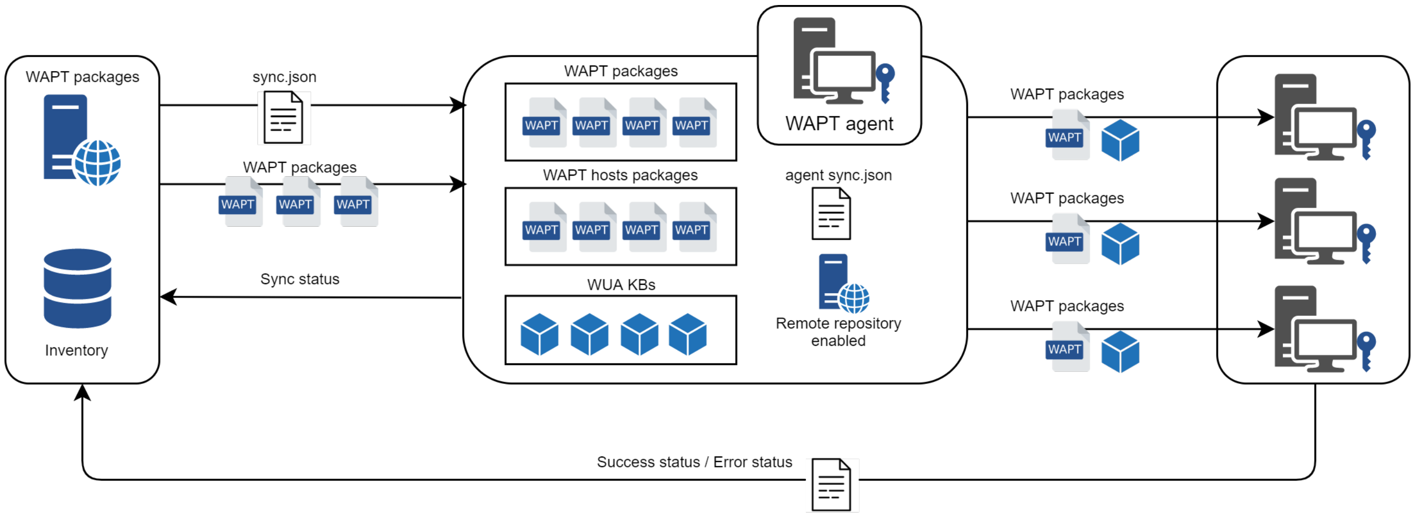


Fig. 142: Replication role behavior

Enabling replication on WAPT Agent

To enable replication on an existing agent (Linux/Windows) you need to deploy a WAPT package. It's role is to:

- install and enable the **`Nginx web server`**;
- configure nginx virtualhost;
- enable remote repository configuration in `wapt-get.ini`;

A ready-to-use WAPT package is available in our public store to enable repository replication on Windows or Linux WAPT agents: https://store.wapt.fr/store/tis-remote-repo-conf.

WAPT Agent replication configuration

WAPT Agent replication configuration is set in the `[repo-sync]` section in the `wapt-get.ini` configuration file of the WAPT agent:

Options	Mandatory	Example value	Definition
`enable_remote_repo`	Yes	`True`	Enables remote repository sync connections.
`local_repo_path`	Yes	`/var/www/`	Set local packages root repository path
`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`	Set sync stop time (HH:MM / 24h format)
`local_repo_sync_task_period`	No	`25`	Set sync period (minutes)
`local_repo_limit_bandwidth`	No	`2`	Set sync allowed bandwidth (Mbits/s)
`remote_repo_dirs`	No	`wapt,waptwua, wapt-host`	Set synced folders (default: wapt,waptwua)

Below is an example of `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
```

WAPT Server replication configuration

The WAPT Server needs to be aware of repositories to sync in the `[options]` section of its `waptserver.ini` located in `/opt/wapt/conf/`

Options	Example value	Definition
`remote_repo_support`	True	Enables remote repository sync server side (sync.json)

Then we must restart both *waptserver* and *wapttask*:

```
systemctl restart waptserver wapttask
```

Repository rules

Repository rules behavior

By enabling repository rules support, the WAPT agents will automatically retrieve their `rules.json` file from the WAPT server.

The `rules.json` file is a signed *.JSON* file that contains a list of sorted rules to apply to the WAPT agent, so to redirect its downloads from the most appropriate repository.

If no rules can be matched, WAPT agent fallbacks onto the `repo_url` settings of the WAPT server `wapt-get.ini` configuration file.

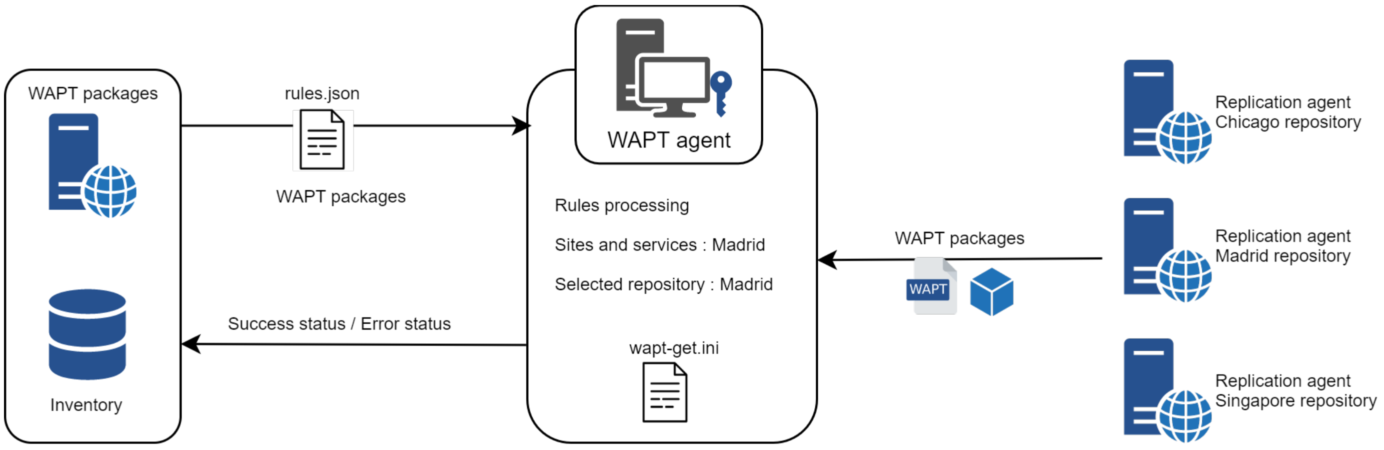


Fig. 143: Repository rules behavior

Enabling repository rules

Repository rules are configured in WAPT Console.

Rules can be based on several parameters:

Options	Example value	Definition
Domain name	`ad.domain.lan`	Rule based on Active Directory domain name
Domain sites and services	`Paris-HQ`	Rule based on Active Directory Sites and Services
Agent IP	`192.168.85.0/24`	Rule based on Agent IP sub-network
Public IP	`256.89.299.22/32`	Rule based on Public IP (NATed hosts)
Hostname	`desktop-04feb1`	Rule based on hostname

Adding a rule in the WAPT Console

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.

Using repository rules on WAPT agents

Warning: If you have configured GeoIP redirects on Nginx, you should disable it as it might conflicts with repository rules.

To enable WAPT Agent repository rules, you must enable this setting in the `[global]` section of `wapt-get.ini` configuration of the WAPT agent:

Options	Mandatory	Example value	Definition
`use_repo_rules`	No	`True`	Enables repository rules usage

Below is an example of `wapt-get.ini`:

```
[global]
...
use_repo_rules = True
```

Speaking of repositories, it is sometimes useful to configure two or more repositories for different uses (prod, dev, licensed, selfservice).

6.11.2 Working with multiple public or private repositories

Multi-repository is now supported by WAPT. This functionality is useful in several use cases:

- to use a secondary private repository, hosting business application packages, independently of your main repository;
- to have remote repositories closer to users in a multi-site architecture scenario;
- to allow the usage of an open repository and a secondary repository with restricted access (licensed software..);

Attention: When using repositories with different signers, the additional signer's public certificates must be added to `C:\Program Files (x86)\wapt\ssl`. You then must deploy WAPT agent with both keys.

Please refer to the documentation to *create the WAPT agent*.

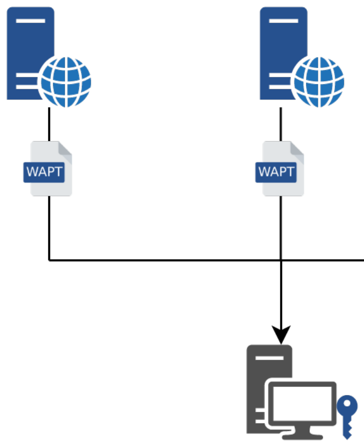


Fig. 144: Multi-repository WAPT architecture

Configuring the WAPT agents

- *repositories* parameter:

  The parameter *repository* in the `[global]` section of the `wapt-get.ini` file allows to set several options for package repositories, for example *private* and *tranquilit* sections here, where their settings are set in additional sections of that file.

```
repositories=private,tranquilit
```

- settings of secondary repositories

```
[private]
repo_url=https://srvwapt.mydomain.lan/wapt

[tranquilit]
repo_url=https://wapt.tranquil.it/wapt
```

  With that configuration, WAPT clients will now see packages from the main repository and from the secondary repository.

  The WAPT agents will look for updates on both repositories.

```
wapt-get search
```

  Packages from the secondary repository will also be visible using the web interface http://127.0.0.1:8088 on WAPT equipped devices.

Configuring the WAPT console with several private repositories

After having configured the WAPT agent for using multiple repositories, we can make the two repositories show up in the WAPT console.

To do so, modify `%appdata%localwaptconsolewaptconsole.ini` file:

```
[private]
repo_url=https://srvwapt.mydomain.lan/wapt

```

(continues on next page)

(continued from previous page)

```
[tranquilit]
repo_url=https://wapt.tranquil.it/wapt
```

6.11.3 Deprecated configurations and features

(Deprecated) Replicating repositories with Syncthing

Hint: WAPT repositories synchronization is now native in WAPT Enterprise.

You can find the new documentation here: *Replicating a repository to preserve the bandwidth on remote sites*.

Warning: This part of the documentation is no longer maintained. You may use it to collect ideas for doing your replication while using the Community version.

Introducing Syncthing

Syncthing is a multi-OS open source peer to peer synchronization utility.

It allows to synchronize folders on several machines while guaranteeing the security, the authenticity and the integrity of the files.

The official Syncthing documentation is available online.

Implementing the replication

Hint: The following documentation is applicable to Linux Debian and CentOS/ RedHat based WAPT Servers and remote repositories.

Setting up the remote WAPT repository

Debian Linux

```
echo  "deb  https://wapt.tranquil.it/debian/  ./  "  > /etc/apt/sources.list.d/wapt.list
apt update
apt upgrade -y
apt install tis-waptrepo
```

CentOS/ RedHat Linux

The remote WAPT repository is set up, we now must implement the Syncthing replication.

Configuring the Syncthing service

```
a2enmod ssl
a2ensite default-ssl.conf
```

- modify the **Apache** configuration files to define the correct roots of the *VirtualHosts*:

```
/etc/apache2/sites-available/default-ssl.conf
/etc/apache2/sites-available/000-default.conf
```

- change the value of *DocumentRoot* in each configuration file:

```
- DocumentRoot /var/www/html
+ DocumentRoot /var/www
```

- finally, restart the **Apache** web service to apply the new configuration:

```
/etc/init.d/apache2 restart
```

Note: It is advisable to specify valid SSL certificates in the Apache configuration of remote repositories.

- empty the content of the folders `/var/www/wapt` and `/var/www/wapt-host`; Syncthing will fill again these folders with data from the main repository.

```
rm -rf /var/www/wapt/*
rm -rf /var/www/wapt-host/*
```

Installing Syncthing on main and remote WAPT repositories

Note: This procedure is to be applied on the main repository and on the remote repositories.

```
## 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
```

Configuring Syncthing

Operations to follow:

- add the Syncthing service to **systemd**;
- change the listening port to **0.0.0.0**;
- create an administrator account and enter a strong password;
- activate the HTTPS protocole for the web access;
- create the definition file for the **waptsync** service by editing `/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
```

- create the tree structure required for the **waptsync** service to start:

```
mkdir /opt/wapt/.config/
mkdir /opt/wapt/.config/syncthing/
```

- change the owner of the files:

```
chown -R wapt:www-data /opt/wapt/.config/
```

- activate the **waptsync** service and start it. The configuration files will appear in the `/opt/wapt/.config/syncthing/` folder:

```
systemctl enable waptsync
systemctl start waptsync
systemctl stop waptsync
```

- change the listening port in the `/opt/wapt/.config/syncthing/config.xml` file:

```
<gui enabled="true" tls="true" debugging="false">
    <address>0.0.0.0:8384</address>
    <apikey>4jvEiL24UbFddsdsAQxqsfixNaLt</apikey>
    <theme>default</theme>
</gui>
```

- start the **waptsync** service:

```
systemctl start waptsync
```

Configuring Syncthing's web service

Syncthing's web interface is now accessible by visiting http://srvwapt.mydomain.lan:8384.

Operations to follow:

- change the host name of the remote repository;
- add a *GUI authorized user*;
- add the password for the *GUI authorized user*;
- check the box *Use HTTPS for the GUI*;
- click on *Save*;
- connect with SSH on the WAPT Server and restart the Syncthing service:

```
systemctl restart waptsync
```

Syncthing's web interface is now only accessible with HTTPS on https://srvwapt.mydomain.lan:8384.

- in the list of shared folders, remove the default folder: *Modify* → *Remove*;
- configure the replication:

  Note: Those actions must be run on WAPT Server.

  In the list of shared folders (*Shares*):

  - add a shared folder with *Add a Share*;
  - fill in the path to the directory to be shared, ex: `/var/www/wapt/`;
  - in the scroll-down menu *Directory Type* → *Only Send*;

- in the scroll-down menu *File Recovery Order* → *Older First*;
- redo the same operation for wapt-host: `/var/www/wapt-host/`;
- add the remote repository to WAPT Server's Syncthing:

  Once Syncthing has been installed on the main and remote repositories, recover the remote repository's ID (*Actions* → *Show My ID*).

  This unique identifier appears like

  ```
  DSINDDC-23ORDNM-PAK6FCL-ZJAKNCH-61GWXAT-77PC3JM-RZ4PPYP-K1QERAV
  ```

  On the main repository, in the list of remote devices (*Other Devices*):

  * add the remote repository with *Add a Device*;
  * fill in the ID of the remote repository;
  * tick the checkbox for the shared folder *wapt* and *wapt-host*;

  On the remote repository, follow these steps:

  * the remote device receives a Syncthing notification that it has been approved and added to the main device's replication schedule;
  * the client receives a popup form asking to accept the synchronized shares `wapt` and `wapt-host`;

The replication is now operational.

- secure the replication:

By default, the following parameters are active in Syncthing:

Table 25: Syncthing parameters

Options	Description
Activate the *NAT*	Use a UPnP port mapping for incoming synchronization connections.
Local discovery	Syncthing will then broadcast to announce itself to other Syncthings.
Global discovery	Syncthing registers on an external cloud service and can use this cloud based service to search other Syncthing devices.
Possible relay	The use of relays allows to use external servers to relay the communications. The relay is activated by default but it will be used only if two devices can not communicate directly between themselves.

This operating mode simplifies the global setup but it is not the most advisable method in relation to security.

- uncheck all boxes in network configuration;
- define the listening port (by default port 22000);
- replace *default* by *tcp://0.0.0.0:22000*;

Then go on the web interface of the remote repositories, click on *Change* and fill in the IP address of the remote repository:

- replace *dynamic* by *tcp://<remote_repo_ip_address>:22000*;

This configuration is useful for limiting inbound connections to the Syncthing service.

Configuring the WAPT agents

WAPT clients on remote sites must now be configured to look for updates from their closest available repositories.

Two solutions exist:

- use automatic detection via *DNS SRV* fields;
- change manually or via a WAPT package the parameter *repo_url* in the WAPT agent's `wapt-get.ini` file;

Configuration example of the WAPT agent - filled in local repository:

```
[global]
waptupdate_task_period=120
waptserver=https://srvwapt.mydomain.lan
repo_url=https://localrepo.mydomain.lan/wapt/
use_hostpackages=1
```

Example of a WAPT package designed to remotely change the *repo_url* in `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 Enhancing the security of the WAPT server

By default, all WAPT packages are signed with your private key, which already provides a great level of security. However you can further improve the security of WAPT.

To fully secure your WAPT setup; you will want to do the following:

- enable authenticated registration to filter who is authorized to register the device with the WAPT server;
- enable https certificate verification on the agents and the console to ensure that the WAPT agents and the WAPT console are connecting to the correct WAPT server;
- configure authentication against Active Directory to allow access to the WAPT console only to authorized WAPT admins;
- enable Client-Side Certificate Authentication to only allow authenticated devices to access the WAPT server (Note: it is especially important if you want to expose your WAPT server to the outside in a DMZ (De-Militarized Zone));

6.12.1 Configuring the firewall on the WAPT Server

WAPT Server firewall configuration is essential and should be the first step towards achieving better security in WAPT.

As WAPT aims to be secure by design, only a minimal set of open ports is needed on the WAPT Server compared to other solutions.

You will find in the following documentation firewall tips to improve WAPT security.

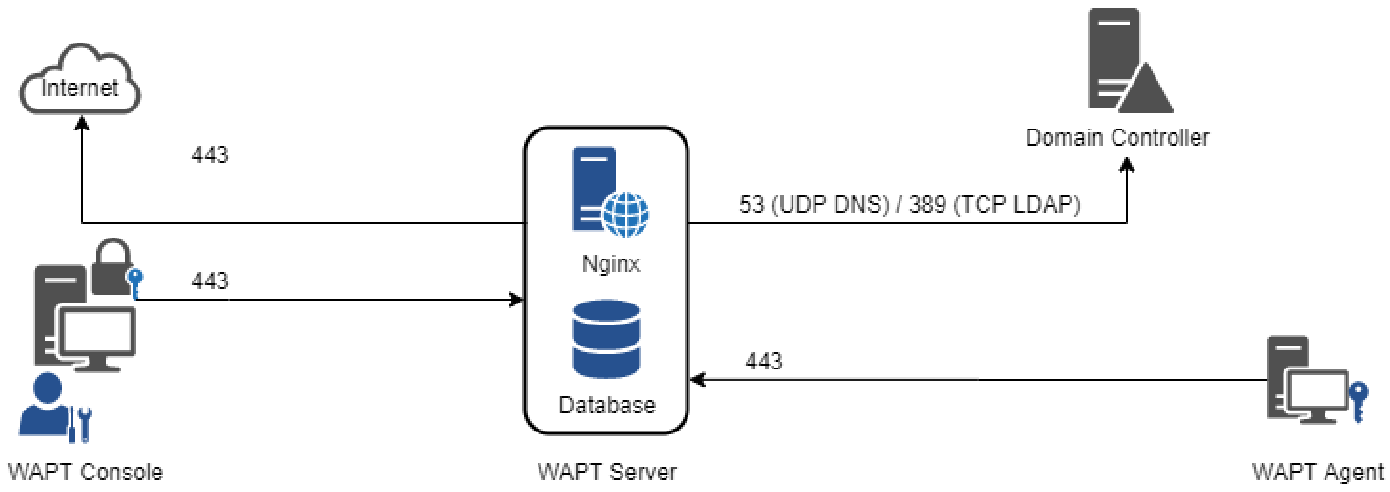


Fig. 145: Data-flow diagram of WAPT

As you can see, only ports 80 and 443 must be opened for incoming connections as the WAPT frameworks works with websockets initiated by the WAPT agents.

Configuring the firewall for WAPT Server on Debian Linux

By default on Debian Linux, no firewall rule applies.

- disable **ufw** and install **firewalld** instead:

```
ufw disable
apt update
apt -y install firewalld
```

- simply apply this **firewalld** configuration:

```
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
```

Configuring the firewall for WAPT Server on CentOS

- simply apply this **firewalld** configuration:

```
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 Configuring Kerberos authentication

Note:

- without Kerberos authentication, you have to either trust initial registration or enter a password for each workstation on initial registration;
- for more information, visit the documentation on *registering a machine with the WAPT Server* and *signing inventory updates*;
- the Kerberos authentication will be used only when registering the device;

Installing the Kerberos components and configuring krb5.conf file

```
#Debian
apt install krb5-user msktutil libnginx-mod-http-auth-spnego

#CentOS
yum install krb5-workstation msktutil nginx-mod-http-auth-spnego
```

Note: The feature is not available with a WAPT Windows server

Modify the `/etc/krb5.conf` file and **replace all the content with the following 4 lines** replacing **MYDOMAIN.LAN** with your Active Directory domain name (i.e. *<MYDOMAIN.LAN>*).

Attention: `default_realm` must be written with **ALL CAPS**!!

```
[libdefaults]
  default_realm = MYDOMAIN.LAN
  dns_lookup_kdc = true
  dns_lookup_realm=false
```

Retrieving a service keytab. Use the :*command:`kinit* and **klist**. You can use an *Administrator* account or any other account with the delegated right to join a computer to the domain in the proper destination container (by default *CN=Computers*).

In the shell transcript below, commands are in black and returned text is commented in light gray:

```
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
```

If the authentication request is successful, you can then create your HTTP Keytab with the **msktutil** command.

Be sure to modify the *<DOMAIN_CONTROLER>* string with the name of your domain controller (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: Be sure to have properly configured your WAPT Server *hostname* before running these commands;

In order to double check your *hostname*, you can run `echo $(hostname)` and it must return the name that will be used by WAPT agent running on client workstations.

- apply the proper access rights to the `http-krb5.keytab` file:

  ```
  #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-configuring

You can now use post-configuration script to configure the WAPT Server to use Kerberos.

The post-configuration script will configure **Nginx** and the WAPT Server to use Kerberos authentication.

Hint: This post-configuration script must be run as **root**.

```
/opt/wapt/waptserver/scripts/postconf.sh --force-https
```

Kerberos authentication will now be configured.

Special use cases

My WAPT server does not have access to a writeable Active Directory

- connect to your Active Directory (Not a RODC);
- create a computer account *srvwapt*;
- add a SPN (Service Principal Name) on the *srvwapt$* account;

```
setspn -A HTTP/srvwapt.mydomain.lan srvwapt
```

- create a keytab for this WAPT server:

```
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: If the address of your WAPT server is different from your active directory domain, replace *HTTP/srvwapt.mydomain.lan@MYDOMAIN.LAN* with *HTTP/srvwapt.othername.com@MYDOMAIN.LAN*.

- transfer this file to `/etc/nginx/` (with **winscp** for example);
- apply the proper access rights to the `http-krb5.keytab` file:

```
#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
```

WAPT agent only have access to a RODC domain controller

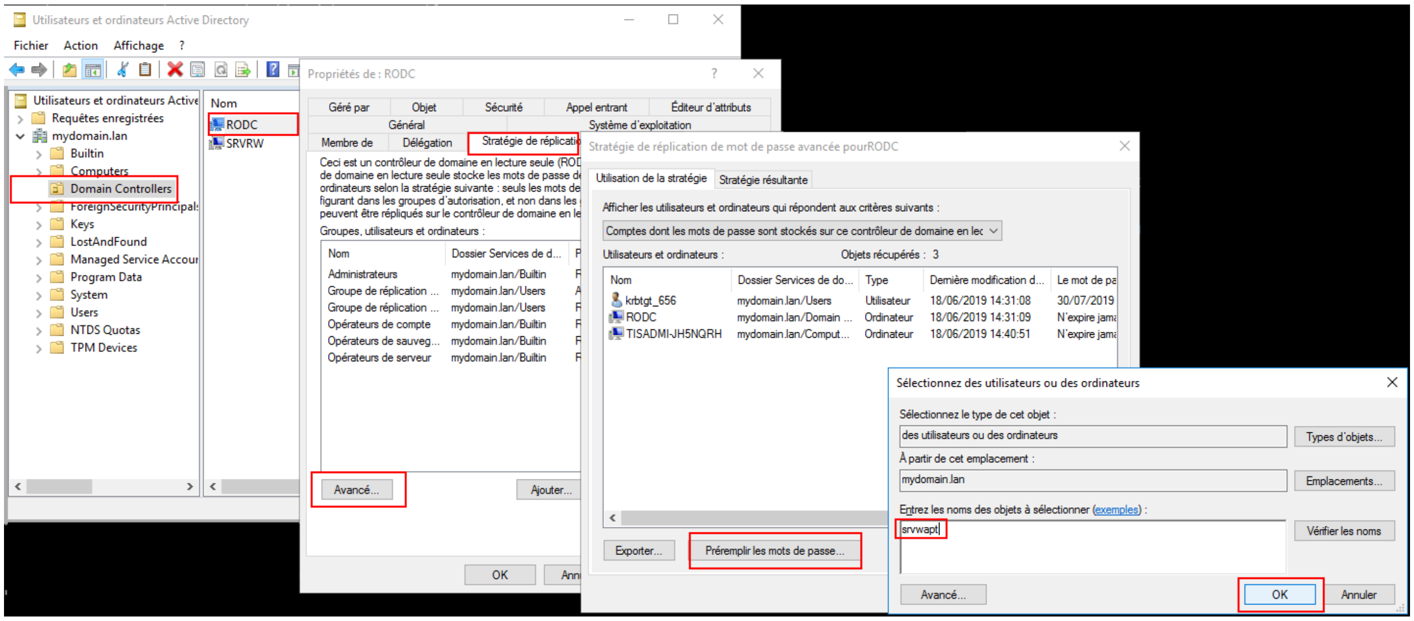
- for RODC (Read-Only Domain Controller), add the *srvwapt* account to the allowed password group for replication;
- remember to preload the password of the WAPT server with the different RODC servers;

You have multiple Active Directory domains with or without relationship

If you have multiple Active Directory domains, you must create one `keytab` per domain by following the procedure above, ex:

- `http-krb5-domain1.local.keytab`;
- `http-krb5-domain2.local.keytab`;
- `http-krb5-domain3.local.keytab`;

You will then have to merge all these `keytabs` into a unique `keytab`:



Ceci est un contrôleur de domaine en lecture seule (ROD
de domaine en lecture seule stocke les mots de passe de
ordinateurs selon la stratégie suivante : seuls les mots de
figurant dans les groupes d'autorisation, et non dans les
peuvent être répliqués sur le contrôleur de domaine en le

Groupes, utilisateurs et ordinateurs :

Nom	Dossier Services de d...	P
Administrateurs	mydomain.lan/Builtin	F
Groupe de réplication ...	mydomain.lan/Users	A
Groupe de réplication ...	mydomain.lan/Users	F
Opérateurs de compte	mydomain.lan/Builtin	F
Opérateurs de sauveg...	mydomain.lan/Builtin	F
Opérateurs de serveur	mydomain.lan/Builtin	F

Afficher les utilisateurs et ordinateurs qui répondent aux critères suivants :

Comptes dont les mots de passe sont stockés sur ce contrôleur de domaine en lec

Utilisateurs et ordinateurs : Objets récupérés : 3

Nom	Dossier Services de do...	Type	Dernière modification d...	Le mot de pa
krbtgt_656	mydomain.lan/Users	Utilisateur	18/06/2019 14:31:08	30/07/2019
RODC	mydomain.lan/Domain ...	Ordinateur	18/06/2019 14:31:09	N'expire jama
TISADMI-JH5NQRH	mydomain.lan/Comput...	Ordinateur	18/06/2019 14:40:51	N'expire jama

Sélectionnez le type de cet objet :
des utilisateurs ou des ordinateurs
À partir de cet emplacement :
mydomain.lan
Entrez les noms des objets à sélectionner (exemples) :
srvwapt

```
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 Activating the verification of the SSL / TLS certificate

When running the WAPT Server post-configuration script, the script will generate a self-signed certificate in order to enable HTTPS communications.

The WAPT agent checks the HTTPS server certificate according to the `verify_cert` value in section `[global]` in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Table 26: Options for `verify_cert`

Options for `verify_cert`	Working principle of the WAPT agent
`verify_cert` = 0	the WAPT agent will not check the WAPT Server HTTPS certificate
`verify_cert` = 1	the WAPT agent will check the WAPT Server HTTPS certificate using the certificate bundle `C:\Program Files (x86)\wapt\lib\site-packages\certifi\cacert.pem`
`verify_cert` = C:\Program Files (x86)\wapt\ssl\srvwapt.mydomain.lan.crt	the WAPT agent will check the WAPT Server HTTPS certificate with the certificate bundle `C:\Program Files (x86)\wapt\ssl\srvwapt.mydomain.lan.crt`

Hint: To quickly and easily enable verification of the https certificate, you can use the *Pinning* method.

Pinning the certificate

The *pinning of certificate* consists of verifying the SSL/ TLS certificate with a well defined and restricted bundle.

Hint: This method is the easiest when using a self-signed certificate.

For this, you need to launch the following commands in the Windows **cmd.exe** shell (with elevated privileges if UAC (User Account Control) is active).

If you already have a Windows **cmd.exe** shell open, close it and open a new shell so to take into account the updated environment variables:

```
wapt-get enable-check-certificate
net stop waptservice
net start waptservice
```

Validate the certificate with **wapt-get update**

When you have executed the **update** command, make sure that everything has gone well, and if in doubt check *Problems when enabling enable-check-certificate*.

Note:

- the command *enable-check-certificate* downloads the certificate `srvwapt.mydomain.lan.crt` in the folder `C:\Program Files (x86)\WAPT\ssl\server`;
- it then modifies the file `wapt-get.ini` to specify the value `verify_cert` = `C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt`;
- the WAPT agent will now verify certificates using the pinned certificate;

Attention: If you use the *certificate pinning* method, be reminded to archive the `/opt/wapt/waptserver/ssl` folder on your WAPT Server.

The file will have to be restored on your server if you migrate or upgrade your WAPT Server, if you want the WAPT agents to continue to be able to establish trusted HTTPS connections.

How to use a commercial certificate or certificates provided by your organization?

If the pinning method does not suit you, you can replace the self-signed certificate generated during the installation of **WAPT**.

Replace the old certificate with the new one in the folder `/opt/wapt/waptserver/ssl/` (linux) or `c:\wapt\waptserver\ssl\` (windows).

The new key pair must be in PEM encoded Base64 format

Note: Special case where your certificate has been signed by an internal Certificate Authority

Certificates issued by an internal *Certificate Authority* must have the complete certificate chain up to the *Certificate Authority*'s certificate.

You can manually add the certificate chain up to the Certificate Authority to the certificate that will be used by **Nginx**.

Example: `echo srvwapt.mydomain.lan.crt ca.crt > cert.pem`

For linux servers it is also necessary to reset the ACLs (Access Control List):

```
#Debian:
chown root:www-data /opt/wapt/waptserver/ssl/*.pem

#Centos:
chown root:nginx /opt/wapt/waptserver/ssl/*.pem
```

- restart **Nginx** to take into account the new certificates;
  - Linux:

    ```
    systemctl restart nginx
    ```

  - Windows:

    ```
    net stop waptnginx
    net start waptnginx
    ```

Configuring the WAPT agent

For a commercial certificate you can set `verify_cert` = 1 in `wapt-get.ini`.

For a certificate issued by an internal Certificate Authority, you must place the certificate in the `C:\Program Files (x86)\wapt\ssl\server\ca.crt` folder and specify the certificate path in `verify_cert` in the agent's `wapt-get.ini`.

To apply the new configuration to your entire fleet, you can regenerate a WAPT agent with the appropriate settings.

Verifying the certificate in the WAPT console

When the WAPT console first starts, it reads the content of `C:\Program Files (x86)\WAPT\wapt-get.ini` and it builds its configuration file `C:\Users\admin\AppData\Local\waptconsole\waptconsole.ini`.

This properly sets the `verify_cert` attribute for the HTTPS communication between the WAPT console and the WAPT Server.

6.12.4 Configuring authentication against Active Directory

New in version 1.5: Enterprise

Hint: Feature only available with WAPT **Enterprise**.

By default, the WAPT Server is configured with a single *SuperAdmin* account whose password is setup during initial post-configuration.

On large and security-minded network, this *SuperAdmin* account should not be used since it cannot provide the necessary traceability for administrative actions that are done on the network.

It is thus necessary to configure authentication against the *Organization*'s Active Directory for the *Administrators* and the *Package Deployers*; this will allow to use named accounts for administrative tasks.

Note:

- Active Directory authentication is used to authenticate access to the inventory via the WAPT Console;
- however, all actions on the WAPT equipped remote devices are based on X.509 signatures, so an *Administrator* will need both an Active Directory login **AND** a private key whose certificate is recognized by the remote devices to manage his installed base of devices using WAPT;
- only the *SuperAdmin* account and the members of the Active Directory security group **waptadmins** will be allowed to upload packages on the main repository (authentication mode by login and password);

Enabling Active Directory authentication

- to enable authentication of the WAPT server on Active Directory, configure the file `/opt/wapt/conf/waptserver.ini` as follows:

```
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
```

Settings	Value	Description
wapt_admin_group_dn	CN=waptadmins,OU=groups,	DN to the group name. All members of this group will be able to connect to WAPT
ldap_auth_server	srvads.mydomain.lan	LDAP server that will be used by WAPT
ldap_auth_base_dn	DC=mydomain,DC=lan	DN for the search
ldap_auth_ssl_enable	True/False	Default value: True

- restart **`waptserver`** with `systemctl restart waptserver`;

Warning: For Microsoft Active Directory, Microsoft has announced that *SimpleBind* authentication on MS-AD without SSL/TLS will be blocked by default from April 2020. If you don't have a certificate installed, you'll have to modify a registry key to have authentication working.

Note: By default Samba-AD does not allow *SimpleBind* authentication without SSL/TLS. If you do not have a valid certificate you'll need to modify the `ldap server require strong auth` parameter in `/etc/samba/smb.conf`. For more information you may refer to Tranquil IT documentation on https://dev.tranquil.it/samba/en/index.html.

Enabling SSL/ TLS support for the LDAP connection to the Active Directory Domain Controller

By default, authentication on Active Directory relies on LDAP SSL (default port 646).

SSL/ TLS is not enabled by default on Microsoft Active Directory until a SSL certificate has been configured for the Domain Controller.

Note: The WAPT Server uses the Certificate Authority *bundles* from the operating system (CentOS) for validating the SSL/ TLS connection to Active Directory.

If the Active Directory certificate is self-signed or has been signed by an internal CA, you'll need to add these certificates to the certificate store of CentOS.

Add a *Certificate Authority* in the `/etc/pki/ca-trust/source/anchors/` and update the CA store.

```
cp cainterne.pem /etc/pki/ca-trust/source/anchors/cainterne.pem
update-ca-trust
```

- once you have setup LDAP SSL/ TLS on your Active Directory (please refer to Microsoft documentation for that), then you can enable support for SSL/ TLS security for AD in `/opt/wapt/conf/waptserver.ini`:

  ```
  ldap_auth_ssl_enabled = True
  ```

- restart **`waptserver`** with `systemctl restart waptserver`;

6.12.5 Configuring Client-Side Certificate Authentication

New in version 1.7: Enterprise

Hint: This feature is only available with WAPT **Enterprise**.

If your business needs a public WAPT server on Internet, it can be secured with **Client-Side Certificate Authentication**.

That configuration restricts the visibility of the WAPT Server to registered clients only. It is done by relying on the WAPT agent's private key generated during registration. It works as follows:

- the WAPT agent sends a CSR (Certificate Signing Request) to the WAPT server which the WAPT server signs and sends back to WAPT agent;
- using the signed certificate, the agent can access protected parts of the **`Nginx`** web server;

Note: We advise you to enable Kerberos or login / password registration in WAPT Server post-configuration.

Enabling Client-Side Certificate Authentication

- be sure to unset the custom headers relative to client side authentication results when requests are proxied without being checked by nginx ssl module. These headers are trusted by the waptserver if `X-Ssl-Authenticated` is SUCCESS and `waptserver.ini` parameter `use_ssl_client_auth` is set to **True**:

```
location / {
    ...
    proxy_set_header X-Ssl-Authenticated "";
    proxy_set_header X-Ssl-Client-DN "";
```

- add a **`Nginx`** configuration file `/etc/nginx/certificate-auth.conf`. This file is used to restrict access to specific actions with Certificate Authentication:

```
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;
}
```

Example config file:

```
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";
    }

    location / {
```

(continues on next page)

(continued from previous page)

```
        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;
```

(continues on next page)

(continued from previous page)

```
            }
        }

}
```

Attention: Be careful, as of 2023-01-10, WAPT does not support CRL, which means that when you delete a machine in the WAPT console, the machine will still have access to the WAPT repository.

6.13 Enhancing your productivity with WAPT

You will find in this section of the documentation some inspiring and time saving methods to make the most of your WAPT setup.

6.13.1 Simplifying the imaging of your workstations

We observe that many companies and administrations include software and configurations in the Windows images they deploy on their fleets of machines.

If you use WAPT, lose this habit now and forever! Why?

- Each time you make a new image, you waste a lot of time installing software and configuring it. You are very limited in the user configurations that you will be able to include in your image.
- each time you make a new image, if you are serious about it, you will have to keep track of the changes in a text document, a spreadsheet, or a change management tool. It's a very heavy and thankless burden. And you know as well as I do, what's ungrateful is usually badly done!
- finally, if you introduce in your image security configurations, network configurations, or configurations to limit the intrusion of Windows telemetry, these configurations can disrupt the normal functioning of WAPT, it will complicate future diagnostics, and it will discourage you from using an efficacious tool very capable of freeing up your time.

What do you suggest to do then?

Tranquil IT recommends:

- to make only one raw image per OS type with MDT or Fog (win10, win2016, etc) without any configuration or software. **Put only the system drivers** you need for your image deployment in the MDT or Fog directories provided for this purpose;
- to configure your WAPT server to register hosts with a random UUID to avoid *UUID Bios or FQDN conflicts*;
- to create as many Organizational Units as you have machine types in the *CN=Computers* OU (ex: *standard_laptop*, *hardened_laptop*, *workstations*, *servers*, etc) in your Active Directory;
- to configure your Active Directory to distribute the WAPT Agent GPO to the different Host Organizational Units; This way, you can opt for fine grained configurations of your `waptagent.ini` for the hosts attached to each OU.

Note: Alternatively, you may include a generic WAPT agent in your OS image.

- to properly configure your DHCP to redirect the PXE to the correct system images;

- to properly configure your MDT or Fog to register the machine in the correct Organizational Unit of your Active Directory;
- to create as many WAPT security configuration packages as you have Organizational Units created above. Thus, you will be able to apply different security profiles depending on the type of machine. These packages will include the desired security configurations (telemetry suppression, firewall configuration, etc);

Hint: To save you time, you can base your security configuration strategy on security WAPT packages already available in the WAPT Store, you will only need to complete them according to your organization's specific security requirements.

- to create in the *CN=Computers* OU as many Organizational Units as there are types of computer usage in your organization (*accounting*, *point_of_sale*, *engineering*, *sedentary_sales*, etc);
- to create generic WAPT packages of your software applications with their associated configurations;

Note: To save you time and effort, you can import many proven WAPT packages from Tranquil IT's public stores or subscribe to Tranquil IT's private stores.

- to associate the WAPT packages created above with the Organizational Units of the typologies of computer use;

How does the scenario work?

- you receive or the IT manager at the remote site receives a new machine in its box;

Hint: Alternatively, you choose or the IT manager at the remote site chooses to switch an existing machine from win7 to win10. You will either have, or he will have previously backed up the user directory(s) to a network drive or another convenient storage media.

For this purpose, you may build a WAPT package that, upon execution, will zip the `C:Users` on the win7 computer, name it with the computer's FQDN, password protect the compressed file using *this procedure* and upload it to a web server or a network share. That same WAPT package can do the reverse process and reinstall the user files after the host has been re-imaged.

- you configure MDT or Fog with the machine's MAC address so that it gets the right system image through DHCP and is positioned in the right Organizational Unit at the end of the cloning process;
- the expected system image is downloaded on the machine in masked time, the machine is placed in the right Organizational Unit;
- the WAPT agent registers the machine with the WAPT server, it appears in the WAPT console;

Hint: If your machines are from a win7 to win10 update, then you will remove the old win7 machines from the WAPT inventory as they will be duplicated due to your choice of random UUID configuration; these machines will be easy to find in the WAPT console because they will be marked as win7 with the same MAC address or the same FQDN as your new machine in win10; after removing the win7, your inventory will be clean and up to date in your WAPT console;

- the WAPT agent detects that it is in an Organizational Unit that requires a particular software set and a particular security configuration;
- the WAPT Agent downloads and executes software packages and security configuration packages in hidden time; the WAPT Agent automatically removes delegated rights that are rendered useless after joining the domain to prevent them from being subsequently exploited in an unauthorized manner;

- either by group of machines or machine by machine, you finalize the configuration of the machines by assigning specific WAPT packets to them;

Hint: If you want, you can even leave the final configuration step to your users by configuring WAPT self-service for them (printer configurations, special software needs, etc).

Conclusion

With little effort, you now have full control over a fleet of several hundreds or even thousands of geographically dispersed machines. All your installations are documented, your users work with adequate rights and you benefit from a clear visibility on your users' tools and uses. In this way, the past is no longer an imponderable burden for you and an obstacle to your future projects.

6.14 Creating WAPT packages

Attention: Installing *tis-pyscripter* will install PyScripter *IDE*. **`PyScripter`** is required for packaging WAPT packages.

Please first visit the documentation on *setting up the environment for developing WAPT packages*.

To create and customize WAPT packages, follow that documentation and you will quickly become a master of WAPT.

6.14.1 Setting up your WAPT development and test environment

Prerequisites

Attention:

- it is **required** to be a *Local Administrator* of the machine to use WAPT's integrated environment for developing WAPT packages;
- we advise you to create/ edit packages in a fully controlled environment that is safe and *disposable*;
- the usage of a disposable virtual machine (like Virtualbox) is recommended;

Make sure that you have your private key stored on the development computer

In the WAPT console, the fields *private key* and *prefix* must be filled.

Install the *tis-pyscripter* WAPT package

- import the *tis-pyscripter* package in your local repository and install it on your development computer;

Recommendations regarding the test environment

The recommended method to correctly test your WAPT packages is to use a representative sample of machines in your inventory. So the more heterogeneous your installed base of machines, the larger your sample must be.

This method consists of testing the installation of the package on as many platforms and configurations as possible, so to improve its reliability, before the WAPT package is transferred to production repositories.

Testing method

Operating systems and architectures

- Windows XP;
- Windows 7;
- Windows 10;
- Windows Server 2008 R2;
- Windows Server 2012 and R2;
- x86;
- x64;
- Physical and virtual machines;
- laptops;

Hint: When possible, RC and Beta version of Operating Systems should be tested (ex: Windows 10 Creators Update).

State of Windows Updates

- **Microsoft Windows machine without any Windows update installed**: the objective is to detect Windows updates that are required for the software to work properly and to adapt the WAPT package accordingly;
- **Microsoft Windows machine with all the latest Windows updates**: the objective is to detect Windows updates that break the package and to adapt the WAPT package accordingly;

State of software installations

- **Machines with many installed packages**: the objective is to detect a possible dependency with an existing application;
- **Machines with many installed packages**: the objective is to detect a possible conflict with an existing application;
- **Install older versions of the software**: it is possible that the software installer does not support uninstalling a previous version of the software, in this case, the WAPT package will have to remove older versions of the software before installing the new version;

6.14.2 Creating a package template from the WAPT console

New in version 1.3.12.

Hint: To create WAPT packages directly from the WAPT console, it is necessary to have installed the WAPT development environment *tis-waptdev*.

Creating a package template from the WAPT console

In that example, we use the 7-zip MSI setup downloaded from the 7-zip official website.

- download the 7-zip MSI installer:
  - download 7-zip MSI x64;
  - download 7-zip MSI x86;
- create a WAPT package Template from the installer;

  In the WAPT console, click on *Tools* → *Make package template from setup file*

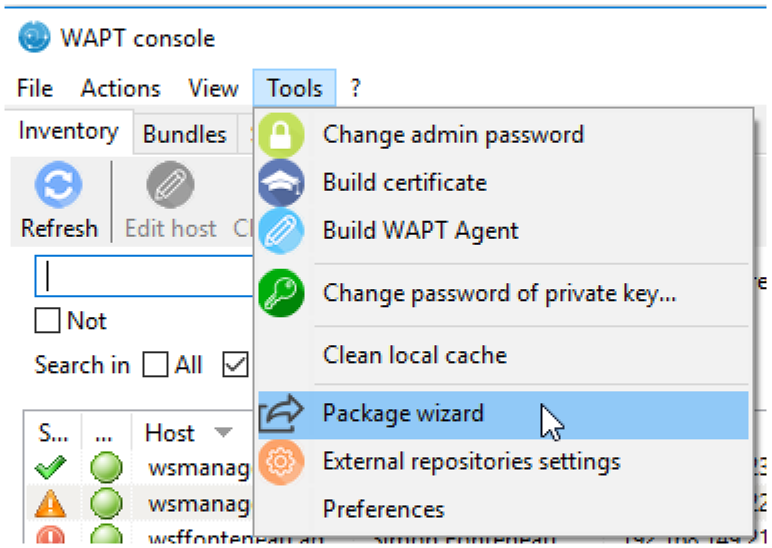
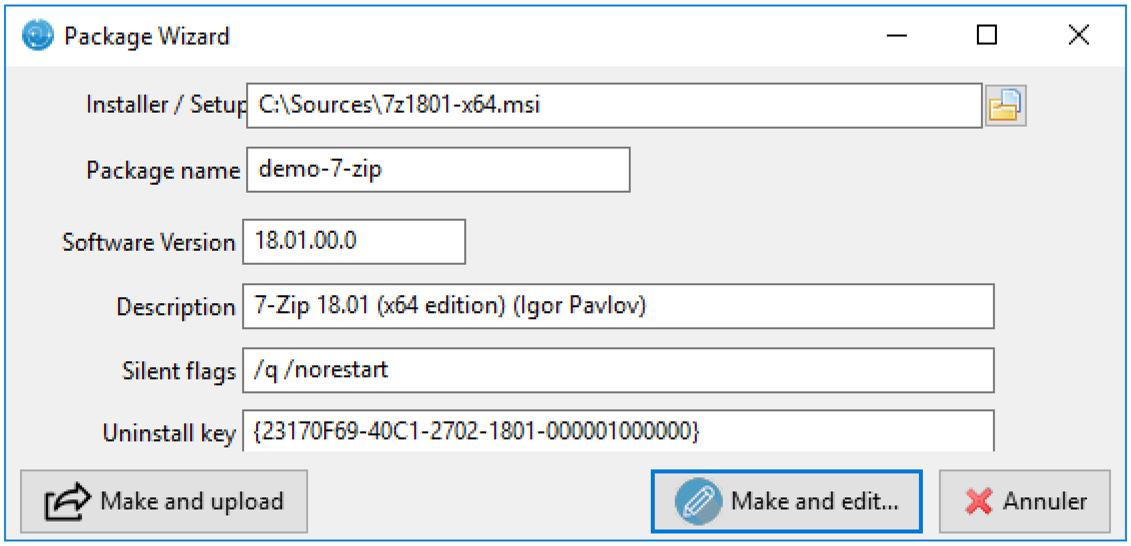


Fig. 146: Pyscripter - WAPT console window for creating a package template

Select the downloaded MSI setup file and fill in the required fields. Verify that the package name does not contains any version number.



Installer / Setup	C:\Sources\7z1801-x64.msi
Package name	demo-7-zip
Software Version	18.01.00.0
Description	7-Zip 18.01 (x64 edition) (Igor Pavlov)
Silent flags	/q /norestart
Uninstall key	{23170F69-40C1-2702-1801-000001000000}

Fig. 147: Informations required for creating the package

- Two solutions are available:
  - click on *Make and edit . . . .* (recommended) to launch package customization;
  - click on *Build and upload* to directly build and upload the package.

Attention: The button *Build and upload* directly uploads the package into the private repository without testing.

This method works relatively well with MSI installers because their installation is more standardized.

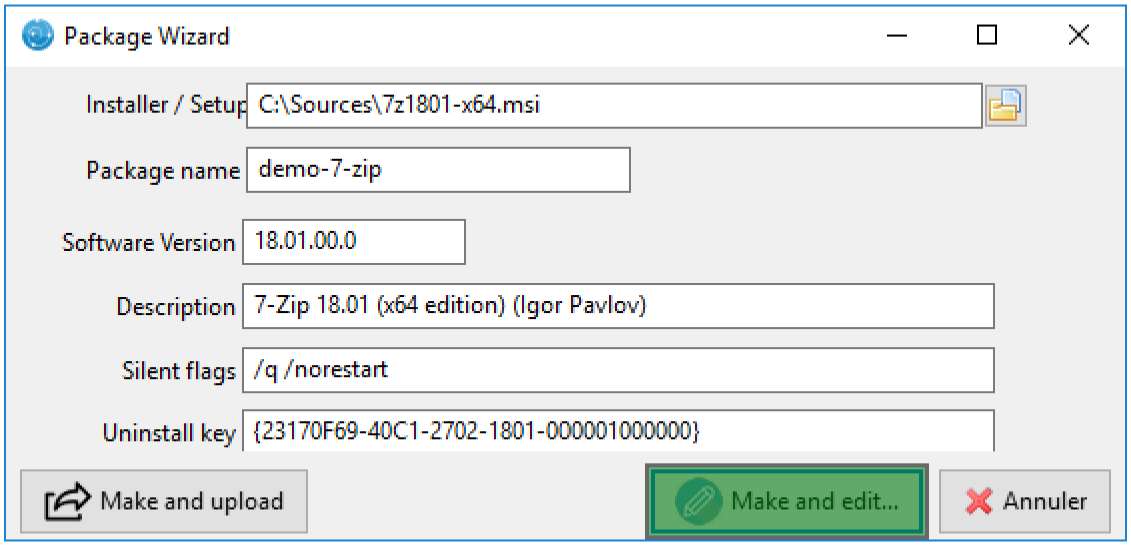
However, the second method that consists of first testing locally the package before uploading is the recommended method.

Customize the package before build-upload

Before uploading a package to your WAPT repository, you may choose to customize its behavior to your Organization's needs by editing it with **`PyScripter`**.

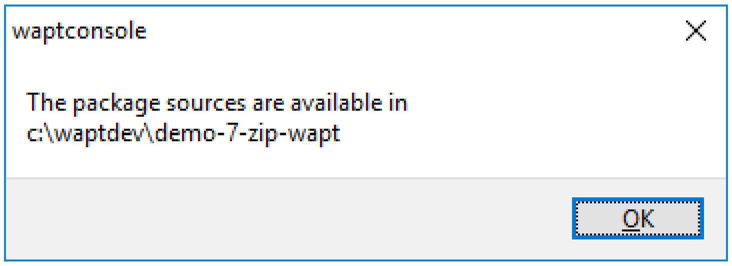
When creating the package template, click on *Make and edit . . . .*.

The **`PyScripter`** IDE allows to edit files in the WAPT package.



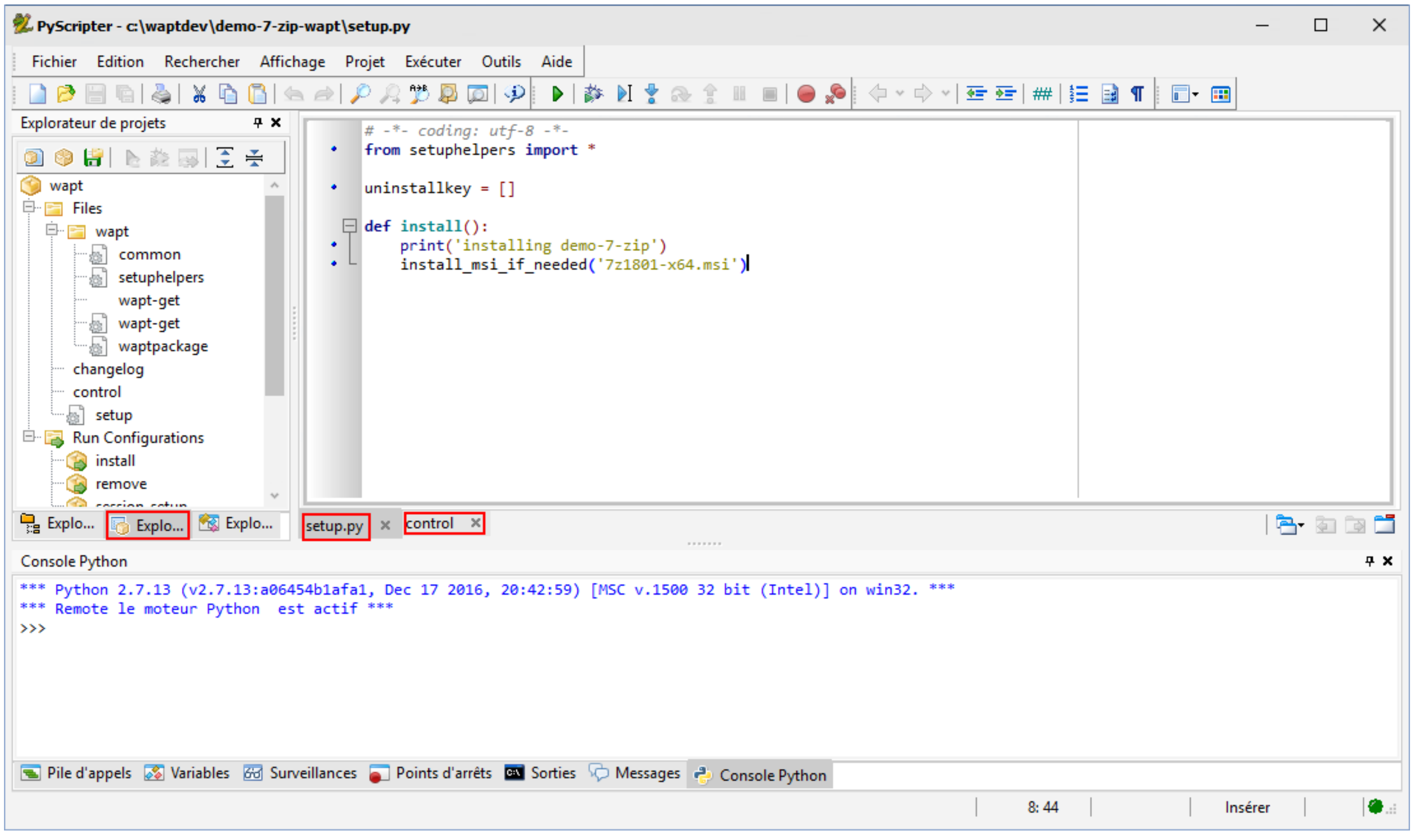
Installer / Setup	C:\Sources\7z1801-x64.msi
Package name	demo-7-zip
Software Version	18.01.00.0
Description	7-Zip 18.01 (x64 edition) (Igor Pavlov)
Silent flags	/q /norestart
Uninstall key	{23170F69-40C1-2702-1801-000001000000}

Fig. 148: PyScripter - Informations required for creating the package



The package sources are available in c:\waptdev\demo-7-zip-wapt

Fig. 149: PyScripter - The package has been created



```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing demo-7-zip')
    install_msi_if_needed('7z1801-x64.msi')
```

```
*** Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote le moteur Python  est actif ***
>>>
```

Fig. 150: PyScripter - Customizing a package with Pyscripter

Presentation of Pyscripter

PyScripter project explorer

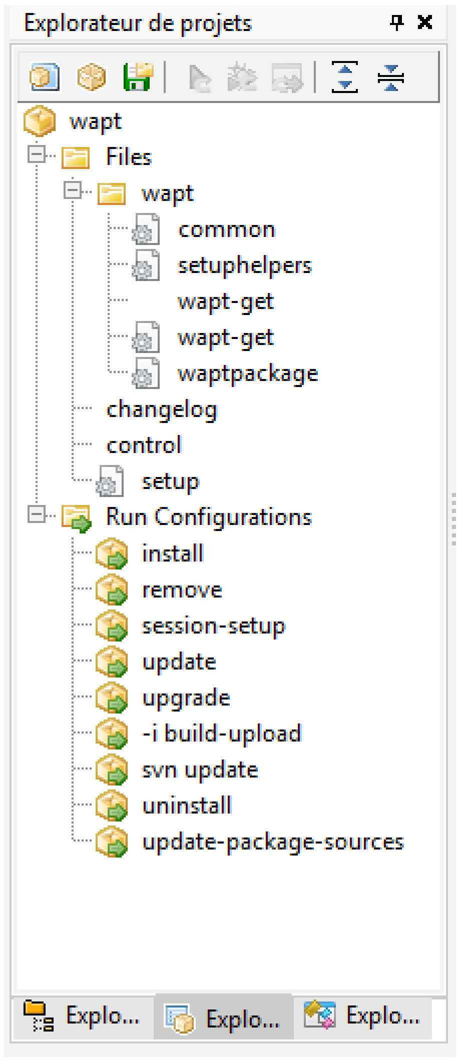


Fig. 151: PyScripter - project explorer

The PyScripter project explorer lists the different files that you might need, notably the `control` file and the `setup.py` file.

Run Configurations

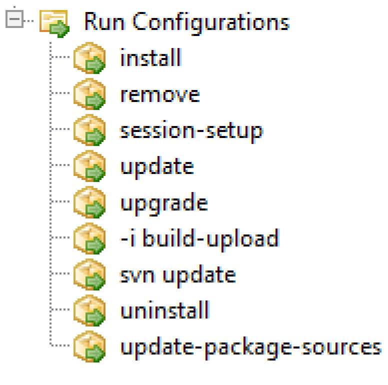
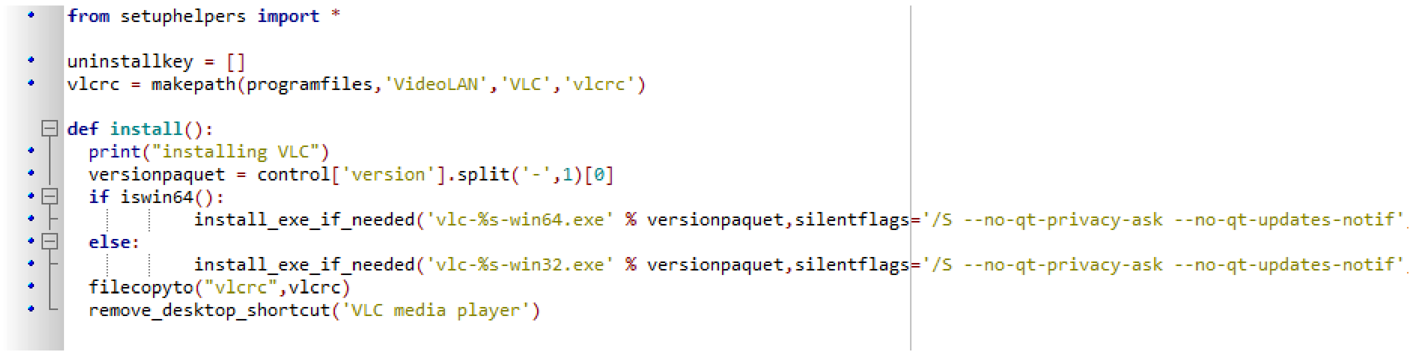


Fig. 152: PyScripter - Run commands in the PyScripter project explorer

The **Run** option in the project explorer of:program:*PyScripter* will allow you to launch actions on the packages that you are editing.

Editor panel



```
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')
```

Fig. 153: PyScripter - Editor panel

The edition panel in **PyScripter** allows to edit the `setup.py` file and the `control` file.

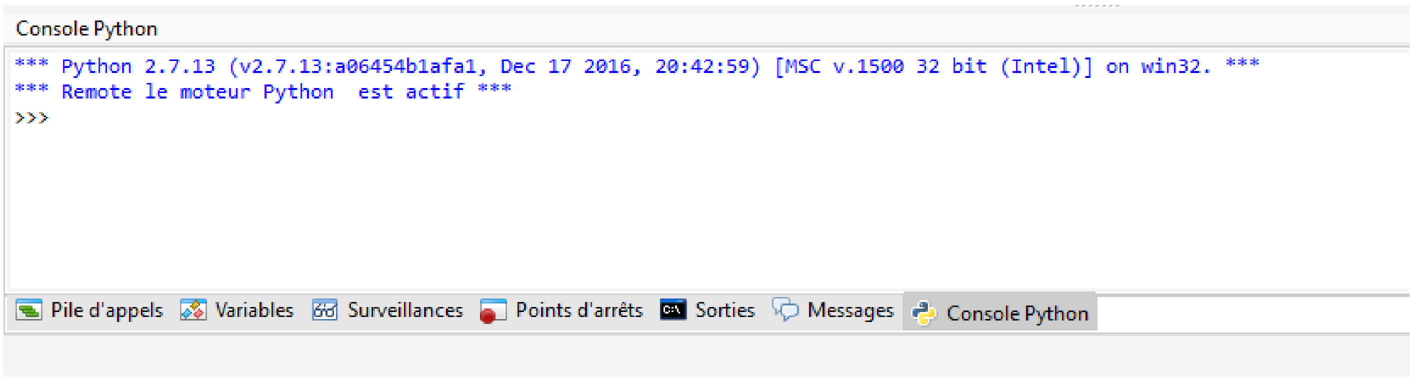
Python Console

This is the python console visible in **PyScripter**, it will allow you to display the python output when you execute **Run** commands.

You can also use it to test/ debug portions of your script `setup.py`.

To learn more about the composition of a wapt package, visit the documentation on the *structure of a WAPT package*.

To customize a package, please visit the documentation on *customizing your WAPT packages*.

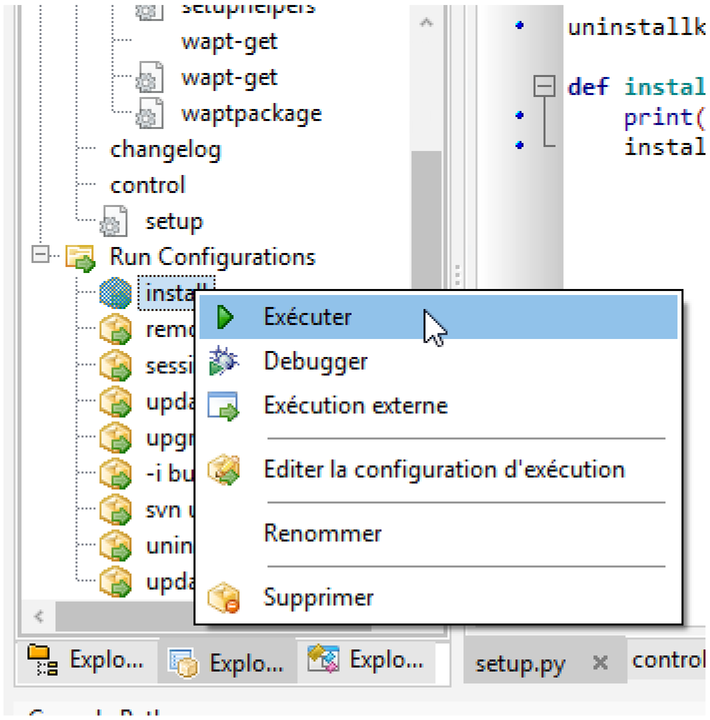


```
*** Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote le moteur Python  est actif ***
>>>
```

Fig. 154: PyScripter - Python console in PyScripter

Testing locally the installation of the WAPT package

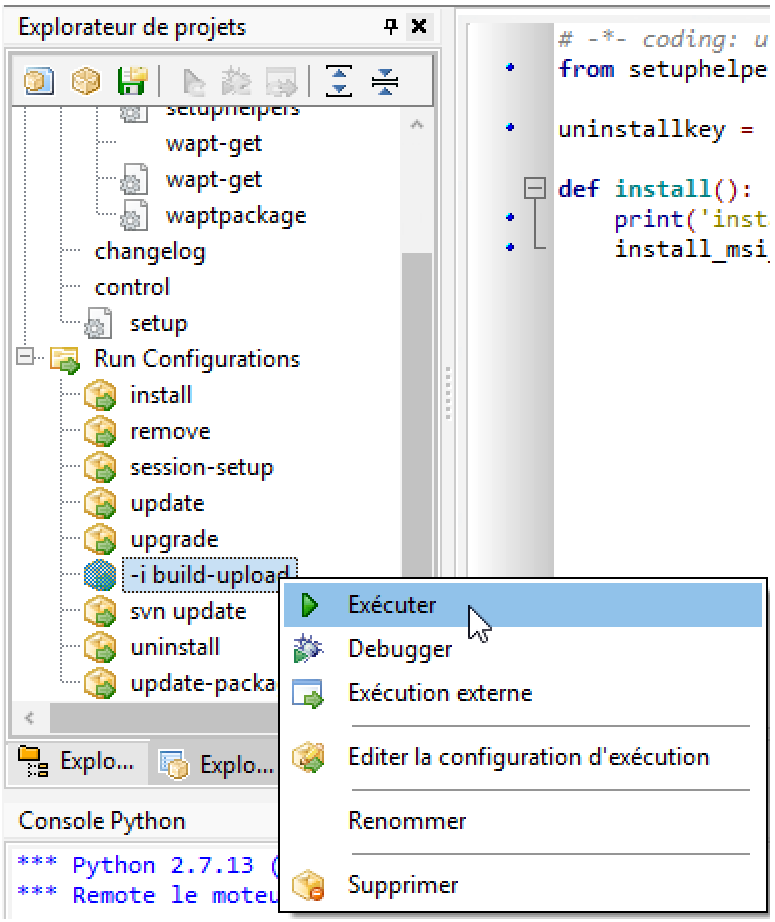
You can then test the launch of an installation on your development station.



The PyScripter Console allows you to check whether the installation went well.

Building the package and sending it to the WAPT server

- once the package is ready, build it and send it to the WAPT server;

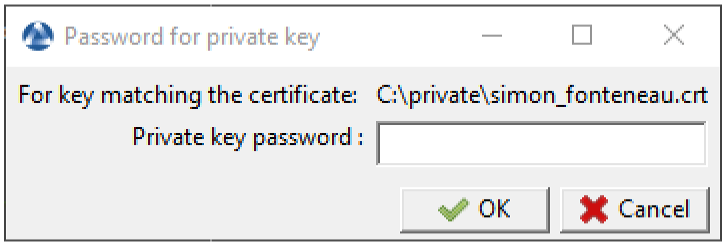


```
# -*- coding: u
from setuphelpe

uninstallkey =

def install():
    print('inst
    install_msi
```

```
*** Python 2.7.13 (
*** Remote le moteu
```

Fig. 155: Option "-i build-upload" of PyScripter project

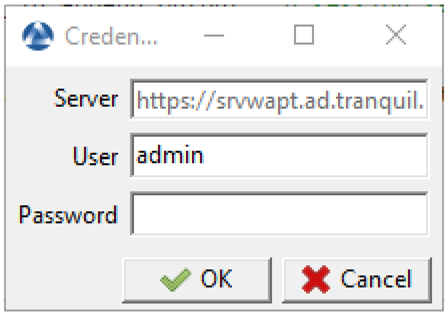
- enter the password of your private key (to sign your WAPT package);



For key matching the certificate: C:\private\simon_fonteneau.crt

Private key password :

- enter the username and password to send the WAPT package to the server;
- the package is now available and visible in the WAPT Console in the tab *private deposit*.
- click on *update available packages* to refresh the list of available WAPT packages;



6.14.3 WAPT package structure

A WAPT package is a zip file containing several things:

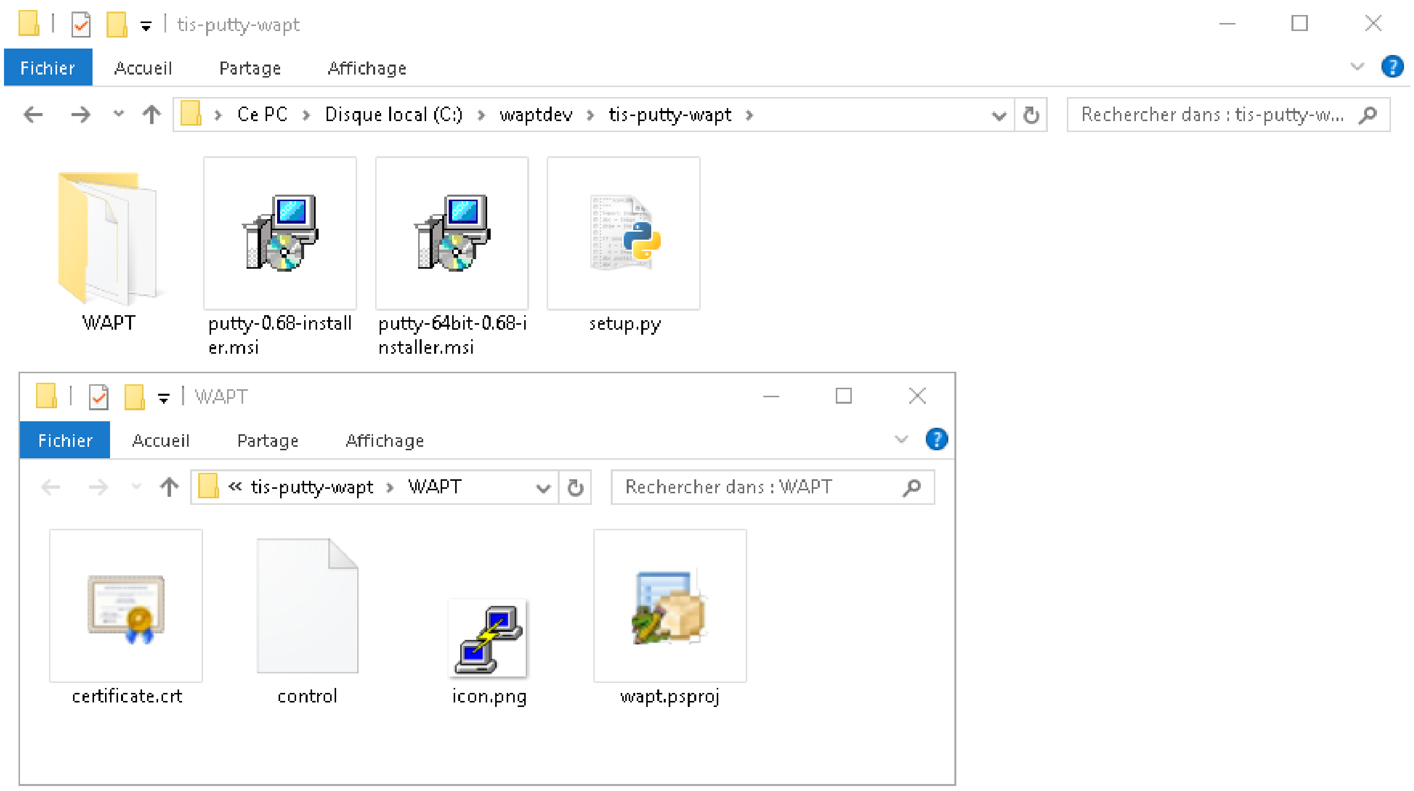

tis-putty-wapt
Fichier Accueil Partage Affichage
Ce PC › Disque local (C:) › waptdev › tis-putty-wapt
Rechercher dans : tis-putty-w...
WAPT
putty-0.68-installer.msi
putty-64bit-0.68-installer.msi
setup.py
WAPT
Fichier Accueil Partage Affichage
« tis-putty-wapt › WAPT
Rechercher dans : WAPT
certificate.crt
control
icon.png
wapt.psproj

Fig. 156: WAPT package structure

- a file `setup.py`;
- one or several binary files;
- some additional optional files;
- a `control` file in the `WAPT` folder;
- a `icon.png` file in the `WAPT` folder;
- a `certificate.crt` file in the folder `WAPT`;

- a `manifest.sha256` file in the folder `WAPT`;
- a `signature.sha256` file in the folder `WAPT`;
- a `wapt.psproj` file in the folder `WAPT`, this file is used to store the **PyScripter** configuration data for the WAPT package;
- since WAPT 1.8, a hidden `.vscode` folder that contains a `launch.json` and a `settings.json` file used to store the **VScode** configuration data for the WAPT package;

The *control* file

The `control` file is the identity card of a package.

```
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
```

Table 27: Description of options of the control file

Settings	Description	Example value
`package`	Package name	tis-geogebra
`version`	Package version, can not contain more than 5 delimiters	5.0.309.0-0
`architecture`	Processor architecture	x64
`section`	Package type (`host`, `group`, `base`)	base
`priority`	Package install priority (optional, not used as of 1.5.15)	Not mandatory for the moment
`maintainer`	Author of the package	Arnold Schwarzenegger terminator@mydomain.lan
`description`	Package description that will appear in the console and on the web interface	The Graphing Calculator for Functions,Geometry, Algebra, Calculus, Statistics and 3D
`description_fr`	Localized description of the package	Calculatrice graphique
`depends`	Packages that must be **installed** before installing the package	tis-java
`conflicts`	Packages that must be **uninstalled** before installing the package	tis-graph
`maturity`	Maturity level (*BETA*, *DEV*, *PROD*)	PROD
`locale`	Language environment for the package	fr,en,es
`target_os`	Accepted Operating System for the package	windows,mac,linux
`min_os_version`	Minimum version of Windows for the package to be seen by the WAPT agent	6.0
`max_os_version`	Maximum version of Windows for the package to be seen by the WAPT agent	8.0
`min_wapt_version`	WAPT's minimal version for the package to work properly	1.3.8
`sources`	Path to the SVN location of the package (**`source`** command)	https://srv-svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/
`installed_size`	Minimum required free disk space to install the package	254251008
`impacted_process`	Indicates a list of impacted processes when installing a package	firefox.exe
`audit_schedule`	Periodicity of execution of the audit function in the WAPT package	60
`editor`	Editor of the software package	Mozilla
`license`	Reference of the software license	GPLV3
`keywords`	Set of keywords describing the WAPT package	Productivity,Text Processor
`homepage`	Official homepage of the software embedded in the WAPT package	https://www.tranquil.it/
`signer`	CommonName (CN) of the package's signer	Tranquil IT
`signer_fingerprint`	Fingerprint of the certificate holder's signature	2BAFAF007C174A3B00F12E9CA1E749
`signature`	SHA256 hash of the package	MLOzLiz0qCHN5fChdylnvXUZ8xNJj4r
`signature_date`	Date when the package was signed	20180307-230413
`signed_attributes`	List of package's attributes that are signed	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: If the `control` file contains special characters, the `control` file must be saved in **UTF-8 (No BOM)** format.

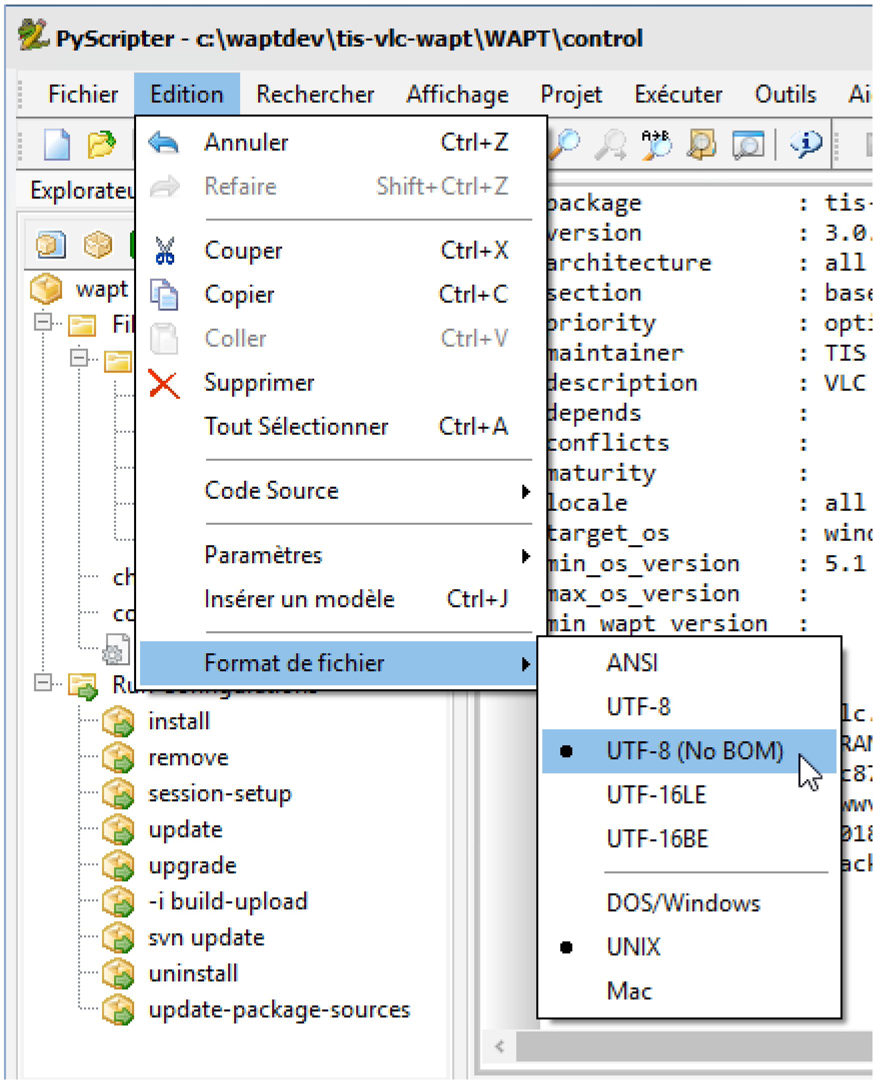


Fig. 157: PyScripter - UTF-8 (No BOM)

Fields details

package

WAPT package name, without any accent, nor space, nor any special or uppercase character.

version

Preferably, always start with the packaged software version (**digits only**) split by points (.) and finish with the WAPT packaging version separated by a dash (-) character.

architecture

New in version 1.5.

Defines whether the package may be installed on x64 or x32 processor equipped computers.

Note: A x64 package will be invisible for a WAPT agent installed on a x86 machine.

Allowed values:

- **x86**: the package is designed for 32bit computers;
- **x64**: the package is designed for 64bit computers;
- **all**: the package is designed for 32bit or 64bit computers;

section

- **host**: host package;
- **group**: group package;
- **base**: software package;
- **unit**: OU package;

priority

This option is not supported at this time. That field will be used to define package installation priority. This feature will become useful to define mandatory security updates.

maintainer

Defines the WAPT package creator.

Note: To define the WAPT package creator's email address may be useful.

description

Describes the functionality of the package that will appear in the console, on the local web interface http://127.0.0.1:8088 and in `wapt-get` command lines.

Hint: Adding a field `description_fr` or `description_es` allows you to internationalize the description of your package. If the language does not exist, the WAPT agent will use the default language description.

depends

Defines the packages that must be installed before, for example *tis-java* is a dependency for the *LibreOffice* package and *tis-java* must be installed before *LibreOffice*.

Several dependencies may be defined by splitting them with commas (,).

example:

```
depends: tis-java,tis-firefox-esr,tis-thunderbird
```

conflicts

Works as the opposite of *depends*.

conflicts defines package(s) that must be removed before installing a package, for example *tis-firefox* must be removed before the package *tis-firefox-esr* is installed, or *OpenOffice* must be removed before *LibreOffice* is installed.

Several conflicts may be defined by splitting them with commas (,).

maturity

New in version 1.5.1.19.

Defines the maturity of a package.

By default, WAPT gents will see packages flagged as *PROD* and packages with an empty maturity.

For a computer to see packages with different maturity levels, you will have to configure the *maturities* atrtibute in `wapt-get-ini`

locale

New in version 1.5.1.19.

Defines the language of the WAPT package.

A WAPT agent will see by default packages that are configured for its language environment and packages with no language specified.

For a computer to see a package in another language, you will have to configure the *locales* in `wapt-get.ini`.

```
locales = fr,en,es
```

The language filled in the field must be in ISO 639-1 format.

target_os

New in version 1.5.1.18.

Defines the Operating System for the package.

A WAPT agent will see by default packages that are configured for its operating system and packages with no operating system specified.

Since version 1.8 the field *target_os* can either be *windows*, *mac*, *linux* or left empty.

min_os_version

New in 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

New in 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

New in version 1.3.8.

WAPT minimum version to install a package

Note: With functionalities in WAPT evolving, some functions that you may have used in old packages may become obsolete with newer versions of WAPT agents.

sources

Defines a SVN repository, for example:

- https://svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/
- https://svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/

This method allows to version a package and collaboratively work on it.

Hint: Package versioning is particularly useful when several people create packages in a collaborative way. This function is also useful to trace the history of a package if you are subject to Regulations in your industry.

installed_size

Defines the required minimum free disk space to install the package.

Example:

```
installed_size: 254251008
```

The testing of available free disk space is done on the `C:\Program Files` folder.

The value set in *installed_size* must be in bytes.

Hint: To convert storage values to bytes, visit https://bit-calculator.com/.

impacted_process

New in version 1.5.1.18.

Indicates processes that are impacted when installing a package.

Example:

```
impacted_process : firefox.exe,chrome.exe,iexplorer.exe
```

This field is used by the functions **`install_msi_if_needed`** and **`install_exe_if_needed`** if *killbefore* has not been filled.

impacted_process is also used when uninstalling a package. This allows to close the application if the application is running before being uninstalled.

audit_schedule

New in version 1.6.

Periodicity of execution of audit checks.

Example:

```
audit_schedule : 60
```

The periodicity may be indicated in several ways:

- an integer (in minutes);
- an integer followed by a letter (*m* = minutes, *h* = hours, *d* = days, *w* = weeks);

editor

New in version 1.6.

Software editor of the binaries embedded in the WAPT base package.

Example:

```
editor: Mozilla
```

The values may be used as filters in the WAPT console and with the self-service.

keywords

New in version 1.6.

Keyword list to categorize the WAPT package.

Example:

```
keywords: editeur,bureautique,tableur
```

The values may be used as filters in the WAPT console and with the self-service.

licence

New in version 1.6.

Reference of the software license for the embedded software binaries.

Example:

```
licence: GPLV3
```

The values may be used as filters in the WAPT console and with the self-service.

homepage

New in version 1.6.

Official homepage of the software binaries embedded in the WAPT package.

Example:

```
homepage: https://wapt.fr
```

The values may be used as filters in the WAPT console and with the self-service.

signer

Automatically filled during package signature.

CN of the certificate. It is typically the signer's full name.

signer_fingerprint

Automatically filled during package signature.

Private key fingerprint of the package signer.

signature

Automatically filled during package signature.

Signature of the attributes of the package.

signature_date

Automatically filled during package signature.

Date when the attributes of the package have been signed.

signed_attributes

Automatically filled during package signature.

List of the package’s attributes that are signed

The *setup.py* file

import setuphelpers

That line is found at the beginning of every WAPT package:

```
from setuphelpers import *
```

The package imports all setuphelpers functions.

Setuphelpers is a WAPT library that offers many methods to more easily develop highly functional packages.

uninstallkey list

We then find:

```
uninstallkey = ['tisnaps2','Mozilla Firefox 45.6.0 ESR (x86 fr)']
```

We associate here a list of *uninstall keys* to the package. When a package is removed, the WAPT agent looks up the *uninstallkey* in the registry associated to the package. This *uninstallkey* will indicate to WAPT the actions to trigger to remove the software.

Even if there is no *uninstallkey* for a software, it is mandatory to declare an empty *uninstallkey* array:

```
uninstallkey = []
```

Function install()

Then comes the `setup.py` function declaration.

It is the recipe of the WAPT package, the set of instructions that will be executed.

```
def install():
    run('"install.exe" /S')
```

The *wapt.psproj* file

Package project file `wapt.psproj` is found in the WAPT folder.

It's the PyScripter project file for the WAPT package.

To edit a package with **`PyScripter`**, just open the file.

The *icon.png* file

The `icon.png` icon file is located in the WAPT folder.

It associates an icon to the package.

That icon will appear in the local web interface of WAPT self-service (http://127.0.0.1:8088).

Hint: The icon must be a 48px per 48px PNG file.

The *manifest.sha256* file

The `manifest.sha256` manifest file is located in the WAPT folder.

It contains the sha256 fingerprint of every file in the WAPT package.

The *control* file

The `signature` signature file is located in the WAPT folder.

It contains the signature of the `manifest.sha256` file.

On installing a package, **`wapt-get`** checks:

- that the signature of `manifest.sha256` matches the actual `manifest.sha256` file (the agent will verify the public certificates in `C:\Program Files (x86)\wapt\ssl`);
- that the sha256 fingerprint of each file is identical to the fingerprint in the `manifest.sha256` file;

Other files

Other files may be embedded in the WAPT package. For example:

- an installer beside your `setup.py` to be called in your **`setup.py`**;
- an answer file to pass on to the software installer;
- a license file;

6.14.4 Packaging simple .msi packages

Hint: Prerequisites

Pre-requisites: to build WAPT packages, *the WAPT development environment must be installed*;

Hint: From WAPT version 1.3.12 and above, a new package template creation wizard is available to help you make basic packages. For more information on package creation using package wizard, please refer to the *documentation for creating packages from the WAPT console*.

- Download the TightVNC MSI installer
  - download TightVNC MSI x64;
  - download TightVNC MSI x86;
- look up documentation relating to silent flags;
  - documentation TightVNC;

  ```
  msiexec /i tightvnc-2.7.1-setup-64bit.msi /quiet /norestart
  ```

  This command should install **TightVNC** with its default parameters. However, MSI allows you to customize the behavior of the installed software with MSI properties. The global syntax is:

  ```
  msiexec /i tightvnc-2.7.1-setup-64bit.msi /quiet /norestart PROPERTY1=value1␣
  ↪PROPERTY2=value2 PROPERTY3=value3
  ```

- launch a Windows Command Line utility **cmd.exe** as *Local Administrator*;
- instantiate a package from the pre-defined MSI template;

```
wapt-get make-template c:\download\file.msi <yourprefix>-tightvnc
```

Exemple with **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
```

Customizing the package

PyScripter opens with the new project, ready to be customized and built.

Note: Using the automatic template has the following effects:

- it creates a WAPT package folder in `C:\waptdev`;
- it copies the MSI setup file in that directory;

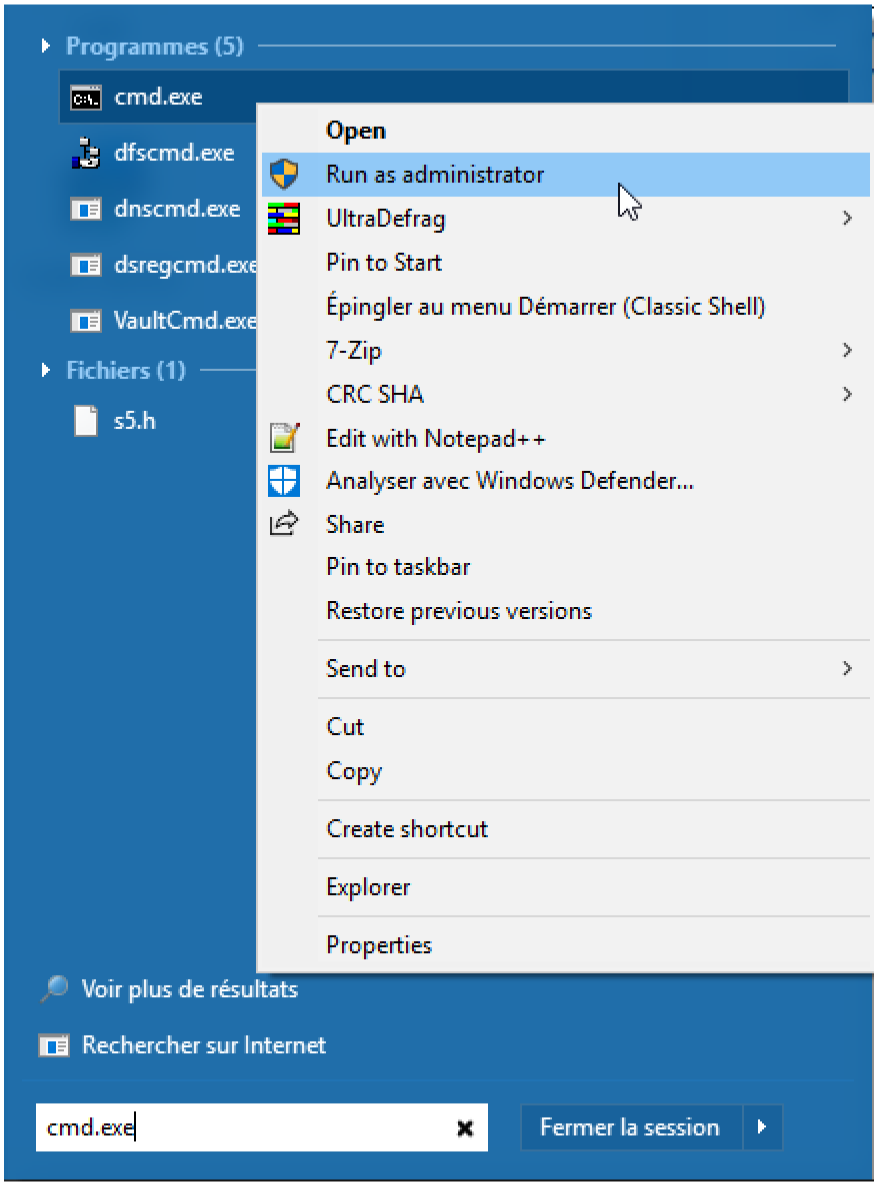
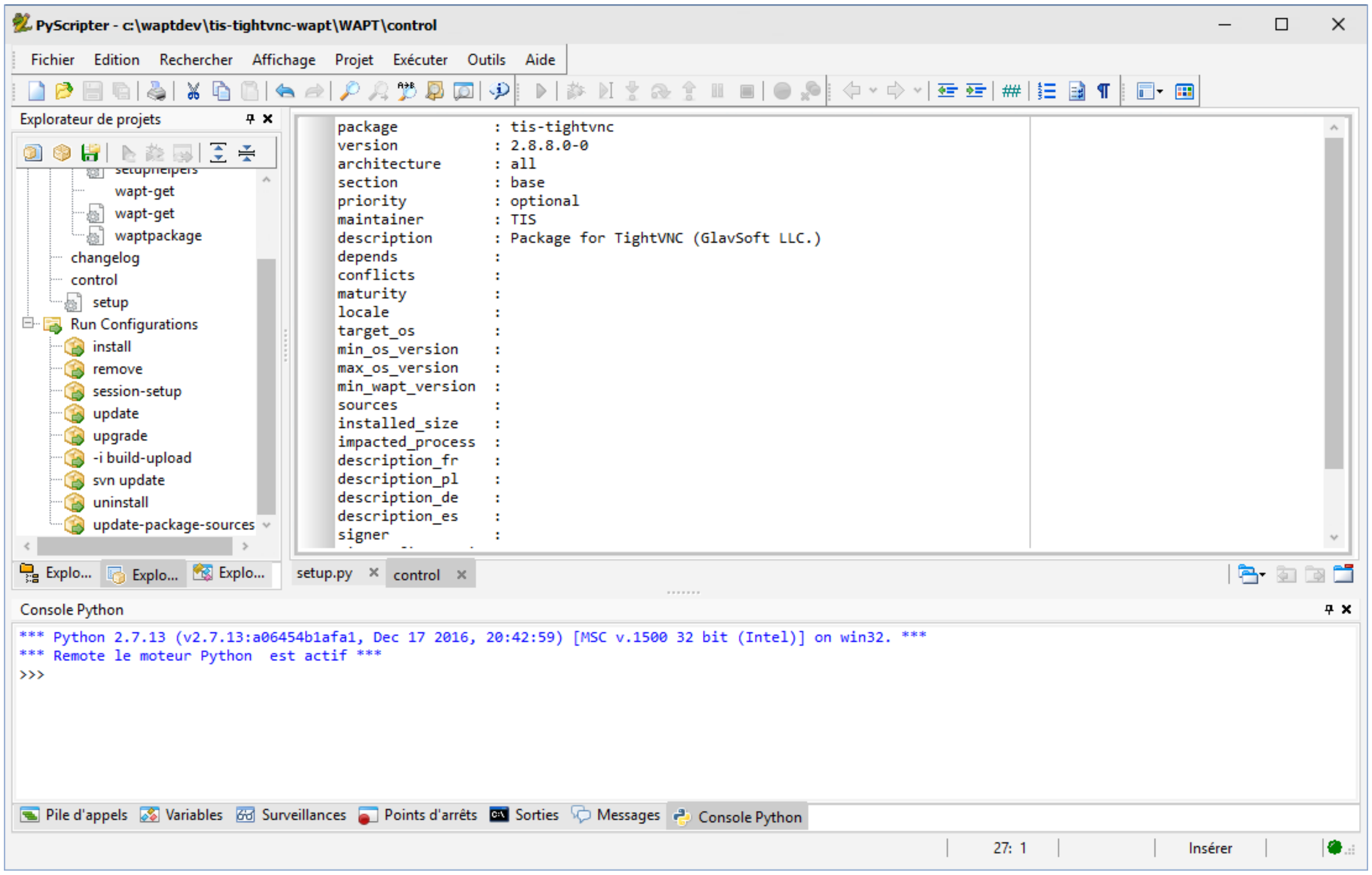


Fig. 158: Windows Command Line utility launched as Local Administrator



```
package          : tis-tightvnc
version          : 2.8.8.0-0
architecture     : all
section          : base
priority         : optional
maintainer       : TIS
description      : Package for TightVNC (GlavSoft LLC.)
depends          :
conflicts        :
maturity         :
locale           :
target_os        :
min_os_version   :
max_os_version   :
min_wapt_version :
sources          :
installed_size   :
impacted_process :
description_fr   :
description_pl   :
description_de   :
description_es   :
signer           :
```

```
*** Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote le moteur Python  est actif ***
>>>
```

Fig. 159: PyScripter with TightVNC project opened

- it creates a generic `setup.py` file;
- it creates the `control` file containing the WAPT package meta-informations;
- it creates the PyScripter project file containing pre-configured WAPT related helpers;

Hint: Then it's possible to:

- check and complete `control` file informations;
- add additional files (x86 version of the installer, configuration files, etc);
- customize setup instructions in `setup.py`;
- make changes to the `control` file as necessary (*Setting up your WAPT development and test environment*);

```
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: You may observe that a subversion (*-1*; dash+number one) has been added. It is the WAPT packaging version.

This allows to preserve the software version when iteratively improving the WAPT package.

- check the *setup.py* file;

In the `setup.py` file, you will have to check whether the *uninstall key* automatically that has been filled matches the MSI and that the silent arguments work correctly.

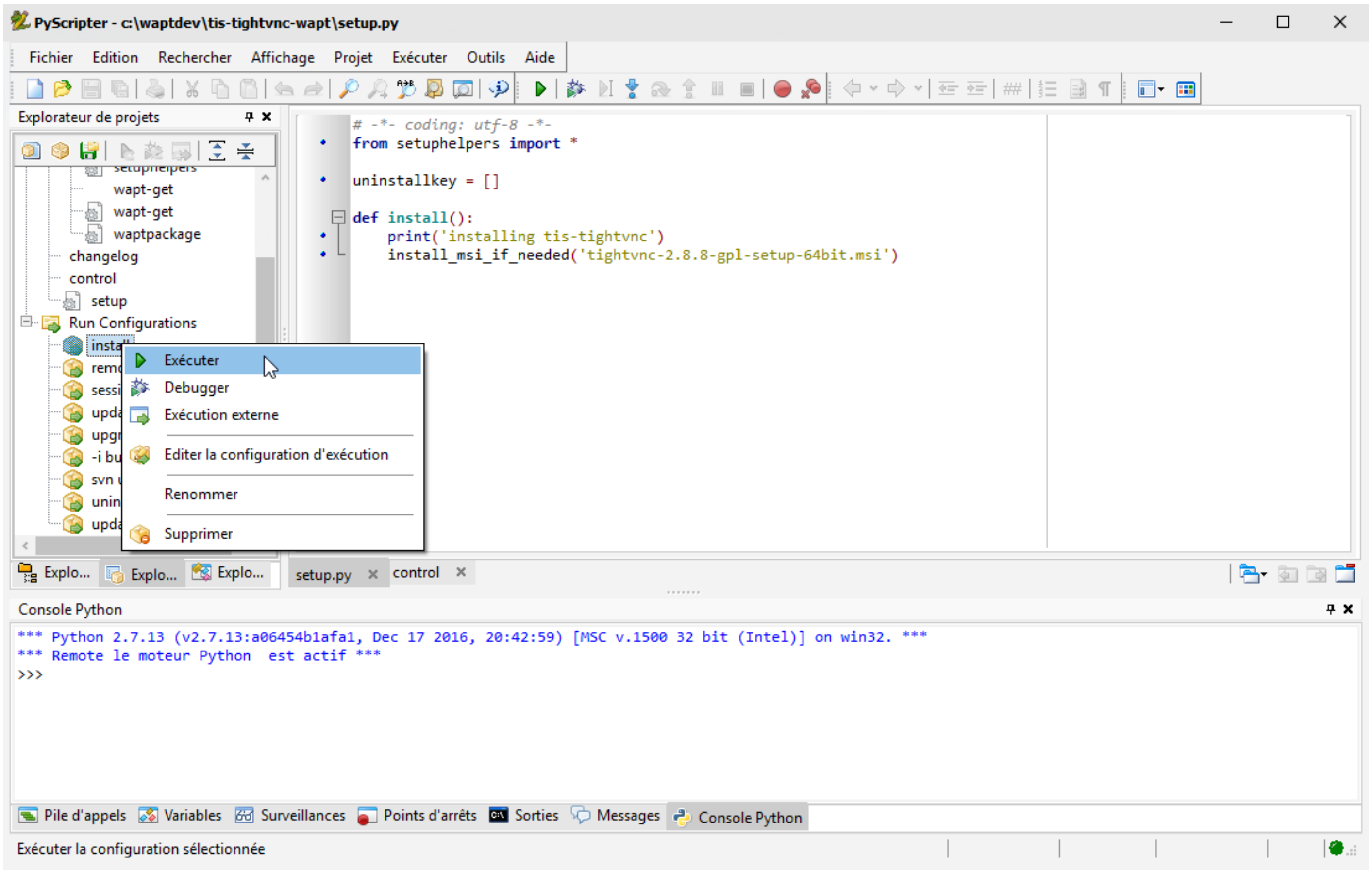
```
# -*- coding: utf-8 -*-
from setuphelpers import *

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: Since WAPT version 1.3.12, **`install_msi_if_needed`** is the function used by default when creating a package Template from a MSI.

> In this example, we will use the **`run`** function to show the evolution of the package. The function **`install_msi_if_needed`** is explained in *the documentation related to advanced packaging of MSI based WAPT packages*.

- test installing the package on you development computer;

A major advantage of **PyScripter** and WAPT is to be able to test WAPT packages locally, to improve them quickly and iteratively.



```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-tightvnc')
    install_msi_if_needed('tightvnc-2.8.8-gpl-setup-64bit.msi')
```

```
*** Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote le moteur Python  est actif ***
>>>
```

Fig. 160: Test installing the package from you development computer

Table 28: Title

Settings	Value	Description
install	Execute	Launch the software installation with its arguments from `setup.py`.
install	Debug	Launch the line by line debugger.
remove	Execute	Launch the uninstallation.
-i build-upload	Execute	Increment the version number of WAPT packaging, build the package and upload it onto the WAPT repository.

Build and upload the package

Once the project has been created, build the package without modifications from the Windows command prompt.

```
wapt-get build-upload -i c:\waptdev\tis-tightvnc-wapt
```

Note: When executing that command, here is what is really happening:

- the `manifest.sha256` file containing the list of files included in the package is created;
- it compresses the folder `C:\waptdevtis-tightvnc-wapt` with a canonical name;
- it adds the signature (requires the private key);
- it loads the WAPT package onto the WAPT repository using HTTPS;
- it recreates the `https://srvwapt.mydomain.lan/wapt/Packages` index to take into account the new or updated package;

The package is now ready to be deployed.

Example:

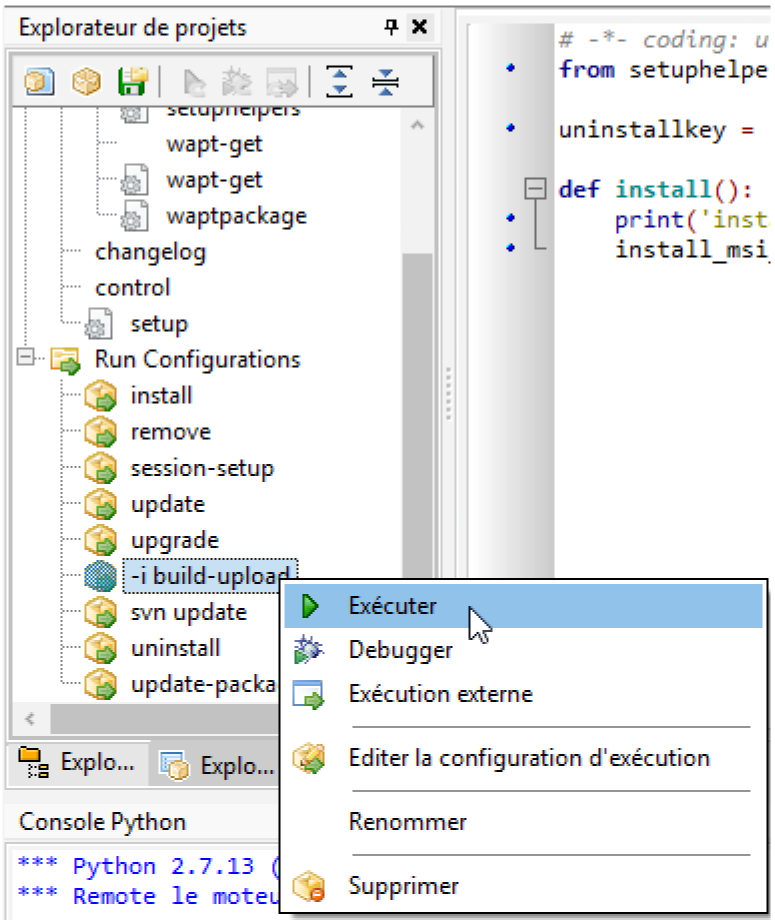
```
wapt-get -i build-upload C:\waptdev\tis-tightvnc-wapt

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==
```

(continues on next page)

(continued from previous page)

```
Uploading files...
WAPT Server user :admin
WAPT Server password:
Status: OK, tis-tightvnc_2.8.5.0-1_all.wapt uploaded, 1 packages analysed
```

It is also possible to execute **build-upload** directly from the *Run Configurations* panel in **PyScripter**:



```
# -*- coding: u
from setuphelpe

uninstallkey =

def install():
    print('inst
    install_msi
```

```
*** Python 2.7.13 (
*** Remote le moteu
```

Fig. 161: Option "-i build-upload" of PyScripter project

6.14.5 Packaging advanced .msi packages

Improving your MSI based package

The installation/ updating of the **TightVNC** package chosen as an example in the previous part of the documentation requires to close all instances of **TightVNC** for the software to upgrade.

Manual method: killalltasks

A simple method is to kill all existing processes of **TightVNC** before launching the installation of the *tis-tightvnc* package.

```
# -*- 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')
```

Elegant method: 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')
```

- the Setuphelpers library provides a function called **install_msi_if_needed** that addresses all the described problems in only one line of code;
- the function will also test whether a version of the software is already installed on the machine using the *uninstall key*;
- if the *uninstall key* is already present, the new version of the software will be installed only if the installed version is older;
- after the installation, the function will finally test that the *uninstall key* is present and its version, to ascertain that all went well;

Table 29: List of arguments available with *install_exe_if_needed*

Settings	Default value	Description
msi		name of the MSI file to execute.
min_version	None	minimal version above which the software will update.
killbefore	[]	list of programs to kill before installing the package.
accept_returncodes	[0,3010]	accepted codes other than 0 and 3010 returned by the function.
timeout	300	maximum installation wait time (in seconds).
properties	{ }	additional properties to pass as arguments to MSI setup file.
get_version	None	value passed as parameter to control the version number instead of the value returned by the *installed_softwares* function
remove_old_version	False	automatically removes an older version of a software whose *uninstall key* is identical
force	False	forces the installation of the software even though the same *uninstall key* has been found

Note: The **install_msi_if_needed** method searches for an *uninstall key* in the MSI file properties, it is not necessary to fill it manually in the `setup.py` file.

Launch the installation and watch for what's happening in the WAPT console when the software is already installed.

```
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)
```

Handling x32 and x64 architectures

To handle different processor architectures, use the function **`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.')
```

Passing additional arguments

To pass additional arguments, store them in a *dict*.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

properties = {
    'SERVER_REGISTER_AS_SERVICE':0,
    'SERVER_ADD_FIREWALL_EXCEPTION':0,
    }

def install():
  print(u'Installation en cours de TightVNC')
  if iswin64():
```

(continues on next page)

(continued from previous page)

```
    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 Packaging simple .exe package

Note: Variation compared to MSI

WAPT prefers MSI installers because most *.exe* are not standardized and silent arguments can be different from one piece of software to another.

Hint: Since WAPT version 1.3.12, a new method for quickly building WAPT packages from the WAPT console has been made available.

This part of the documentation is still actual, we recommend however that you use the WAPT console to instantiate your package templates, see *Creating a package template from the WAPT console*.

- download the exe installer from a reliable source;

  Download the installer in exe format (for example: Firefox ESR x64):

- look up documentation relating to silent flags;
  - On the Official Mozilla website;
  - other methods for finding information on silent flags:
    * WPKG packages repository;
    * Chocolatey packages repository;
    * search on the Internet with the search terms: *Firefox silent install*;

Creating a base template from the .exe file

- start up a Windows Command Line utility **cmd.exe** as *Local Administrator*;
- create a WAPT package template;

```
wapt-get make-template <chemin_exe> <nom_du_paquet>
```

  Example with Mozilla Firefox ESR:

```
wapt-get make-template "Firefox Setup 52.6.0esr.exe" "tis-firefox-esr"

Template created. You can build the WAPT package by launching
```

(continues on next page)

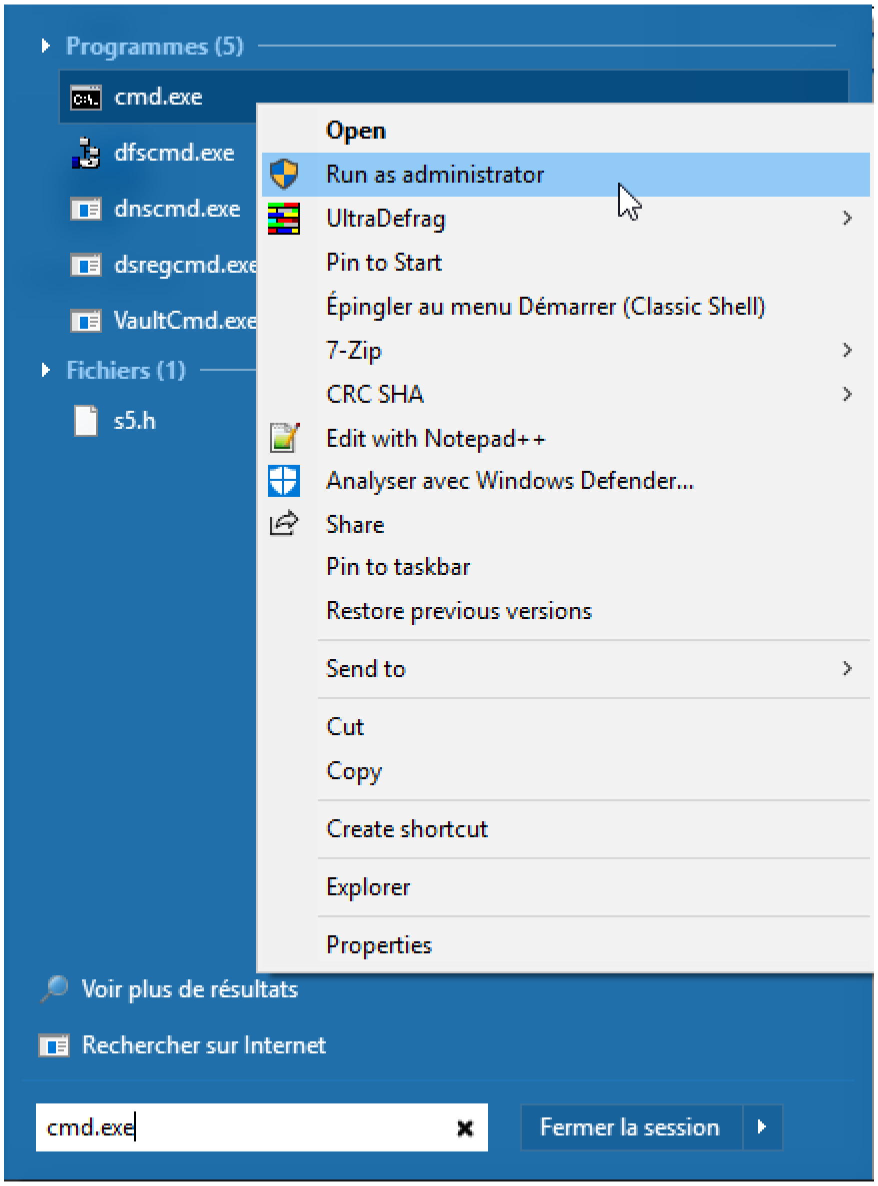
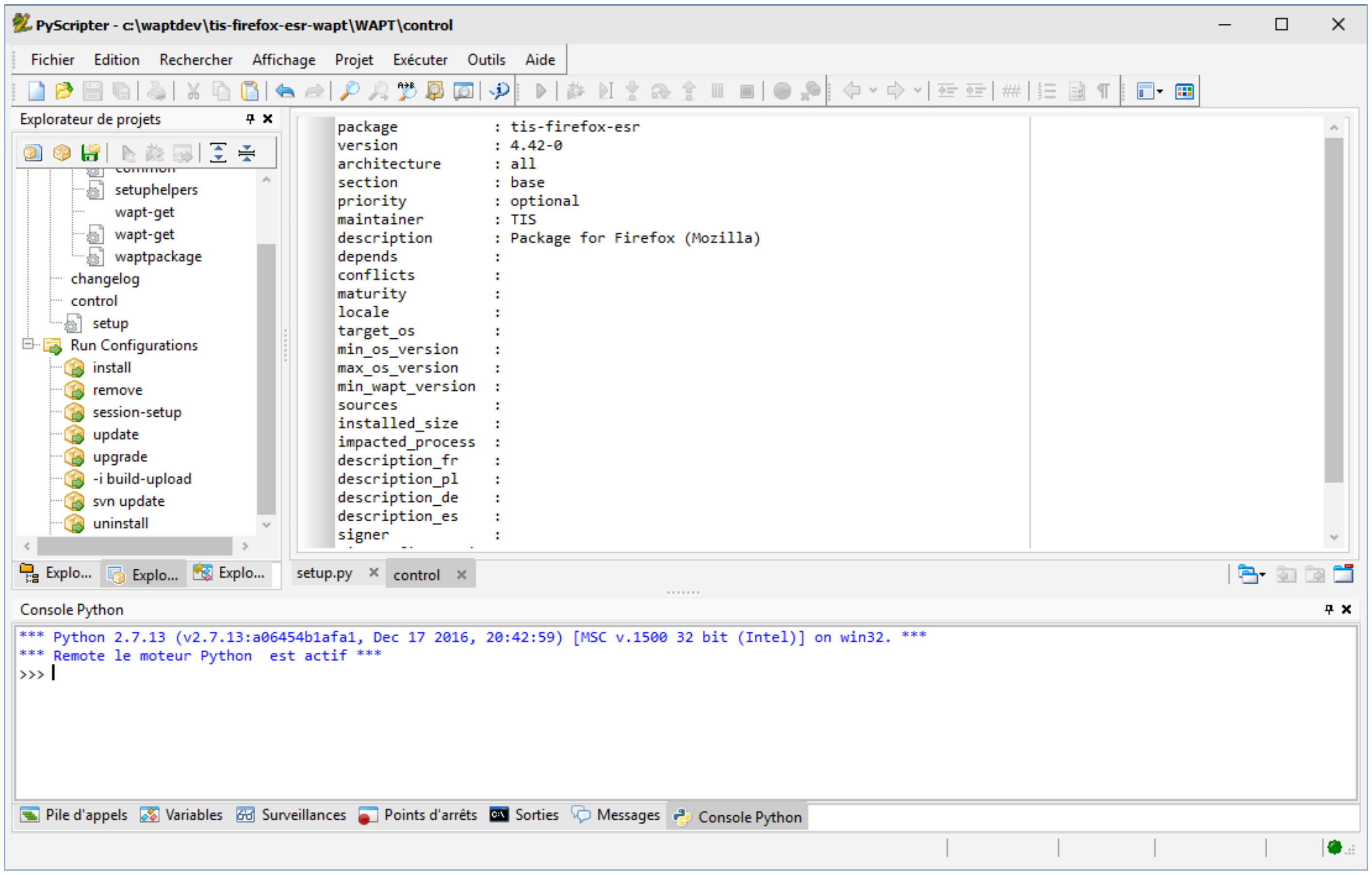


Fig. 162: Launching the Windows Command Line utility as Local Administrator

(continued from previous page)

```
  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 loads up and opens open the `.exe` package project.


PyScripter - c:\waptdev\tis-firefox-esr-wapt\WAPT\control
Fichier Edition Rechercher Affichage Projet Exécuter Outils Aide
Explorateur de projets
common
setuphelpers
wapt-get
wapt-get
waptpackage
changelog
control
setup
Run Configurations
install
remove
session-setup
update
upgrade
-i build-upload
svn update
uninstall
Explo... Explo... Explo...
setup.py control
Console Python
Pile d'appels Variables Surveillances Points d'arrêts Sorties Messages Console Python

```
package          : tis-firefox-esr
version          : 4.42-0
architecture     : all
section          : base
priority         : optional
maintainer       : TIS
description      : Package for Firefox (Mozilla)
depends          :
conflicts        :
maturity         :
locale           :
target_os        :
min_os_version   :
max_os_version   :
min_wapt_version :
sources          :
installed_size   :
impacted_process :
description_fr   :
description_pl   :
description_de   :
description_es   :
signer           :
```

```
*** Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)] on win32. ***
*** Remote le moteur Python  est actif ***
>>>
```

Fig. 163: PyScripter opening with focus on the *control* file

- check the `control` file content;

  Mozilla Firefox-ESR does not comply to industry standards and returns an erroneous version number (it appears to be the installer packaging software version number).

  Original *control* file

```
package             : tis-firefox-esr
version             : 4.42.0.0-0
architecture        : all
section             : base
priority            : optional
maintainer          : user
description         : automatic package for firefox setup 52.6.0esr
```

  Modified *control* file

```
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: It is to be noted that a sub-version *-1* has been added. It is the packaging version of WAPT package.

It allows the Package Developer to release several WAPT package versions of the same software.

- check the `setup.py` file;

  WAPT has added a generic silent */VERYSILENT* flag that may or may not work with Mozilla Firefox ESR.

  In that case, we will replace the suggested silent flag with the one that we found in the Mozilla documentation.

- make changes to the code in the `setup.py` file accordingly;

```
: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')
```

- save the package;

Managing the uninstallation

With an exe installer, the *uninstall key* is not available until the software has been installed once.

The *uninstall key* is available in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
```

or on 64bits systems

```
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall
```

- open a Windows Command Line **`cmd.exe`** prompt;
- retrieve the software *uninstall key* with `wapt-get list-registry firefox`

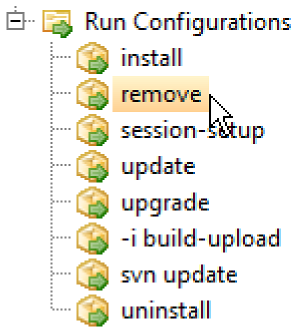
```
UninstallKey                          Software                              Version            ␣
↪Uninstallstring
------------------------------------- ------------------------------------- ------------------ -
↪-----------------------------------------------------
Mozilla Firefox 52.6.0 ESR (x64 fr)   Mozilla Firefox 52.6.0 ESR (x64 fr)   52.6.0
↪"C:\Program Files\Mozilla Firefox\uninstall\helper.exe"
```

- copy the *uninstall key* **UninstallKey**: *Mozilla Firefox 45.4.0 ESR (x64 en)*;
- make changes to the `setup.py` file with the correct *uninstall key*;

```
: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: The *UninstallKey* must be the exact same as the one listed with **`list-registry`** command. The *UninstallKey* may be a GUID such as *95160000-0052-040C-0000-0000000FF1CE*, a GUID with bracketed characters, *{95160000-0052-040C-0000-0000000FF1CE}*, or simply a character string such as *Git_is1* or *Mozilla Firefox 52.6.0 ESR (x64 fr)*.

- relaunch the package setup to set the correct *uninstall key* in the local WAPT database;
- test uninstalling the package;
- launch a *remove* from PyScripter *Run Configurations*;



After uninstallation, the software is correctly removed

We can notice the correct uninstallation by launching again the **`wapt-get list-registry`** command.

```
UninstallKey           Software           Version            Uninstallstring
---------------------- ------------------ ------------------- ----------------
---------------------- ------------------ ------------------- ----------------
```

Specific case of a non-silent uninstaller

It sometimes occurs that the software installs silently, but does not uninstall silently.

In that precise case it is necessary to override the **uninstall()** function.

Example with Mozilla Firefox:

```
# -*- 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')
```

Hint: In the **uninstall()** function, it is not possible to call for files included inside the WAPT package. To call files from the package, it is necessary to copy/ paste the files in a temporary directory during package installation.

Build and upload the package

Once the installation and the de-installation are configured and tested and the package is customized to your satisfaction, you may build and upload your new WAPT package onto your WAPT repository.

6.14.7 Packaging advanced .exe packages

Improving the functionalities of packages

Our package Mozilla Firefox ESR is functional but it requires some improvements.

- the package installs the software even if the software is already installed, it will overwrite older versions;
- the package does not check whether the software version is already installed;
- the package does not check whether the software version is already installed;

As with MSI installers, those problems are addressed using the function **install_exe_if_needed**.

Using *install_exe_if_needed*

The function is slightly the same as that used with MSI installers, with some differences:

- the function requires to pass the silent flag as an argument;
- the function requires to pass the *uninstall key* as an argument;

We make changes to the Mozilla Firefox ESR setup accordingly.

```
# -*- 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")
```

Table 30: List of arguments available with *install_exe_if_needed*

Settings	Default value	Description
`exe`		name of the `.exe` file to execute.
`silentflags`		silent parameters to pass as arguments to the installer.
`key`	None	software *uninstall key*.
`min_version`	None	minimal version above which the software will update.
`killbefore`	[ ]	list of programs to kill before installing the package.
`accept_returncodes`	[0,3010]	accepted codes other than 0 and 3010 returned by the function.
`timeout`	300	maximum installation wait time (in seconds).
`get_version`	None	value passed as parameter to control the version number instead of the value returned by the **`installed_softwares`** function.
`remove_old_version`	False	automatically removes an older version of a software whose uninstallkey is identical
`force`	False	forces the installation of the software even though the same uninstallkey has been found

As a consequence, the package will have a different behavior:

- Firefox will install only if it is not yet installed or if the installed version of Firefox is less than 45.5.0, unless the `--force` option is passed as argument when installing the package;
- on installing, the running **`firefox.exe`** processes will be killed;
- the function will add by itself the *uninstall key*, so leave the *uninstall key* argument empty;
- when finishing the installation of the package, the function will check that the *uninstall key* is present on the machine and that the version of Firefox is greater than 45.5.0; if this not the case, the package will be flagged as **ERROR**;

Managing the de-installation of applications

Special case of a non-silent uninstaller

In that particular case, a package using **install_exe_if_needed** fills in the *uninstall key*, but the *uninstall key* points to a non silent uninstaller.

We have to circumvent that problem by using a function that will remove the *uninstall key* at the end of the 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 Additional features

Customizing the user environment

One of the main advantages of WAPT is the silent installation and uninstallation of software, **even when restricted users are connected**.

The WAPT agent runs with Windows SYSTEM rights, it allow to define global settings in system context and customized settings in restricted user mode.

User settings customization relies on the `session_setup` function defined in WAPT library *Setuphelpers*.

Working principle

User mode customization operates as follows:

- the instructions are defined in the **`session_setup()`** function of the `setup.py` file;
- when the package is deployed on the machine, instructions are stored in the local database of the WAPT agent;
- when the user connects to her computer, **`session_setup`** customizations are executed in the limit of one execution per user and per WAPT package version;

Hint: The session_setup customization will occur only once per user per package; a shortcut created in that context is created only once and not at every startup. To execute a task at each startup, it is preferable to define a Windows scheduled task to be launched by a local *GPO* or by a startup script.

Example: create a personalized desktop shortcut

One of the possibilities offered by *Setuphelpers* is adding personalized shortcuts on user desktops, instead of a desktop shortcut common to all users.

For that purpose, we will use the **create_user_desktop_shortcut()** function to create shortcuts containing the username and passing a website as an argument to Firefox.

```
# -*- 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")
```

Deploying a portable software with WAPT

A good example of a WAPT package is a self-contained/ *portable* software package:

- create the folder for the software in `C:\Program Files (x86)`;
- copy the software in that folder;
- create the shortcut to the application;
- manage the uninstallation process for the application;
- close the application if it's running;

Example with ADWCleaner

- create a group package and modify the `control` file to transform it to a software package;

```
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
```

The file `C:\waptdev\tis-adwcleaner-wapt` is created.

- download and copy/ paste **`adwcleaner.exe`** binary in `C:\waptdev\tis-adwcleaner-wapt` directory;
- open and make desired changes to `C:\waptdev\tis-adwcleaner-wapt\setup.py` installation file;

```
# -*- 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 Customizing the user environment

session_setup principles

It is sometimes necessary to customize a software in user context to set specific settings or to comply to the Organization's rules and preferences:

- creating user desktop shortcut with specific arguments;
- making changes to user Windows registry keys;
- making changes to files, to browser settings of the user;

- configuring shortcuts to the Organization's set of templates for Documents, Spreadsheets or Presentations in Office Suites to encourage or insure that editorial and graphical guidelines are followed;
- setting up the user's email or instant messaging from the Organization's main user data repository (LDAP directory, database, etc);
- customizing an office suite or business software based on the Organization's main user data repository (LDAP directory, database, etc);

The **session_setup** function benefits from the power of python to achieve a high level of automation.

Principles of *session_setup*

The WAPT **session_setup** function executes at every user startup, launching that command:

```
C:\Program Files (x86)\wapt\wapt-get.exe session-setup ALL
```

Calling that function executes the **session_setup** script defined within each WAPT package installed on the computer.

The WAPT agent stores in its local database (`C:\Program Files (x86)\wapt\waptdb.sqlite`) the instruction sets of all WAPT packages.

Attention: **session_setup** is launched only **once per WAPT package version and per user**.

The WAPT agent stores in is local `%appdata%\wapt\waptsession.sqlite` database the instances of **session_setup** that have been already been played.

To launch a program at every startup, consider using a startup script located in **User Startup** (*startup(0)*) or **All Users Startup** (*startup(1)*), or via a GPO.

Output example of `wapt-get session-setup ALL`:

Note: the connected user's `session_setup` had already been launched.

```
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
```

Using *session_setup*

The :command`session_setup` scripts are located in the section *def session_setup()* of the `setup.py` file:

Example:

```
def session_setup():
   registry_setstring(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows Live\\Common",'TOUVersion
↪','16.0.0.0', type=REG_SZ)
```

Attention: With **`session_setup`**, there is no possibility to call files contained inside the WAPT package.

To call external files when uninstalling, copy and paste the needed files in an external folder during the package installation process (example: a sub-directory created in the User's own directory).

6.14.10 Using the Audit functions

Note: This feature is available in the **Enterprise** version.

Why auditing?

The audit function allows to make regular checks to desktop configurations and to centralize the results of these checks in the WAPT console. This feature allows you to ascertain that your installed base of machines matches your set of conformity rules over time.

For example you can:

- regularly check the list of local administrators on the desktops;
- ascertain over time the correct configuration of a critical software;
- regularly check the presence of the correct version of a piece of software;
- ascertain the security settings of a workstation;

The **`audit`** function benefits from the depth and the breadth of python libraries for unmatched levels of precision and finesse for your auditing needs.

Working principle of the audit function

The **`audit`** tasks are launched once after every **`upgrade`**, then regularly as defined with the `audit_schedule` value.

To manually launch an audit check, you may also use the following command:

```
C:\Program Files (x86)\wapt\wapt-get.exe audit
```

Note: By default, the audit function will not launch if the audit is note necessary.

To force the execution, you may launch the following command:

```
C:\Program Files (x86)\wapt\wapt-get.exe audit -f
```

Calling this function will launch the **audit** scripts present in each WAPT package installed on the machine.

WAPT saves in its local database `C:\Program Files (x86)\wapt\waptdb.sqlite` the audit scripts of all installed WAPT packages.

Output example of `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: In the example above, the audit script had already been executed for *tis-disable-js-chrome* and *tis-disable-ipv6* … but not for *tis-java*.

How to write the audit function

The **audit** script is defined in the package's `setup.py` with a function **def audit()**:

Example:

```
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"
```

Hint: This example ascertains that USB storage is not allowed on the workstation.

The audit function returns one of these 3 values:

- **OK**;
- **WARNING**;
- **ERROR**;

Attention: With the **audit** function, it is not possible to use files that are contained in the WAPT packages.

To use files embedded in the WAPT package that will be used for an audit, you must first copy the file(s) in a temporary folder during package installation.

Planning an audit

The **audit** tasks are launched once after every **upgrade**, then regularly as defined with the `audit_schedule` value.

The value is contained in the control file of your package.

By default, if **audit_schedule** is empty, the audit task will need to be launched manually or from teh WAPT console.

Otherwise, the periodicity may be indicated in several ways:

- an integer (in minutes);
- an integer followed by a letter (m = minutes, h = hours , d = days , w = weeks);

Default behavior of the audit function

By default, the only audit function checks the presence of UninstallKey for its WAPT package.

This way, WAPT ascertains that the software is still present on the host, according to the host configuration.

6.14.11 Simple examples of commonly used functions

Presentation of several functions implemented in *Setuphelpers* and frequently used to develop WAPT packages.

Testing and manipulating folders and files

Creating a path recursively

Command **makepath** ...

```
makepath(programfiles,'Mozilla','Firefox')
```

... makes the path variable for `C:\Program Files (x86)\Mozilla\Firefox`.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=makepath#setuphelpers.makepath

Creating and destroying directories

Command **mkdirs** . . .

```
mkdirs('C:\\test')
```

. . . creates the directory `C:\test`.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=mkdirs#setuphelpers.mkdirs

Command **remove_tree** . . .

```
remove_tree(r'C:\tmp\target')
```

. . . destroys the directory `C:\tmp\target`.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_tree#setuphelpers.remove_tree

Checking if a path is a file or a folder

Command **isdir** . . .

```
isdir(makepath(programfiles32,'software')):
    print('The directory exists')
```

. . . checks if `C:\Program Files (x86)\software` is a directory.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=isdir#setuphelpers.isdir

Command **isfile** . . .

```
isfile(makepath(programfiles32,'software','file')):
    print('file exist')
```

. . . checks if `C:\Program Files (x86)\software\file` is a file.

Hint: For more informations or to learn more on arguments on that function,* please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=isfile#setuphelpers.isfile

Check if a directory is empty

Command **dir_is_empty** . . .

```
dir_is_empty(makepath(programfiles32,'software')):
    print('dir is empty')
```

. . . checks that directory `C:\Program Files (x86)\software` is empty.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=dir_is_empty#setuphelpers.dir_is_empty

Copying a file

Command **filecopyto** . . .

```
filecopyto('file.txt',makepath(programfiles32,'software'))
```

. . . copies `file.txt` into the `C:\Program Files (x86)\software` directory.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=filecopyto#setuphelpers.filecopyto

Copying a directory

Command **copytree2** . . .

```
copytree2('sources','C:\\projet')
```

. . . copies the `sources` folder into the `C:\projet` directory.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=copytree2#setuphelpers.copytree2

Retrieving the version of a file

Command **get_file_properties** …

```
get_file_properties(makepath(programfiles32,'InfraRecorder','infrarecorder.exe'))['ProductVersion
↪']
```

… shows package properties.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_file_properties#setuphelpers.get_file_properties

Manipulating registry keys

Checking the existence of a registry key

Command **registry_readstring** …

```
if registry_readstring(HKEY_LOCAL_MACHINE, "SOFTWARE\\Google\\Update\\Clients\\{8A69D345-D564-
↪463c-AFF1-A69D9E530F96}", 'pv'):
    print('key exist')
```

… checks if registry key *{8A69D345-D564-463c-AFF1-A69D9E530F96}* exists in registry path `SOFTWARE\Google\Update\Clients` of *HKEY_LOCAL_MACHINE*.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_readstring#setuphelpers.registry_readstring

Showing the value of a registry key

Command **registry_readstring** …

```
print(registry_readstring(HKEY_LOCAL_MACHINE, r'SOFTWARE\Google\Update\Clients\{8A69D345-D564-
↪463c-AFF1-A69D9E530F96}', 'pv'))
```

… reads the value *{8A69D345-D564-463c-AFF1-A69D9E530F96}* stored in the registry path `SOFTWARE\Google\Update\Clients` of *HKEY_LOCAL_MACHINE*.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_readstring#setuphelpers.registry_readstring

Modifying the value of a registry key

Command **`registry_setstring`** …

```
registry_setstring(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows Live\\Common",'TOUVersion',
↪'16.0.0.0', type=REG_SZ)
```

… modifies the value of the registry key *TOUVersion* stored in the registry path `SOFTWARE\Microsoft\Windows Live` of *HKEY_CURRENT_USER*.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=registry_setstring#setuphelpers.registry_setstring

Creating and destroying shortcuts

create_desktop_shortcut/ remove_desktop_shortcut

Command **`create_desktop_shortcut`** …

```
create_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

… creates the shortcut *WAPT Console Management* into `C:\Users\Public` directory pointing to `C:\Program Files (x86)\wapt\waptconsole.exe`; the shortcut is available for all users.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=create_desktop_shortcut#setuphelpers.create_desktop_shortcut

Command **`remove_desktop_shortcut`** …

```
remove_desktop_shortcut('WAPT Console Management')
```

… deletes the *WAPT Console Management* shortcut from the folder `C:\Users\Public`; the shortcut is deleted for all users.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_desktop_shortcut#setuphelpers.remove_desktop_shortcut

create_user_desktop_shortcut/ remove_user_desktop_shortcut

Hint: These functions are used in session_setup context

Command **`create_user_desktop_shortcut`** …

```
create_user_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

… creates the shortcut *WAPT Console Management* on user desktop pointing to `C:\Program Files (x86)\wapt\waptconsole.exe`.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=create_user_desktop_shortcut#setuphelpers.create_user_desktop_shortcut

Removing a shortcut for the current user

Command **`remove_user_desktop_shortcut`** …

```
remove_user_desktop_shortcut('WAPT Console Management')
```

… deletes the *WAPT Console Management* shortcut from the logged in user's desktop.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=remove_user_desktop_shortcut#setuphelpers.remove_user_desktop_shortcut

Windows environment/ Software/ Services

windows_version

Command **`windows_version`** …

```
windows_version()<Version('6.2.0'):
```

… checks that the Windows version is stricly inferior to *6.2.0*.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=windows_version#setuphelpers.windows_version

Visit also Microsoft Windows version number.

iswin64

Command **iswin64** …

```
if iswin64():
    print('Pc x64')
else:
    print('Pc not x64')
```

… checks that the system architecture is 64bits.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=iswin64#setuphelpers.iswin64

programfiles/ programfiles32/ programfiles64

Return different *ProgramFiles* locations

Command **programfiles64** …

```
print(programfiles64())
```

… returns native Program Files directory, eg. `C:\Program Files (x86)` on either win64 or win32 architecture.

```
print(programfiles())
```

… returns path of the 32bit Program Files directory, eg. `Programs Files (x86)` on win64 architecture, and `Programs Files` on win32 architecture.

```
print(programfiles32())
```

user_appdata/ user_local_appdata

Hint: These functions are used with **session_setup**

Command **user_appdata** …

```
print(user_appdata())
```

… returns roaming *AppData* profile path of logged on user (`C:\Users\%username%\AppData\Roaming`).

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=user_appdata#setuphelpers.user_appdata

Command **user_local_appdata** . . .

```
print(user_local_appdata())
```

. . . returns the local *AppData* profile path of the logged on user (`C:\Users\%username%\AppData\Local`).

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=user_local_appdata#setuphelpers.user_local_appdata

disable_file_system_redirection

Command **disable_file_system_redirection** . . .

```
with disable_file_system_redirection():
    filecopyto('file.txt',system32())
```

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=disable_file_system_redirection#setuphelpers.disable_file_system_redirection

Disable wow3264 redirection in the current context

get_computername/ get_current_user

Command **get_current_user** . . .

```
print(get_current_user())
```

. . . shows the currently logged on username

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_current_user#setuphelpers.get_current_user

Command **get_computername** . . .

```
print(get_computername())
```

. . . shows the name of the computer

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=get_computername#setuphelpers.get_computername

Command **get_domain_fromregistry** . . .

```
get_domain_fromregistry()
```

. . . returns the FQDN of the computer.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

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

Command **installed_softwares** . . .

```
installed_softwares('winscp')
```

. . . returns the list of installed software on the computer from registry in an array.

```
[{'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}]
```

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=installed_softwares#setuphelpers.installed_softwares

uninstalll_cmd

Command **uninstall_cmd** . . .

```
uninstall_cmd('winscp3_is1')
```

. . . returns the silent uninstall command.

```
"C:\Program Files\WinSCP\unins000.exe" /SILENT
```

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=uninstall_cmd#setuphelpers.uninstall_cmd

uninstalling software

```
for soft in installed_softwares('winscp3'):
    if Version(soft['version']) < Version('5.0.2'):
        run(WAPT.uninstall_cmd(soft['key']))
```

- for each item of the list return by *installed_softwares* containing keyword *winscp*;
- if the version is lower than 5.0.2;
- then uninstall using the *uninstall_cmd* and specifying the corresponding *uninstallkey*;

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://dev.tranquil.it/sphinxdocs/source/setuphelpers.html?highlight=uninstall_cmd#setuphelpers.uninstall_cmd

killalltasks

Command **`killalltasks`** …

```
killalltasks('firefox')
```

… kills the process named *Firefox*.

Hint: For more informations or to learn more on arguments on that function, please visit official Setuphelpers reference documentation:

https://www.wapt.fr/en/api-doc-1.5/source/setuphelpers.html?highlight=killalltasks#setuphelpers.killalltasks

Using control file fields

```
def setup():
    print(control['version'])
```

… shows the *version* value from the `control` file.

```
def setup():
    print(control['version'].split('-',1)[0])
```

… shows the software version number without the WAPT version number from the `control` file.

Calling WAPT actions in a WAPT package

Installing a package

Command **install** ...

```
WAPT.install('tis-scratch')
```

... installs *tis-scratch* on the computer.

Removing a package

Command **remove** ...

```
WAPT.remove('tis-scratch')
```

... uninstalls *tis-scratch* from the computer.

Forgeting a package

Command **forget_packages** ...

```
WAPT.forget_packages('tis-scratch')
```

... informs WAPT to forget *tis-scratch* on the selected computer.

Hint: If the desired result is to remove *tis-scratch*, you should either reinstall the package (`wapt-get install "tis-scratch"`) then remove it (`wapt-get remove "tis-scratch"`), either removing it manually from the Control Panel menu *Add/ Remove Programs*.

6.14.12 Packaging simple Linux packages

Before starting, we assume several conditions:

- you have a graphical interface on your Linux system;
- you have installed the **vscode** package from the Tranquil IT repository;
- your user is named *linuxuser* and is a member of the *sudoers* group;

create a base template from you linux computer

- start up a Command Line utility;
- as *linuxuser*, create a WAPT package template;

```
wapt-get make-template <template_name>
```

Warning: Do not launch this command as root or with sudo.

When you create a template, there will be several files in the folder `.vscode` inside your package folder:

- settings.json;
- launch.json;

Example with 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
```

Hint: All packages are stored in linuxuser's home.

VSCode loads up and opens package project.

- check the `control` file content;

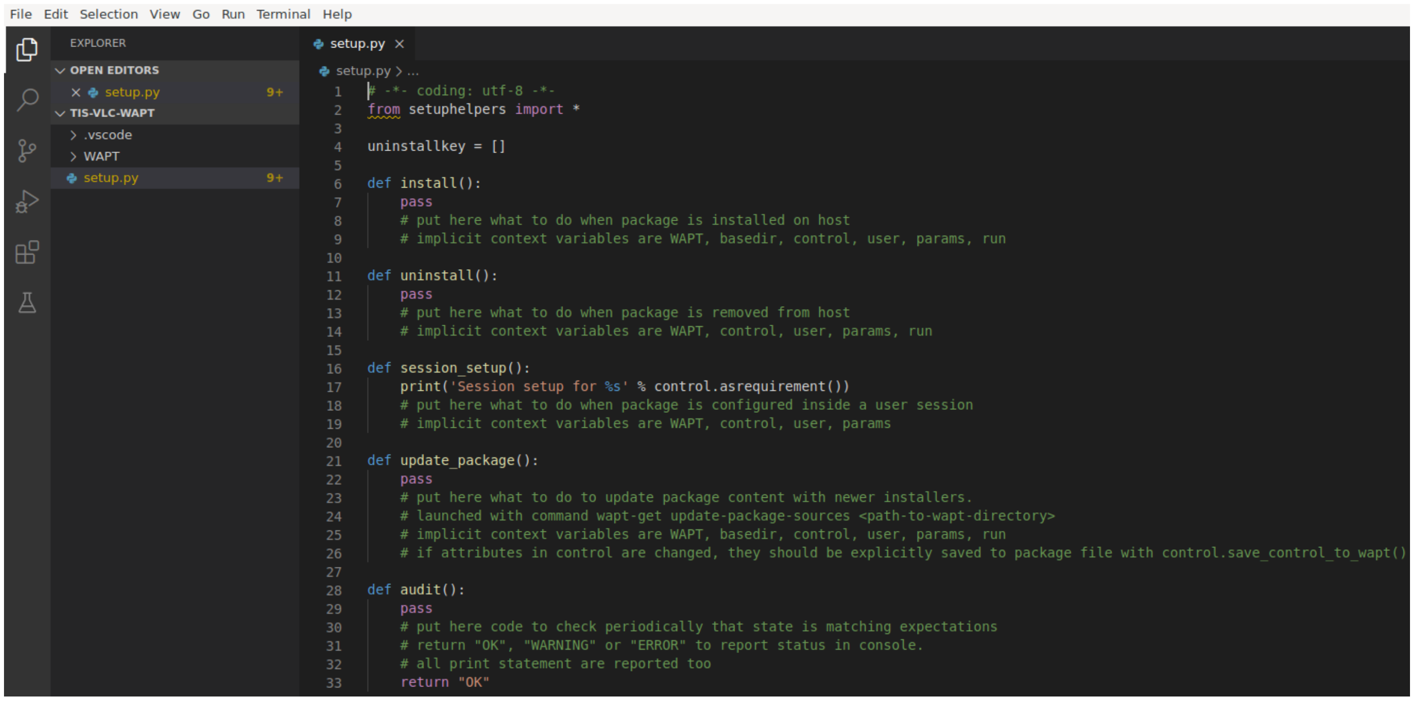
You have to give a **description** to you package, give the **os_target** and the **version** of you package.

Hint: **os_target** for unix is *linux*

Warning: *version* in you `control` file must start at 0, not the version number of the software, we don't know precisely from apt/yum repo which version will be.

Original *control* file

```
package             : tis-vlc
version             : 0-0
architecture        : all
section             : base
priority            : optional
maintainer          : user
description         : automatic package for vlc
```



```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    pass
    # put here what to do when package is installed on host
    # implicit context variables are WAPT, basedir, control, user, params, run

def uninstall():
    pass
    # put here what to do when package is removed from host
    # implicit context variables are WAPT, control, user, params, run

def session_setup():
    print('Session setup for %s' % control.asrequirement())
    # put here what to do when package is configured inside a user session
    # implicit context variables are WAPT, control, user, params

def update_package():
    pass
    # put here what to do to update package content with newer installers.
    # launched with command wapt-get update-package-sources <path-to-wapt-directory>
    # implicit context variables are WAPT, basedir, control, user, params, run
    # if attributes in control are changed, they should be explicitly saved to package file with control.save_control_to_wapt()

def audit():
    pass
    # put here code to check periodically that state is matching expectations
    # return "OK", "WARNING" or "ERROR" to report status in console.
    # all print statement are reported too
    return "OK"
```

Fig. 164: VSCode opening with focus on the *setup* file

Modified *control* file

```
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: It is to be noted that a sub-version *-1* has been added. It is the packaging version of WAPT package.

It allows the Package Developer to release several WAPT package versions of the same software.

- make changes to the code in the `setup.py` file accordingly;

```
:emphasize-lines: 8
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    apt_install('vlc')
```

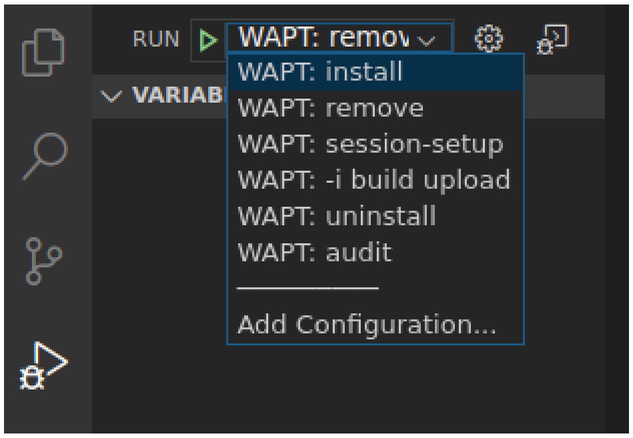
- save the package;

Managing the uninstallation

- make changes to the `setup.py` file with an uninstall ;

```
def uninstall():
apt_remove('vlc')
```

- launch a *remove* from VSCode *Run Configurations*;



- check that the software has been correctly removed

```
dpkg -l | grep vlc
```

Hint: In the **`uninstall()`** function, it is not possible to call for files included inside the WAPT package. To call files from the package, it is necessary to copy/ paste the files in a temporary directory during package installation.

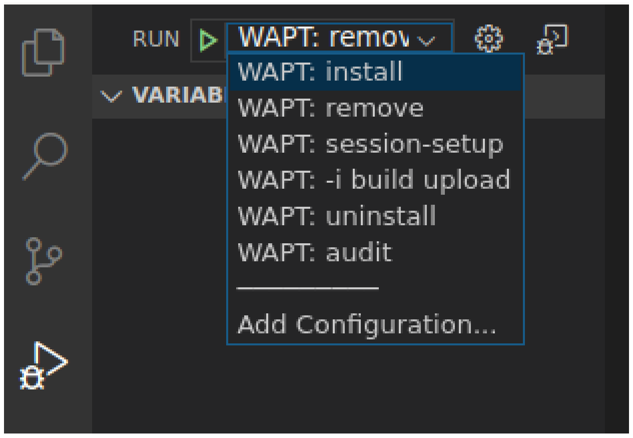
Managing the session-setup

- make changes to the `setup.py` file with an session-setup ;

  In this example, you'll need a `vlcrc` file in your package to copy in home user. `ensure_dir` function and `filecopyto` are from **setuphelpers**, the first one will test if the path exists, the second one will copy your file from the WAPT package to its destination.

```
def session-setup():
  vlcdir = os.path.join(os.environ['HOME'], '.config', 'vlc')
  ensure_dir(vlcdir)
  filecopyto('vlcrc',vlcdir)
```

- launch a *session-setup* from VSCode *Run Configurations*;



Build and upload the package

Once the installation and the de-installation are configured and tested and the package is customized to your satisfaction, you may build and upload your new WAPT package onto your WAPT repository.

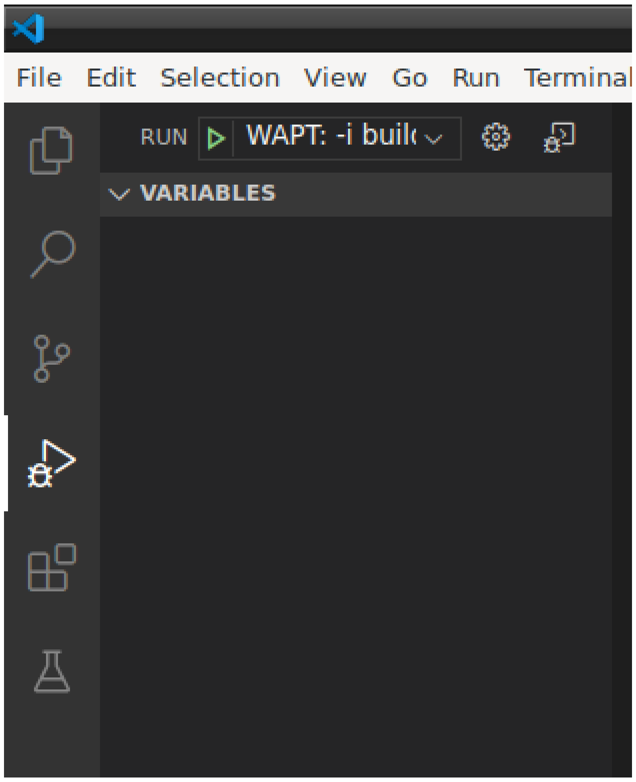
If you have built packages on a different machine (ex: Windows for building your Windows WAPT packages), you have to copy your *.pem* and *.crt* keys on your Linux machine with **WinSCP** or equivalent. Usually, this certificate bundle will be located in `C:\private` on your Windows computer. Then, provide the path to the certificates in `/opt/wapt/wapt-get.ini`.

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

- provide the path to your certificate;

```
personnal_certificate_path=/opt/wapt/private/mykey.crt
```

- then launch a **build-upload** from VSCode *Run Configurations*;



- provide the password to your private key then admin/password of your *waptconsole*;

Your package is now uploaded and available in your private repository on your WAPT server.

6.14.13 Updating automatically a software package

Note: This part of the documentation is for advanced users of WAPT.

Why is it useful to do that?

update_package functions are very practical, they allow to gain a lot of time when it is time to update a WAPT package with the most recent version of a piece of software.

Working principle

The *update_package* function will:

- fetch online the latest version of the software;
- download the latest version of the software binaries;
- remove old versions of the software binaries;
- update the version number of the software in the `control` file;

If you base your *install* function on the version number inside the `control` file, then you do not even need to modify your `setup.py`.

You just have to do your usual Quality Assurance tests before you **build-upload** your new package.

Example

Here is the *update_package* script for **firefox-esr** as an example:

```
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()
```

You may launch the *update_package* script by pressing `F9` in **PyScripter**.

You will find many inspiring examples of *update_package* scripts in packages hosted in the Tranquil IT store.

6.14.14 Packaging Windows Update .msu packages

Hint: Pre-requisites: to build WAPT packages, *the WAPT development environment must be installed*;

Between *Patch Tuesday* releases, Microsoft may release additional KBs or critical updates that will need to be pushed to hosts quickly.

For that purpose, WAPT provides a package template for *.msu files.

In that example, we use the KB4522355 downloaded from Microsoft Catalog website.

- download the KB package from Microsoft Catalog website:
  - download KB4522355 MSU;

Creating a MSU package template from the WAPT console

- create a WAPT package Template from the downloaded MSU file;

  In the WAPT console, click on *Tools* → *Package Wizard*;

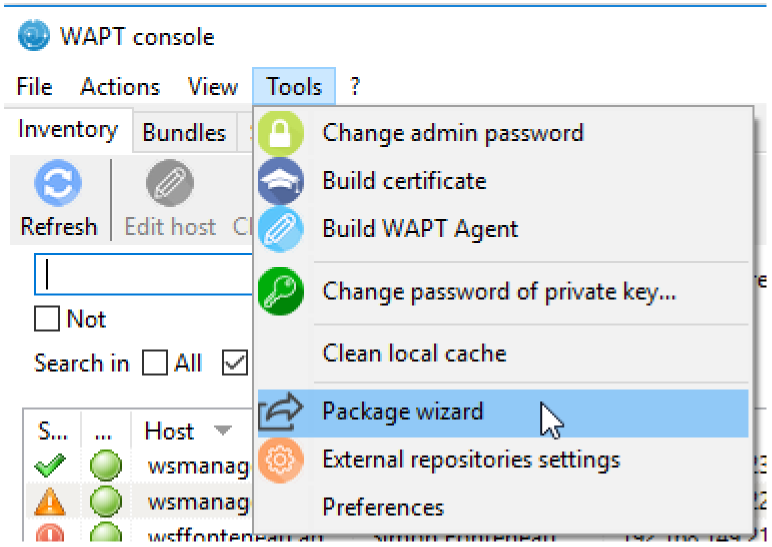


Fig. 165: Pyscripter - WAPT console window for creating a package template

- select the downloaded MSU package and fill in the required fields;
- click on *Make and edit* . . . . (recommended) to launch package customization;
- WAPT package IDE is launched using the source code from the pre-defined MSU template.
- as usual with WAPT packages, test - build - sign - upload - affect to hosts and it is done!!
- if the KB becomes bundled with the following *Patch Tuesday*, you can select the hosts onto which the package has been applied and forget the KB package on the hosts;

Creating a MSU package template from command line

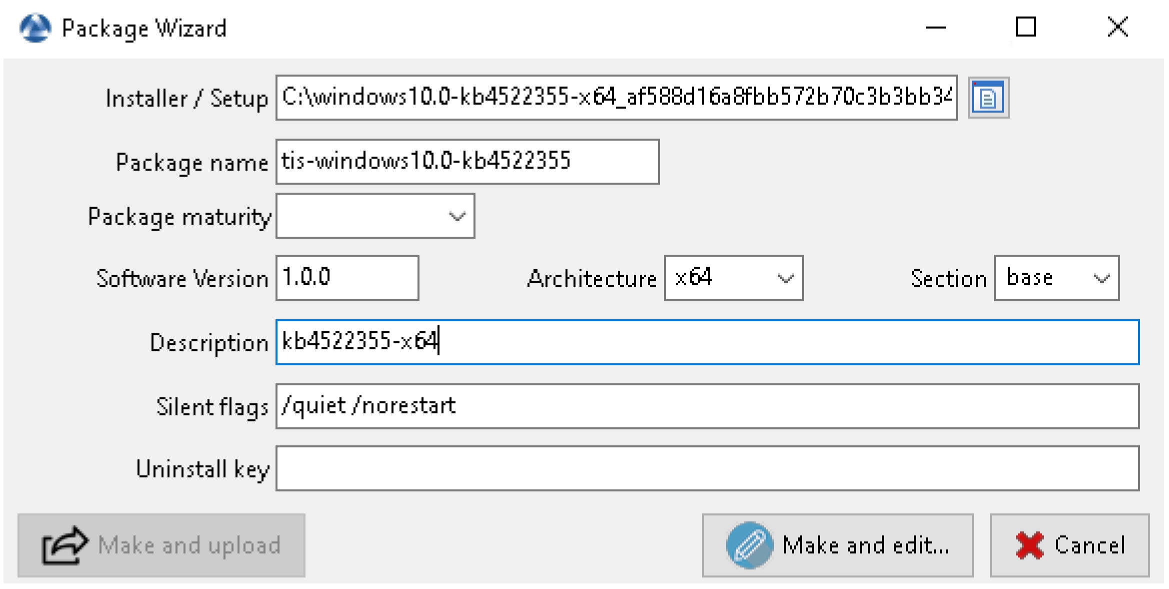
- launch a Windows Command Line utility **cmd.exe** as *Local Administrator*;
- instantiate a package from the pre-defined MSU template;

```
wapt-get make-template c:\download\file.msu <yourprefix>-kb4522355
```

- output example with **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
```

(continues on next page)



Installer / Setup: C:\windows10.0-kb4522355-x64_af588d16a8fbb572b70c3b3bb34
Package name: tis-windows10.0-kb4522355
Package maturity:
Software Version: 1.0.0
Architecture: x64
Section: base
Description: kb4522355-x64
Silent flags: /quiet /norestart
Uninstall key:

Fig. 166: Informations required for creating the MSU package

(continued from previous page)

```
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
```

- WAPT package IDE is launched, here is an example source code from the pre-defined MSU template:

```
# -*- 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',
```

(continues on next page)

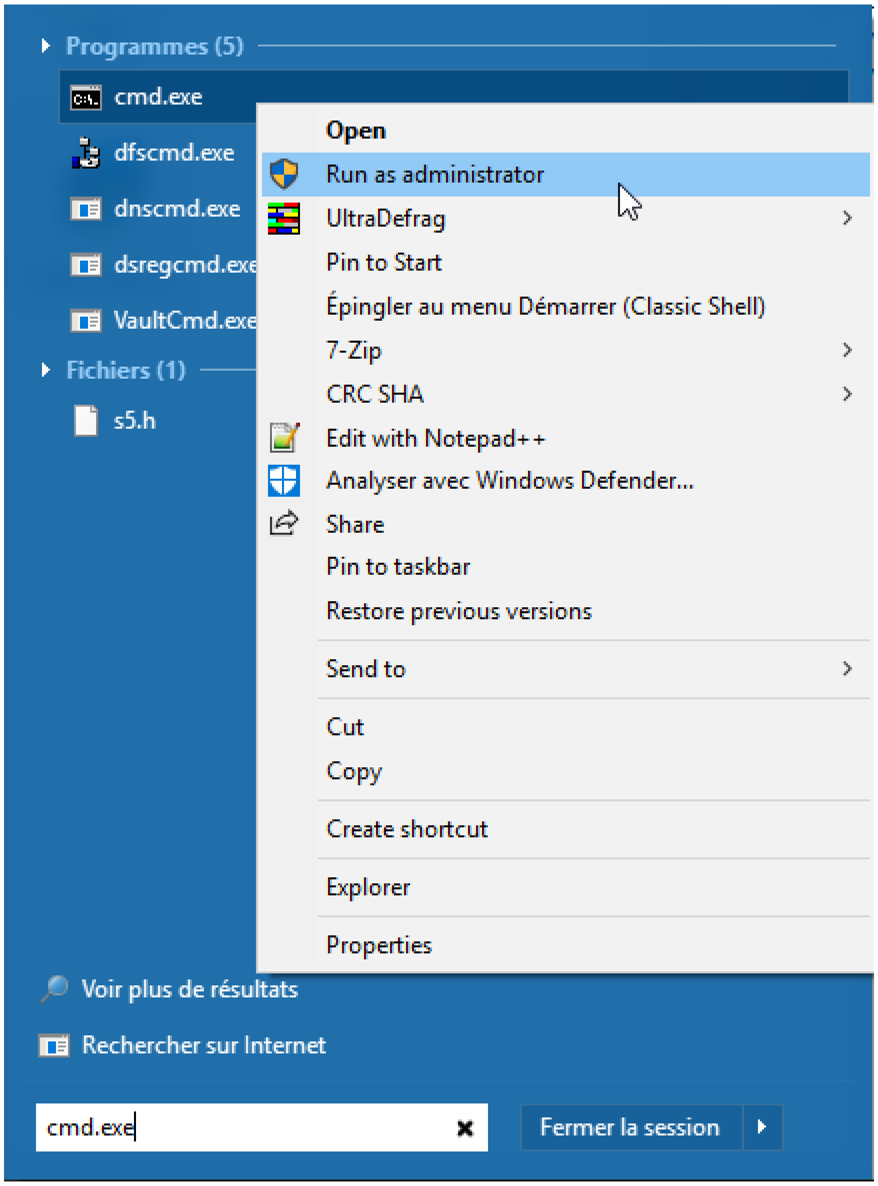


Fig. 167: Windows Command Line utility launched as Local Administrator

(continued from previous page)

```
            ]
        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 Encrypting some sensitive data contained in a WAPT package

Note: This part of the documentation is not recommended for users starting with WAPT.

What is the purpose for doing that?

With WAPT, the integrity of the package is ensured. A package whose content has been modified without being re-signed will systematically be refused by the WAPT client.

On the other hand, the content of a WAPT package is not encrypted and will be readable by everyone. This technical model of transparency brings nevertheless many benefits.

This can be annoying in the case of a package that contains a password, a license key, or any sensitive or confidential data.

Fortunately, **we have a solution**!

Working principle of encrypting portions of the content of a WAPT package

When a WAPT agent registers with the WAPT server, it generates a private key/ public certificate pair in `C:\Program Files (x86)\wapt\private`.

- The certificate is sent to the server with the inventory when the WAPT client is first registered;
- The private key is kept by the agent and is only readable locally by *Local Administrators*;

We will therefore encrypt the sensitive data contained inside the package with the certificate belonging to the machine.

During installation, the WAPT agent will be able to decrypt the sensitive data using its private key.

With this mode of operation, the WAPT server and secondary repositories have no knowledge of the sensitive data.

Practical case

You will find here an example of a WAPT package where we encrypt a string of text in an **`update_package`** function and then decrypt this text in the **`install`** function.

In this example, the **`update_package`** function allows us to browse the WAPT server database to retrieve the certificate from each machine and then encrypt the sensitive text with it.

The encrypted text for each machine is then stored in a `encrypt-txt.json` file at the root of the package.

During the installation of the package, the WAPT agent will take the encrypted text and decrypt it with its private key.

```
# -*- 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: The python output (log install of the package) is readable by the users on the machine, so **you should not display the decrypted text with a :command:`print` during installation**.

6.14.16 Working with non standard return codes

Return codes are used to feed back information on whether a software has installed correctly.

In Windows, the standard successfull return code is [0].

If you know that your WAPT packages installs correctly, yet you still get a return code other than [0], then you can explicitly tell WAPT to ignore the error code by using the parameter `accept_returncodes`.

You can find out how to use the `accept_returncodes` parameter by exploring this package 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!')
```

Hint: The full list of Windows Installer Error Messages can be found by visiting this page.

6.14.17 Using another IDE for WAPT

Introduction

If you are used to work with another *IDE*, you can be relieved now as WAPT supports other editors.

Some code editors are natively supported:

- PyScripter;
- VSCode;
- VSCodium;

Other editors can be selected and will be launched when you create a new template for a WAPT package from WAPT Console.

Configuring WAPT to use another IDE

Note: Using a supported IDE will launch the WAPT package project with a valid debug configuration.

Using Microsoft Windows

To configure another editor for WAPT, you must modify the `editor_for_packages` attribute in the `[global]` section of your WAPT console's `%LOCALAPPDATA%\waptconsole\waptconsole.ini` configuration file.

Possible values are:

Editor name	editor_for_packages value
PyScripter	None
Microsoft Visual Studio Code	**vscode** or **code**
Microsoft Visual Studio Codium	**vscodium** or **codium**

Example config in `waptconsole.ini`:

```
[global]
...
editor_for_packages=vscode
```

Using Linux / macOS

To configure another editor for WAPT, you must modify the `editor_for_packages` attribute in the `[global]` section of your WAPT agent configuration file: `/opt/wapt/wapt-get.ini`.

By default, if the `editor_for_packages` attribute is empty, WAPT will try to launch (in that order):

- **vscodium**;
- **vscode**;
- **nano**;
- **vim**;

- **vi**;

Possible values are:

Editor name	editor_for_packages value
Microsoft Visual Studio Code	**vscode** or **code**
Microsoft Visual Studio Codium	**vscodium** or **codium**
Nano	**nano**
Vim	**vim**
Vi	**vi**

```
[global]
...
editor_for_packages=vim
```

Configuring WAPT to use a custom editor

Using Microsoft Windows

Custom editors can be used, for example **Notepad++** or **PyCharm**.

Custom editors example:

Editor name	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}
```

Using Linux/Apple macOS

Custom editors can be used, for example **PyCharm**.

Custom editors example:

Editor name	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}
```

Custom arguments

Arguments can be passed in the `editor_for_packages` command:

Argument	Description
`{setup_filename}`	Launches custom editor and edit WAPT package setup.py file
`{control_filename}`	Launches custom editor and edit WAPT package control file
`{wapt_sources_dir}`	Launches custom editor and opens WAPT package folder
`{wapt_base_dir}`	Launches custom editor and opens WAPT install folder

6.14.18 Videos showing WAPT in action

Creating and deploying an msi package with WAPT

Creating, configuring and deploying an exe package with WAPT

6.15 Using the WAPT server APIs

Note: This documentation does not describe all the available APIs (Application Protocol Interfaces), it will however concentrate on the most useful ones.

All available URLs may be found in `/opt/wapt/waptserver/server.py`.

URLs are formed by calling the proper command from the WAPT Server, ex: `https://srvwapt/`*`command_path`*.

Hint: This documentation contains examples using Python code or curl.

6.15.1 API V1

/api/v1/hosts

- get registration data of one or several hosts:

```
# 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:
```

(continues on next page)

(continued from previous page)

```
#     result (dict): {'records':[],'files':[]}
#     query:
#       uuid=<uuid>
#     or
#       filter=<csvlist of fields>:regular expression
# """
```

- list all hosts. Available parameters are;
  - *reachable*
  - *computer_fqdn* ==> computer_name
  - *connected_ips*
  - *mac_addresses*

  This example shows a request with parameters:

```
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))
```

  This example is a global request:

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/hosts
```

This one just show request with reachable status, the computer name, its connected ips and its mac addresses. Display limit is 10000

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/hosts?columns=reachable,
↪computer_fqdn,connected_ips,mac_addresses&limit=10000
```

/api/v1/groups

- get all group packages. Group is found with section *group* in the package.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/groups
```

/api/v1/host_data

dmi

- get DMI (Desktop Management Interface) info for a host:

Note: # # Get additional data for a host # query: # uuid=<uuid> # field=packages, dmi or softwares

Example: get *dmi* information of host which has UUID 14F620FF-DE70-9E5B-996A-B597E8F9B4AD: https://srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-B597E8F9B4AD&field=dmi

Note: *dmi* is not the only available option. You can also lookup information using *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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=dmi
```

installed_packages

Option *installed_packages* will list all packages installed on a specific host.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=installed_packages
```

installed_softwares

Option *installed_softwares* will list all softwares installed on a specific host.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=installed_softwares
```

wsusupdates

Option *wsusupdates* will list all windows update installed on a specific host.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/host_data?uuid=14F620FF-DE70-9E5B-996A-
↪B597E8F9B4AD&field=wsusupdates
```

/api/v1/usage_statistics

Get usage statistics from the server.

Hint: This API is useful if you have several wapt servers and you want to know how many hosts are there.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/usage_statistics
```

6.15.2 API V2

/api/v2/waptagent_version

Display **`waptagent.exe`** version on the server.

```
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))
```

Hint:

This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v2/waptagent_version
```

6.15.3 API V3

/api/v3/packages

List packages on the repository, get control file on package.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/packages
```

/api/v3/known_packages

List all packages with last *signed_on* information.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/known_packages
```

/api/v3/trigger_cancel_task

Cancel a running task.

```
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

List OU seen by hosts and displayed in the WAPT console.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_ou
```

/api/v3/get_ad_sites

List Active Directory sites.

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_sites
```

/api/v3/hosts_for_package

List hosts with a specific package installed https://srvwapt.mydomain.lan/api/v3/hosts_for_package?package=demo-namepackage

```
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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/hosts_for_package?package=demo-namepackage
```

/api/v3/host_tasks_status

List tasks on a particular host.

Example with host uuid: 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))
```

Hint: This is the same exemple with a simple html request:

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/host_tasks_status?uuid=14F620FF-DE70-9E5B-
↪996A-B597E8F9B4AD
```

Attention: Next API are with POST method.

/api/v3/upload_packages

Todo: Tests

/api/v3/upload_hosts

Todo: Tests

/api/v3/change_password

Change admin password [only this account]. Request must be a python dictionnary *{}*. Keys must be:

- 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

Initialize a connection to the server.

```
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}
```

Hint: We can make a connection by html form than POST: https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_sites

/api/v3/packages_delete

Delete package with a precise version. Request must be in python list *[]*. It can takes several packages separated by commas *,*.

Example:

```
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

There is several possibilities: https://srvwapt.mydomain.lan/api/v3/reset_hosts_sid will reinitialize all host connections.

For the POST method:

Syntax is: `--data-raw` a dictionnary list with `uuids` as keys and the UUID of the hosts as values.

```
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}[
```

Hint: If you want several hosts:

```
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

If hosts are WakeOnLan enabled, this API is useful.

Syntax is `--data-raw`: a dictionnary with key *uuids* and a list of host uuids.

```
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}
```

Hint: If you want several hosts:

```
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):
"""
```

If you want to delete a host from the inventory:

```
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}
```

If you do not want to delete in the inventory server:

```
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

Todo: 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 get general information from a WAPT server.

```
# 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 Frequent problems and questions

6.16.1 I have lost my SuperAdmin password

It sometimes happens to setup a WAPT Server and then forget its password.

To reset the WAPT console *SuperAdmin* password you have to relaunch the post-configuration process on the WAPT Server.

Resetting the WAPT Linux Server password

- connect to the server with SSH;
- connect with user root (or use sudo);
- launch post-configuration script:

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

Attention: To avoid breaking the existing WAPT Server setup, accept all the other steps, **DO NOT CREATE a new private key**!

6.16.2 I lost my WAPT private key

WAPT's security and its correct functioning rely on sets of private keys and public certificates.

Losing a private key thus requires to generate a new key and its associated certificates, and then to deploy the new keys and the new certificates on the Organization's computers.

Therefore, losing a key bears some consequences, the process to recover from a lost key is not trivial, although it is relatively simple.

Generating or renewing a private key

The procedure is:

- generate a new private key/ public certificate. You will then keep the private key (file `.pem`) in a safe location;
- deploy the new certificate `.crt` on your clients in the folder `C:\Program Files (x86)\ssl` manually or using a GPO;

Re-signing packages in the repositories

WAPT packages hosted on the repositories were signed using the former private key, so you must re-sign every package of the repository using the new key.

To re-sign every WAPT packages using the new key (*base*, *host*, *group* and *unit* packages), use the command:

```
wapt-get sign-packages C:\\waptdev\\*
```

6.16.3 My private key has been stolen

Attention: WAPT security relies on protecting your private keys.

WAPT does not handle key revocation yet using a CRL.

The solution consists in deleting every `.crt` certificate associated to the stolen private key, located in the `C:\Program Files (x86)\wapt\ssl` folder.

That operation can be done using a GPO, manually, or with a WAPT package.

6.16.4 My BIOS UUID bugs

- some problems happen sometimes with some BIOSes. WAPT uses the *UUID* of the machine as the host identifier;
- the *UUID* is supposed to be unique. Unfortunately, for some OEMs (Original Equipment Manufacturers) and some manufacturing batches, BIOS *UUID* are identical;
- the machine will register in the WAPT console but it will replace an existing device, considering that the machine has only changed its name;

Solving the BIOS UUID issue

WAPT allows to generate a random *UUID* to replace the one retrieved from the BIOS.

```
wapt-get generate-uuid
```

6.16.5 WAPTdeploy does not work

Symptoms

The **waptdeploy** utility does not succeed in installing the WAPT agent.

Solving the BIOS UUID issue

Adding the waptagent.exe url

Add `waptsetupurl` argument in WAPTdeploy GPO arguments of **waptdeploy**.

```
--waptsetupurl=https://monserverserveurwapt/waptagent.exe
```

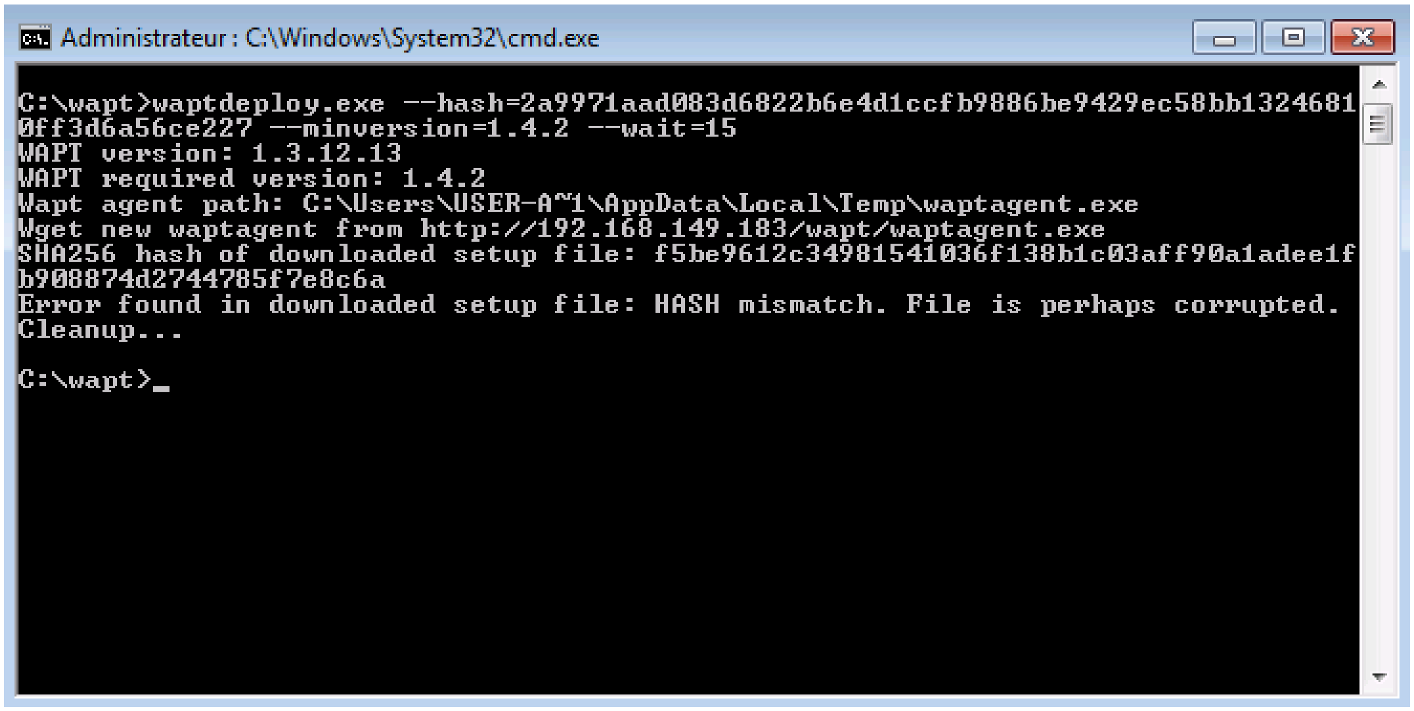
Launching WAPTdeploy locally

Launching **waptdeploy** locally can be a good method for showing errors explicitly.

Example of command to launch:

```
C:\Program Files (x86)\wapt\waptdeploy.exe --
↪hash=2a9971aad083d6822b6e4d1ccfb9886be9429ec58bb13246810ff3d6a56ce887 --minversion=1.4.2.0 --
↪wait=15
```

In our case the hash is not correct.



```
C:\wapt>waptdeploy.exe --hash=2a9971aad083d6822b6e4d1ccfb9886be9429ec58bb1324681
0ff3d6a56ce227 --minversion=1.4.2 --wait=15
WAPT version: 1.3.12.13
WAPT required version: 1.4.2
Wapt agent path: C:\Users\USER-A~1\AppData\Local\Temp\waptagent.exe
Wget new waptagent from http://192.168.149.183/wapt/waptagent.exe
SHA256 hash of downloaded setup file: f5be9612c34981541036f138b1c03aff90a1adee1f
b908874d2744785f7e8c6a
Error found in downloaded setup file: HASH mismatch. File is perhaps corrupted.
Cleanup...

C:\wapt>_
```

Fig. 168: Error with WAPTDeploy

Attention: Do not forget to start the command prompt as a *Local Administrator*.

WAPTdeploy works manually but does not work with GPO

Check that port 8088 is listening correctly on host:

```
gpresult /h gpo.html & gpo.html
```

To force the application of the GPO:

```
gpupdate /force
```

If **waptdeploy** does not show up you will have to double check the GPO settings.

# you may be using an old waptdeploy version, then download the latest version of **waptdeploy** from the WAPT store.

# thanks to Emmanuel EUGENE from French INSERM who submitted this possible cause for **waptdeploy** not functioning properly, if you are replicating domain controllers, ensure that the GPOs are correctly synchronized between your DCs and that ACLs are identically applied on the `SysVols`.

Windows does not wait for the network to be up on startup

By default Windows does not wait for the network to be up at computer startup.

This can cause problems during **waptdeploy** execution because **waptdeploy** requires network connectivity to retrieve the new WAPT agent.

You can enable the GPO: **Always wait for the network at computer startup and logon**:

> *Computer Configuration → Administrative Templates → System → Logon → Always wait for the network at computer startup and logon*

6.16.6 WAPT Exit will not launch

Despite the script actually being registered in the local security shutdown strategy, the **waptexit** script does not launch at computer shutdown.

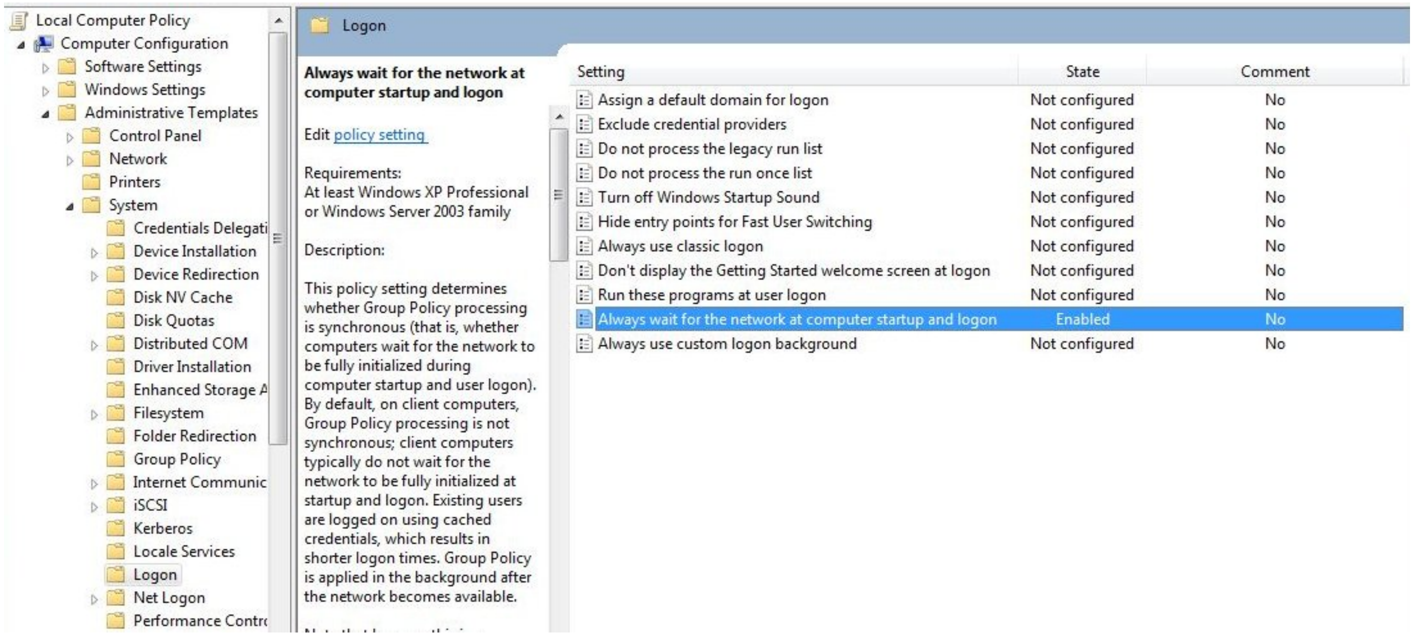
Solution: Hybrid shutdown

Windows 10 hybrid shutdown must be disabled because it causes many problems and strange behaviors, disabling Hybrid Shutdown will restore exit script execution at shutdown.

Hybrid shutdown can be disabled by setting a value in `wapt-get.ini` file *of the WAPT agent*.

There is a WAPT package to solve the Hybrid Shutdown problem:

- a WAPT package exists for this purpose: tis-disable-hybrid-shutdown.



Local Computer Policy
Computer Configuration
Software Settings
Windows Settings
Administrative Templates
Control Panel
Network
Printers
System
Credentials Delegati
Device Installation
Device Redirection
Disk NV Cache
Disk Quotas
Distributed COM
Driver Installation
Enhanced Storage A
Filesystem
Folder Redirection
Group Policy
Internet Communic
iSCSI
Kerberos
Locale Services
Logon
Net Logon
Performance Contr

Logon

Always wait for the network at computer startup and logon

Edit policy setting

Requirements:
At least Windows XP Professional or Windows Server 2003 family

Description:

This policy setting determines whether Group Policy processing is synchronous (that is, whether computers wait for the network to be fully initialized during computer startup and user logon). By default, on client computers, Group Policy processing is not synchronous; client computers typically do not wait for the network to be fully initialized at startup and logon. Existing users are logged on using cached credentials, which results in shorter logon times. Group Policy is applied in the background after the network becomes available.

Setting	State	Comment
Assign a default domain for logon	Not configured	No
Exclude credential providers	Not configured	No
Do not process the legacy run list	Not configured	No
Do not process the run once list	Not configured	No
Turn off Windows Startup Sound	Not configured	No
Hide entry points for Fast User Switching	Not configured	No
Always use classic logon	Not configured	No
Don't display the Getting Started welcome screen at logon	Not configured	No
Run these programs at user logon	Not configured	No
Always wait for the network at computer startup and logon	Enabled	No
Always use custom logon background	Not configured	No

Solution: Windows Home edition

Local security policies are not available when using a Windows Home edition computer, so it is normal that the script will not launch. To circumvent the problem, use scheduled tasks.

The workaround consists in using a scheduled task that will launch `C:\Program Files (x86)\wapt\wapt-get.exe` with the argument `upgrade`.

Solution: corrupted local GPO

It sometimes occurs that local security policies on the computer are corrupted.

One of the possible solutions is to remove local security strategies by deleting the file `C:\Windows\System32\GroupPolicy\gpt.ini`, to restart the computer, and finally to re-install the shutdown scheduled tasks:

```
wapt-get add-upgrade-shutdown
```

If the problem occurs again, this may mean that another application also manipulates the local GPO.

6.16.7 WAPTExit halts after 15 minutes and does not finish the installing the packages

By default, Windows shutdown scripts are only allowed to run for 15 minutes.

If a script has not finished before that limit, it will be interrupted.

Solution: increase the installation timeout

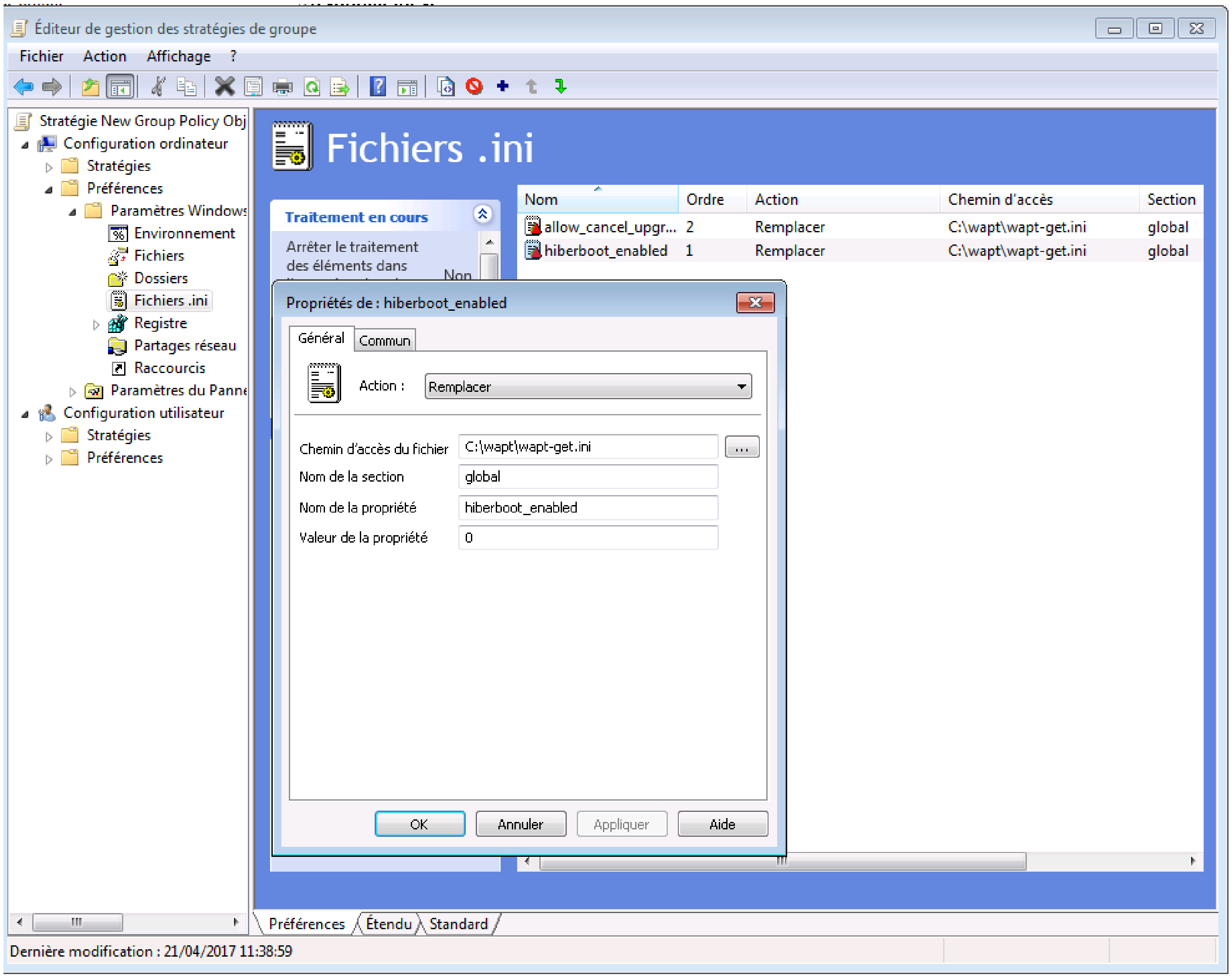
To solve that problem, increase the `preshutdowntimeout` value and the `max_gpo_script_wait` value.

Define these values in `C:\Program Files (x86)\wapt\wapt-get.ini` file to change the default behavior.

```
max_gpo_script_wait=180
pre_shutdown_timeout=180
```

The WAPT package tis-wapt-conf-policy sets this configuration.

The other solution may be to use the GPO `File.ini`.



Nom	Ordre	Action	Chemin d'accès	Section
allow_cancel_upgr...	2	Remplacer	C:\wapt\wapt-get.ini	global
hiberboot_enabled	1	Remplacer	C:\wapt\wapt-get.ini	global

Propriétés de : hiberboot_enabled

Action :	Remplacer
Chemin d'accès du fichier	C:\wapt\wapt-get.ini
Nom de la section	global
Nom de la propriété	hiberboot_enabled
Valeur de la propriété	0

Dernière modification : 21/04/2017 11:38:59

Fig. 169: GPO ini File

6.16.8 Error message when opening the WAPT console

Connection refused

The WAPT console can not contact the WAPT Server on port 443.

- check whether the **Nginx** web service is running on the WAPT Server:

```
ps aux | grep nginx
```

- if **Nginx** is not running, restart the **Nginx** service:

```
service nginx restart
```

- if **Nginx** still does not start, you'll need to analyze journal logs in `/var/log/nginx/` on Linux or in `C:\Program Files (x86)\wapt\waptserver\nginx\logs` on Windows.

Service unavailable

It is possible that the *waptserver* service is stopped.

- check whether **waptserver** is running:

```
ps aux | grep wapt
```

- if the command returns nothing, then start the **waptserver** using:

```
service waptserver start
```

Error connecting with SSL . . . verify failed

The WAPT console seems not to be able to verify the server's HTTPS certificate.

Attention: Before doing anything, be sure that your are not facing a MITM (Man in the Middle) attack!

Note: If you have just redone your WAPT Server and that you use a self-signed certificate, you can recover the old keys of your old WAPT Server in `/opt/wapt/waptserver/apache/ssl`.

- close your WAPT console;
- delete the folder `%appdata%\..\Local\waptconsole`;
- launch the command `wapt-get enable-check-certificate`;
- be sure that the previous command has gone well;
- restart the WAPT service with `net stop waptservice && net start waptservice`;
- restart the WAPT console;

In case you do not use the certificate pinning method, this tells you that the certificate sent by the server can not be verified with the python **certifi** bundle of certificates. Be sure to have the full chain of certificates on the WAPT Server.

6.16.9 Problems when enabling enable-check-certificate

Message "Certificate CN ### sent by server does not match URL host ###"

This means that the CN in the certificate sent by the WAPT Server does not match the value of the *wapt_server* attribute in `wapt-get.ini`.

Two solutions:

- check the value of *wapt_server* in your `wapt-get.ini`;

  If the value is correct, this surely means that an error has happened during the generation of the self-signed certificate during server post-configuration (typing error, . . .).

  You must then regenerate your self-signed certificates.

- on the WAPT Server, delete the content of the `/opt/wapt/waptserver/apache/ssl/` folder.

  Then, relaunch the postconfiguration script (the same as the one used during initial installation, with the same arguments and values).

  Then, be sure that the value of *FQDN for the WAPT Server* is correct.

- you may now retry **`enable-check-certificate`**.

6.16.10 Problems when creating a package

Creating a package via the WAPT console

The drag and drop method of a software in the WAPT console does not work:

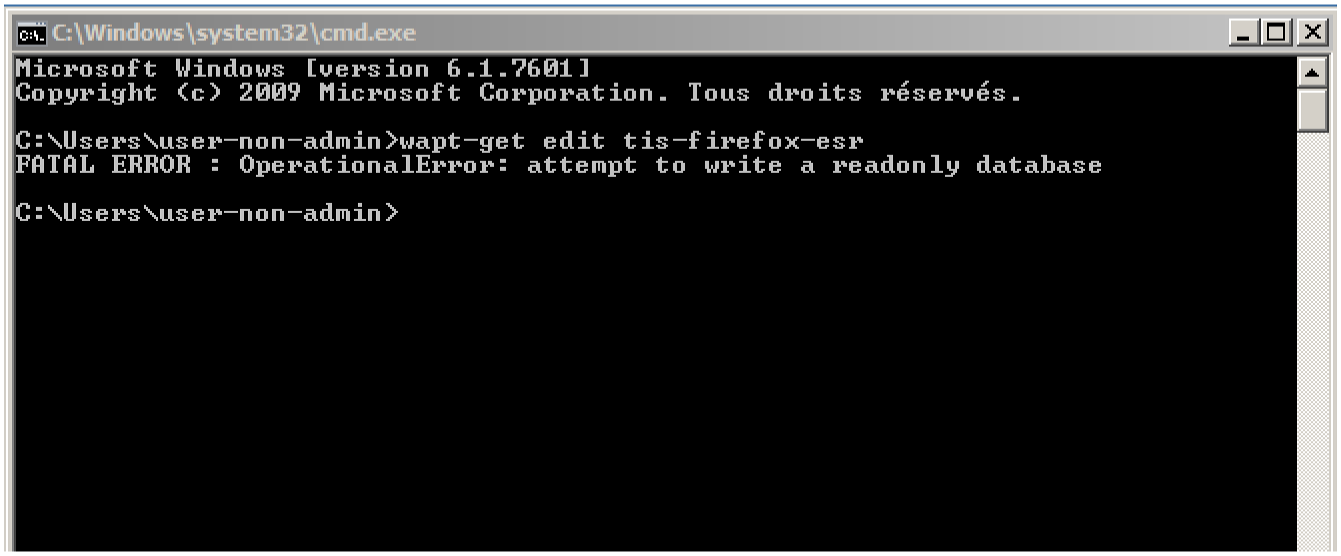
- the method will not work if the WAPT console has been started without *Local Administrator* privilege;
- the method will not work if the WAPT console has been started with UAC;

  Simple alternative solution: go to *Tools* → *Create a package template from an installer* → *Choose the installer*.

- the WAPT console does not fill in automatically the informations in the fields:
  - there are special characters in some file path of the binary;
  - the installer does not provide the desired informations;

Problem with rights in the Windows Command Line utility

When editing a package, if the following message appears:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\user-non-admin>wapt-get edit tis-firefox-esr
FATAL ERROR : OperationalError: attempt to write a readonly database

C:\Users\user-non-admin>
```

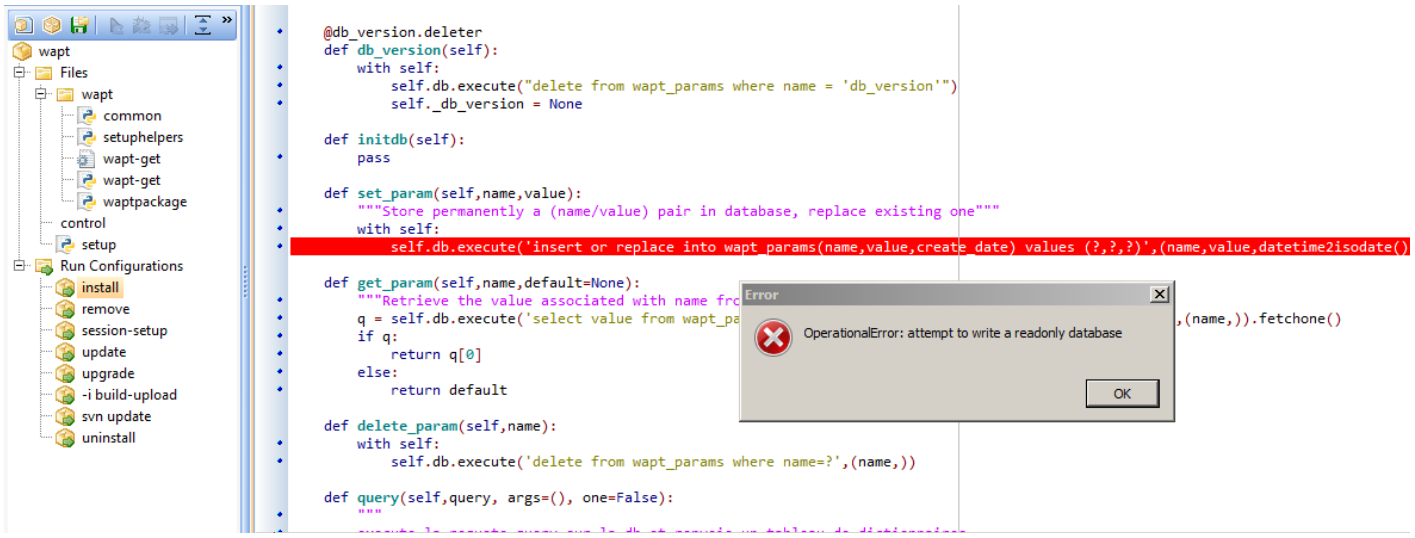
Fig. 170: OperationnalError: attempt to write a read-only database

Solution

Open a session as *Local Administrator* and redo the desired action.

Problems with access rights and PyScripter

When trying to install a package from **PyScripter**, if the following message appears:



```
    @db_version.deleter
    def db_version(self):
        with self:
            self.db.execute("delete from wapt_params where name = 'db_version'")
            self._db_version = None

    def initdb(self):
        pass

    def set_param(self,name,value):
        """Store permanently a (name/value) pair in database, replace existing one"""
        with self:
            self.db.execute('insert or replace into wapt_params(name,value,create_date) values (?,?,?)',(name,value,datetime2isodate()

    def get_param(self,name,default=None):
        """Retrieve the value associated with name fro
        q = self.db.execute('select value from wapt_pa                                                ,(name,)).fetchone()
        if q:
            return q[0]
        else:
            return default

    def delete_param(self,name):
        with self:
            self.db.execute('delete from wapt_params where name=?',(name,))

    def query(self,query, args=(), one=False):
        """
```

OperationalError: attempt to write a readonly database

Fig. 171: OperationnalError: attempt to write a read-only database

Solution

Open a session as *Local Administrator* and redo the desired action.

My WAPT package is too big and I can not upload it on the repository

When a package is too big, it is necessary to **build** it locally then **upload** it with **WinSCP** or an equivalent utility.

Solution

- build the package with **PyScripter** or manually *build the package*.

  Hint: If the previous **upload** failed, you can find the package in `C:\waptdev`.

- download and install **WinSCP** using WAPT:

  ```
  wapt-get install tis-winscp
  ```

- using **WinSCP**, **upload** your package in `/var/www/html/wapt/` path of you Linux server.
- once the upload has finished, you'll need to recreate the `Packages` index file on your repository:

  ```
  wapt-scanpackages /var/www/wapt/
  ```

6.16.11 WAPT package in error

Problem installing a package

Symptoms

I have a package that returns in error and the software is not installed on the computer when I physically go to check on the computer.

Explanation

An error has occurred during the execution of the `setup.py`.

You can read and analyze error messages returned in the console and try to understand and solve them.

The installation of the package will be retried at each **upgrade** cycle until the package does not return an error.

Solution

- if WAPT returns an error code, research the error code on the Internet;

  Example for a MSI: *1618*: another installation in already running. Restarting the computer should solve the problem.

  Note: MSI error codes are available by visiting this website.

- go to the computer and try to install the package with the WAPT command line utility. Then check that the software has installed;

  Attention: Once the silent installation has finished, do nothing else.

  The objective is to reproduce the behavior of the WAPT agent.

- if the package installs silently in user context, this may mean that the software installer does not work in *SYSTEM* context;
- if it is still not working, launch the installation manually. It is possible for an error to appear explicitly describing the problem (ex: missing dependency, etc);
- it is possible that the installer does not support installing over an older version of the software, so you will have to explicitly remove older versions of the application before installing the new one;

Error "timed out after seconds with output '600.0'"

Symptoms

Some packages return the following error in the WAPT console:

```
"Erreur timed out after seconds with output '600.0'"
```

Explanation

By default, when installing a package **run**, **install_msi_if_needed**, WAPT will wait 600 seconds for the installer to finish its task.

if the installer has not finished in this delay, WAPT will stop the running installation.

Solution: large software installs

If the software to be installed is known to be big (Microsoft Office, Solidworks, LibreOffice, Katia, Adobe Creative Suite), it is possible that the 600 second delay will be too short.

You will have to increase the timeout value, ex: *timeout* = 1200:

```
run('"setup.exe" /adminfile office2010noreboot.MSP',timeout=1200)
```

Error “has been installed but the uninstall key can not be found”

Symptoms

Some packages return the following error in the WAPT console:

```
XXX has been installed but the uninstall key can not be found.
```

Explanation

WAPT relies on Windows to install *.msi* binaries with **install_msi_if_needed** and *.exe* binaries with **install_exe_if_needed**.

By default, WAPT accepts return codes *0* (OK) and *3010* (computer restart required) and it verifies that the *uninstall key* is present.

Unfortunately, we can not fully trust these return codes, so WAPT does additional checks after completing the installation to make sure that all has gone well:

- it checks the presence of the *uninstall key* on the computer;
- it checks that the version number of the software is equal or greater than the version number in the `control` file;
- if this is not the case, it infers that the software may not be present on the computer;

The function returns the package in error. The installation will be retried at every **upgrade** cycle until the package returns no error.

Solution

Attention: Before doing anything, it is advisable to go physically to the computer returning in error and to **manually check whether the software has correctly installed**. If the software has not installed correctly, refer to the *section of this documentation on installing a package*.

- if the software has installed correctly, this may mean that the uninstall key or the software version in the package is not correct;
- retrieve the correct *uninstall key* and make changes to the WAPT package accordingly;
- if the error happens when using the **install_msi_if_needed** function, this means that the MSI installer is badly packaged and that it is returning an incorrect *uninstall key*;

Error “has been installed and the uninstall key found but version is not good”

Symptoms

Some packages return the following error in the WAPT console:

```
has been installed and the *uninstall key* found but version is not good
```

Explanation

When using **install_msi_if_needed** or **install_exe_if_needed** functions, additional checks are performed to make sure that all has gone well.

Solution

Attention: Before doing anything, it is advisable to go physically to the computer returning in error and to **manually check whether the software has correctly installed**. If the software has not installed correctly, refer to the *section of this documentation on installing a package*.

Solution: with install_msi_if_needed

The informations being extracted from the MSI installer, this means that the MSI file does not return correct values or that the *uninstall key* is incorrect.

You can check using the Windows Command Line utility:

```
wapt-get list-registry
```

If the returned key is not that which has been entered in the install section of the `setup.py`, it is not possible to use **install_msi_if_needed**.

You must review the install section of your `setup.py`, use the **run()** function and manually manage exceptions.

Solution: with install_exe_if_needed

This probably means that the version number entered in the **install_exe_if_needed** function is not correct. Make corrections to the WAPT package accordingly.

Note: If the `min_version` argument has not been entered, WAPT will try to retrieve the version automatically from the exe installer.

You can check the *uninstall key* and version number using the command:

```
wapt-get list-registry
```

If no version is provided with the **wapt-get list-registry** command, this means that the software installer does not provide an *uninstall key*.

Two solutions:

- use the argument `get_version` to provide the path to another `uninstallkey`;

```
def install():

    def versnaps2(key):
        return key['name'].replace('NAPS2 ','')
```

(continues on next page)

(continued from previous page)

```
    install_exe_if_needed('naps2-5.3.3-setup.exe',silentflags='/VERYSILENT',key='NAPS2 (Not␣
↪Another PDF Scanner 2)_is1',get_version=versnaps2)
```

- providing an empty value for `min_version` tells WAPT not to check for versions;

```
min_version=' '
```

Attention: With this method, **versions are no longer checked during updates!**

6.16.12 Frequent problems caused by Anti-Virus software

Some Anti-Virus software falsely raise errors when checking some internal components of WAPT.

Among the components is **nssm.exe** used by WAPT as a service manager for starting, stopping and restarting the WAPT service.

Below is a list of useful exceptions to declare in your central AV interface to solve false positives related to WAPT:

```
"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

If you get this message, it may mean that you have not set up correctly your development environment. Visit this *section of the documentation on setting up UTF-8 (no BOM)*.

6.16.14 I have a lot more hosts in the console than I have host packages on my server?

Following a remark from Philippe LEMAIRE from the Lycée Français Alexandre Yersin in Hanoï, if you use the Enterprise version of WAPT and you make heavy use of the *unit packages* or *profile packages*, you may realize that you will have many more hosts in your console than *host packages* on you WAPT server. **This is normal**.

In fact *unit packages* and *profile packages* are not explicitly assigned to the host (i.e. as dependencies in the *host package*) but are implicitly taken into account by the WAPT agent dependency engine during the WAPT upgrade.

So one might have no *host package* on the server if only *unit packages* are used for managing a fleet of devices.

6.17 Common mistakes

6.17.1 Using a network drive to store and deliver WAPT packages

The standard way WAPT works is with a secure web server delivering WAPT packages to the WAPT Clients.

Tranquil IT advises against using a network drive for delivering WAPT packages for several reasons:

- a web server is extremely easy to setup, secure, maintain, backup and monitor;
- to work correctly, a WAPT package needs to be self-contained. Indeed, we do not know if the network will be available at the time of the installation launch (for example if we have a waptexit that starts when the workstation is shutting down on a network with 802.1x user authentication, there will no longer be a network available at the time of installation). The self-contained nature of WAPT makes it more deterministic than other deployment solutions;
- network congestion may result from downloading large packages on large fleets of devices because you have less control over bandwidth rates or you may not be able to finish a partial download;
- this method breaks or at least weakens the security framework of WAPT;
- this method does not allow you to expose your repositories to internet for your traveling personnel;

Attention: Even though WAPT *can work* independently of the transport mode, **Tranquil IT will not officially support using a network drive to store and deliver WAPT packages**.

6.17.2 Using the register() function in your audit scripts

The register() function forces the sending to the WAPT server of the WAPT agent's hardware and software inventory.

This function is very taxing on the server's performance because it forces the server to parse a relatively large JSON (Java Script Object Notation) BLOB and to inject the result into the PostgreSQL database.

The function is by default triggered manually or when a new package upgrade is applied.

When you use the register() function in an audit script, it will run every time the audit script is triggered and load the server with no apparent benefit.

Therefore, **we do not recommend the use of the register() function in audit scripts**.

6.18 Glossary

Administrator

Administrators

Package Developer

Package Developers In WAPT, an **Administrator** is a person with a **Code Signing** certificate that can sign packages, whether or not they contain python code or binary files, and upload the packages to the main repository.

Local Administrator

Local Administrators A **Local Administrator** is a person with administrative rights on computers managed with WAPT.

Package Deployer

Package Deployers A **Package Deployer** is a person that can create and sign WAPT packages that do not contain python code or binaries, eg. *host* and *group* packages, and upload them to the repository. They are typically members of local IT teams that have knowledge of specific user needs to be satisfied by deploying WAPT packages built by central IT teams.

SuperAdmin The **SuperAdmin** is a *User* whose login and password are set during the post-configuration of the WAPT Server. In the Community version of WAPT, he is the unique *Administrator* of WAPT.

User

Users A **User** is an individual who uses a machine that is equipped with a WAPT agent (WAPT Enterprise and Community).

Organization

Organizations The **Organization** is the perimeter or responsibility within which WAPT is used.

ANSSI **Agence Nationale de la Sécurité des Systèmes d'Information** is a French service assuming Cyber Security for the French State and has a responsibility for counseling and helping government agencies and Critical Infrastructure Operators (OIV) with securing their IT systems.

DNS **Domain Name System** translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices.

FQDN **Fully Qualified Domain Name** is a domain name that specifies its exact location in the tree hierarchy of the Domain Name System. It specifies all domain levels, including the top-level domain and the root zone. FQDN example: wapt.nantes.pdl.organisation.fr.

EPEL **Extra Packages for Enterprise Linux** is an extra repository for CentOS and RedHat.

GPO **Group Policy Object** is a feature of the Microsoft Windows NT family of operating systems that controls the working environment of user accounts and computer accounts. Group Policy provides the centralized management and configuration of operating systems, applications, and users' settings in an Active Directory environment.

IDE **Integrated Development Environment** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.

MMC **Microsoft Management Console** is a component of Windows that provides system administrators and advanced users an interface for configuring and monitoring the system.

NAT **Network Address Translation** is a mechanism to allow computers from one network, usually with private IP addresses, to connect to another network, usually the Internet, using only one outgoing IP address of the NAT router.

Setuphelpers **SetupHelpers** is a python library specifically designed for WAPT. It's main purpose is to provide a set of functions useful for package development, file and folder manipulation, shortcut creation, etc.

SRV A **Service Record** (SRV record) is a specification of data in the Domain Name System defining the location, i.e. the hostname and port number, of servers for specified services.

virtualhost **Virtual hosting** is a method for hosting multiple domain names (with separate handling of each name) on a single server (or pool of servers). This allows one server to share its resources, such as memory and processor cycles, without requiring all services provided to use the same host name. The term virtual hosting is usually used in reference to web servers but the principles do carry over to other Internet services.

Websocket **Websockets** is a network protocol extending HTTP protocol in order to allow bidirectional client-server socket using the TCP connexion to a web server.

UUID **Universally Unique IDentifier** is a unique standard normalized identifier for practical purposes. In WAPT every computer is referred uniquely by its UUID. For more information see https://en.wikipedia.org/wiki/Universally_unique_identifier.

CNAME field A **CNAME** DNS field is an alias name for another A *DNS* field.

A field A **DNS A field** matches a name (generally the name of a machine) with an IP address.

Certificate Authority An CA (Certificate Authority) is a third party entity that vouches the identity of individuals or services exchanging information.

PKI **Public Key Infrastructure** is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information.

6.19 History of WAPT

6.19.1 WAPT 0.8 Community (novembre 2013)

- cleaning and stabilisation of console code;
- transfer of **`waptservice`** local service from **`Lazarus`** to **`Python`**;
- addition of firewall rules to allow access to **`waptservice`** from **`waptserver`**;
- display wapt package update status in wapt console;
- possibility to import packages from TIS repository and adding them to local repository;
- build system embryo;
- after **`waptsetup`** customization build in **`waptconsole`**, possibility to upload it directly from **`waptconsole`**;

6.19.2 WAPT 0.9 Community (septembre 2014)

- Windows **`waptserver`** packaging;
- Windows postconfiguration tool to ease server installation;
- transition to https for all connexions;
- possibility to ask remote package removal;
- improving error feedbacks;
- display package description when selecting in the WAPT console;
- host identification through computer account thanks to Kerberos (inactive by default);
- waptselfservice group to delegates installation group;
- possibility to push **`waptagent`** installation on a host through MS-RPC (if firewall allows it);

6.19.3 WAPT 1.0 Community (févier 2015)

The release of that milestone has been announced at Bruxelles FOSDEM the 1st of February 2015.

- internationalization of the console, service, server and notification messages;
- scenari documentation;
- addition of icon for packages for a better rendering on wapt website store.wapt.fr;
- windows installer improvements to avoid conflicts during installation (open ports check, etc.);
- bug correction with obscure network specificities;

- `builbot` continuous building process;

6.19.4 WAPT 1.1 Community (février 2015)

- several bugs and anomalies have been corrected on 1.0.0 version thanks to feedback from users;
- display of reachability of hosts in the WAPT console;

6.19.5 WAPT 1.3 et 1.5 Community (2016-2017)

- integration of the authentication of local service in Windows Active Directory or Samba-AD;
- adding of a integrated agent building wizard in the WAPT console;
- role segregation between *Package Developers* and *Package Deployers*;
- replacement of `MongoDB` by `PostgreSQL` with JSON extension;
- Websockets implementation;
- host identification through a shared secret for workgroup hosts that cannot access MSAD or Samba-AD domain;

6.19.6 WAPT 1.5 Enterprise (début 2018)

The features and functionalities described in the section are only relevant to the **Enterprise** version of WAPT.

- management by Organisational Units (Machine OU);
- taking into account of the Certificate Authority for signing packages, in addition to individual certificates;
- kerberos based SSO authentication of *Administrators* in the WAPT console;

6.19.7 WAPT 1.6 (August 2018)

- recurring audit function to insure configurations are maintained over time (**Enterprise**);
- (tech preview) Windows Update management in WAPT, reproducing WSUS functionnalities (**Enterprise**);
- authentication by certificate of the WAPT client when accessing a repository or connecting to the WAPT Server (inventory, websockets);

6.19.8 WAPT 1.7

- customizable WAPT reporting integrated within the WAPT console (**Enterprise**);
- discrimination between user self-service packages and restricted packages that may be installed only by *Administrators* (**Enterprise**);
- global updates according to the package's criticity level (**Enterprise**);
  - immediate upgrade for critical updates;
  - with the user accepting the upgrade if it impacts the user's current activities;
- software and configuration management using AD Organizational Units (*unit* packages) (**Enterprise**);

6.19.9 WAPT 1.8

- client agent for Linux Debian, Linux CentOS, Ubuntu and Apple MacOS;
- built-in WAPT packages repository replication;
- built-in repository selection rules;

6.20 Applying best practices to packaging software

Note: _benwa is a system administrator and he has authorized Tranquil IT to republish his excellent rant on reddit Developers, you can make sysadmins happier.

6.20.1 Environnement variables

- Environmental variables have been around since DOS. They can make your (and my) life easier.

6.20.2 Program directories

- Not every system uses `C:\` as the main drive. Some enterprises use folder redirection, and relocate the Documents folder. Some places in the world don't speak English and their directories reflect that. **Use those environmental variables to make your programs "just work"**:
  - `%SystemDrive%` is the drive where `%SystemRoot%` is located. You most likely don't need to actually know this;
  - `%SystemRoot%` is where the Windows directory is located. You hopefully don't care about this. Leave the Windows directory alone;
  - `%ProgramFiles%` is where you should place your program files, preferable in a `Company\Program` structure;
  - `%ProgramFiles(x86)%` is where you should place your 32-bit program files. Please update them for 64-bit. 32-bit will eventually be unsupported, and business will be waiting for you to get your shit together for far longer than necessary;
  - `%ProgramData%` is where you should store data that isn't user specific, but still needs to be written to by users (Users don't have write access to this folder either).

    Your program shouldn't require administrator rights to run as you shouldn't have us writing to the `%ProgramFiles%` directory. Also, don't throw executables in here.
  - `%Temp%` is where you can process temporary data. Place that data within a unique folder name (maybe a generated GUID perhaps) so you don't cause an incompatibility with another program. Windows will even do the cleanup for you. Don't put temporary data in in `%ProgramData%` or `%ProgramFiles%`;
  - `%AppData%` is where you can save the user running your program settings. This is a fantastic location that can by synced with a server and used to quickly and easily migrate a user to a new machine and keep all of their program settings. Don't put giant or ephemeral files here.

    You could be the cause of a very slow login if you put the wrong stuff here and a machine needs to sync it up. **DON'T PUT YOUR PROGRAM FILES HERE**. The business decides what software is allowed to run, not you and a bunch of users who may not know how their company's environment is set up;

  - `%LocalAppData%` is where you can put bigger files that are specific to a user and computer. You don't need to sync up a thumbnail cache. They won't be transferred when a user migrates to a new machine, or logs into a new VDI station, or terminal server. **DON'T PUT YOUR PROGRAM FILES HERE EITHER**;

Note: More and more of you software editors offer *portable* versions of your software that will install in and run from `%AppData%` or `%LocalAppData%`. Your aim is to let users install software even though they are not Local Administrators and you market that as a feature, although it is more of a security NOGO. Even worse, you tend to make it difficult to find the proprer *MSI* that would allow your customers to correctly install your software in `%ProgramFiles%`. Please, make it easy to find your *MSI* that will install in `%ProgramFiles%`, this way you'll make your customer AppLock and Software Restriction Policies work well and their sysadmins happy.

You can get these directory paths through API calls as well if you don't/can't use environmental variables.

6.20.3 Logs

- Use the Windows Event Log for logging. It'll handle the rotation for you and a sysadmin can forward those logs or do whatever they need to. You can even make your own little area just for your program.

6.20.4 Error codes

- Use documented Error Codes when exiting your program.

6.20.5 Printing

- Use the Windows printing API and do not use direct printing in your program.

6.20.6 Distribution

- Distribute your program in MSI. It is the standard for Windows installation files (even though Microsoft sometimes doesn't use it themselves).
- Sign your installation file and executables. It's how we know it's valid and can whitelist in AppLocker or other policies.

Note: Applocker and Software Restriction Policies can be very effective and the **management of these policies can be made simpler with WAPT**.

6.20.7 Update

- Want to have your application update for you? That can be fine if the business is okay with it. You can create a scheduled task or service that runs elevated to allow for this without granting the user admin rights. I like the way Chrome Enterprise does it: gives a GPO to set update settings, the max version it will update to (say 81.* to allow all minor updates automatically and major versions are manual), and a service. They also have a GPO to prevent user-based installs;

Note: WAPT is designed for businesses that don't allow users to run software updates, which is the policy often chosen in large security conscious enterprises.

6.20.8 Version numbers

- Use semantic versioning (should go in the version property in the installer file and in the Add/Remove Programs list, not in the application title) and have a changelog. You can also have your installer download at a predictable location to allow for automation. A published update path is nice too;

Note: If you apply this practice, then you will make system administrators who deploy your software updates using the *WAPT function def_update()* **very happy**!!

6.20.9 GPO

- ADMX templates are dope;

Note: We completely agree with you _benwa on this at Tranquil IT. If developers advise their customers to use GPOs to deploy their software or system or users settings, then, **they must know that GPOs are not fully reliable**.

Instead, package your software, your system and user configurations using WAPT. A `setup.py` is so much easier than an *xml* file for system admins to audit before deploying.

WAPT packages can be applied recursively to trees of Organisational Units, so your WAPT package will behave in production exactly as a GPO would, **just much easier**.

6.20.10 License dongles

- USB license dongles are a sin. Use a regular software or network license. I'm sure there are off the shelf ones so you don't have to reinvent the wheel;

Note: You can make your software accept a licence key as a parameter in your *msi* executable.

WAPT can be used to assign licence keys to individual workstations at install using a *method that ensures that the licence key can not be read during transport*.

Then, if you want your software to call home to check on the validity of the licence, make the routine work with *proxies*.

6.20.11 Networking

- Don't use that damn custom IPv4 input field. Use FDQNs. IPv6 had been around since 1998 and will work with your software if you just give it a chance;
- The Windows Firewall (can't really say much about third party ones) is going to stay on. Know the difference between an incoming and outgoing rule. Most likely, your server will need incoming. Most likely, you clients won't even need an outgoing. Set those up at install time, not launch time. Use Firewall Groups so it's easy to filter. Don't use Any rules if you can help it. The goal isn't to make it work, it's to make it work securely. If you don't use version numbers in your install path, you might not even have to remake those rules after every upgrade;
- Proxies are good for hygiene and proxies are now a default security feature not just in corporate IT environments, but even on small networks. Making your software not compatible with proxies will require the network administrators of your customer to make and maintain special rules in their firewall, just for you. It is easy to code your software to work with proxies, so please do!

6.20.12 PDFs

- Don't ship a software that requires allowing javascript to run in PDF readers. Business logic should be run before outputting to a PDF, not after.

Note: *PDF* files is the file format people use by default to exchange documents. PDF readers are meant to display documents, not execute unsigned programs.

6.21 WAPT release Strategy

WAPT does not release on a fixed date schedule.

Instead Tranquil IT will release a new major version of WAPT when new major functional updates are integrated into the core of the product.

Tranquil IT will release intermediary minor versions of WAPT between major releases to correct functional and security defects.

6.22 Release delay between the Enterprise and the Community versions

A new major version will be released as **Community** and that same version will be released as an **Enterprise** RC1 (Release Candidate #1). Before releasing, the Community version will have undergone thorough internal testing to insure no regression has slipped into the core of WAPT.

Community users are encouraged to update to the new major version.

The Enterprise release will cycle through several RCs and the final general availability Enterprise version will then become available between 4 and 8 weeks after the release of the Community version.

This delay will provide several benefits to the Enterprise release process:

- allow additional time to perform in depth testing of the new Enterprise features while avoiding regressions;
- allow Tranquil IT to work with a small set of selected Enterprise customers to insure upgrade procedures work smoothly;

- give a little time to the forum and the mailing list to index questions and answers to eventually include into the official documentation;
- give the documentation team a fixed functional target to document the new or improved features;
- give the translation team the necessary delay to update translations;
- give the communication and marketing team a fixed functional target and a capacity to backward schedule announcements, video podcasts and overall promotion;

6.23 Contributing to WAPT

The public part of the project is maintained on github at https://github.com/tranquilit/WAPT.

Todo: work still in progress

6.23.1 Recompiling WAPT from source

Building the WAPT Agent for Windows

WAPT requirements

Python environment

- Python 2.7.13;
- client python libraries in `requirements.txt`;
- server python libraries in `requirements-server.txt`;

Lazarus environment

WAPT relies on the following third-party freepascal/ lazarus librairies:

- pl_indy: https://www.pilotlogic.com/sitejoom/index.php/115-wiki/ct-packages/networking/271-pl-indy;
- 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;
- python4delphi: https://github.com/pyscripter/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/;

Tranquil IT packages

- pltis_python4delphi: https://github.com/tranquilit/pltis_python4delphi;
- pltis_utils: https://github.com/tranquilit/pltis_utils: subset of libraries from JEDI JCL project adapted to lazarus;
- pltis_sogrid: https://github.com/tranquilit/pltis_sogrid;
- pltis_superobject: https://github.com/tranquilit/pltis_superobject;

Creating a development environment with virtualenv

With a clean Windows installed:

- install python2.7.15 from https://www.python.org/ftp/python/2.7.15/python-2.7.15.msi;
- upgrade **python-setuptools**:

```
c:\python27\python -m pip install -U pip setuptools
```

- create a development environment with `virtualenv`;

  ```
  mkdir c:\tranquilit
  git clone git@github.com:tranquilit/WAPT.git (ou git clean -fxd ...)
  cd c:\tranquilit\wapt init_workdir.bat
  ```

Installing the WAPT development environment

On a clean Windows 7 install as a *Local Administrator*:

- install the WAPT agent from https://srvwapt.mydomain.lan/wapt/waptagent.exe;
- deactivate UAC;
- show hidden files and file extensions;
- increase the width of the CMD windows and flip to quick edit mode;

Installing 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
```

(continues on next page)

(continued from previous page)

```
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
```

Installing the server environment on Windows

```
cd \tranquilit\wapt
waptpython waptserver\winsetup.py all
```

Creating the InnoSetup installers

- install Innosetup from https://jrsoftware.org/download.php/ispack-unicode.exe

The `.iss` files are located in `C:\tranquilit\wapt\waptsetup`;

The **`waptsetup`** installer includes the python libraries, the command line tool **`wapt-get`**, the local webservice **`waptservice`**, the packaging tool and the WAPT console **`waptconsole`**.

The file `waptserver.iss` allows to build an installer that includes a Nginx web server in front and the Flask webservice **`waptserver.py`**.

The `waptstarter` installer only includes the local webservice and the command line tool **wapt-get**. It does not include the WAPT console **waptconsole**, nor the packaging tools.

Right-click on the .iss file → Compile ` will compile an installer with :program:`InnoSetup.

or using the command line:

```
"C:\Program Files (x86)\Inno Setup 5\ISCC.exe" C:\tranquilit\wapt\waptsetup\waptsetup.iss
```

The installer's global parameters are defined with *#define* in the file header.

If you do not sign the installers, you may comment the lines `#define signtool ...`

Building the WAPT Agent for MacOS

Generating the agent package

- if you do not have access to the **sudo** command, you'll need to enable the root user;
- from the root of the WAPT directory, navigate to waptservice/pkg;
- execute the *createpkg* script with administrator rights;

```
sudo ./createpkg.py
```

It may ask for additional software (the Command Line Developer Tools) and install them after a prompt which you should answer *Yes* to;

- the agent package should have been generated, under a name along the lines of `tis-waptagent-1.7.6.6550-tismacos-fdc24bca.pkg`;

Installing the agent package

- execute the following command on your MacOS:

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

If the installation is successful, you should have the wapt files in `/opt` and access to the **wapt**, **wapt-get**, **waptpython** and **waptservice** commands.

- the agent should launch at the next reboot, but you probably want to start it right away with the following command:

```
sudo launchctl load -w /Library/LaunchDaemons/wapt.plist
```

Building the WAPT Agent for Linux

Building the environment on Debian Linux

```
mkdir ~/tranquilit/
cd ~/tranquilit/
git clone git@github.com:tranquilit/WAPT.git
cd ~/tranquilit/wapt/waptserver/deb
```

(continues on next page)

(continued from previous page)

```
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.

These licenses help us achieve our goal of providing reliable and long-lived software products through collaborative open source software development.

In all cases, contributors retain full rights to use their original contributions for any other purpose outside of WAPT while providing WAPT and its projects the right to distribute and build upon their work.

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.

Hint: 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 *Configuring Client-Side Certificate Authentication*.

- [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 External components licenses used in WAPT

WAPT software development was started in March 2012 by Tranquil IT.

Developments done within the WAPT public project are licensed under the GNU Public License v3.0.

The extensions incorporated into the Enterprise version of WAPT are proprietary.

Table 31: External components licenses used in WAPT

WAPT components	License
Python	Python Software License
Python Libraries	Various open-source licenses
Lazarus	GNU Public Licence
Lazarus Component Library	GNU Lesser General Public License
Lazarus Libraries	Various open-source licenses
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 Contacting Tranquil IT

Contact us for more 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 in French: https://forum.tranquil.it/
- Forum in English: https://www.reddit.com/r/WAPT
- Mailing-list: https://lists.tranquil.it/listinfo/wapt

CHAPTER SEVEN

INDICES AND TABLES

- genindex
- *Glossary*
- search