WAPT Documentation

Release 2.4

Tranquil-IT

Sep 20, 2024

PRESENTING WAPT

Welcome to WAPT's official documentation by Tranquil IT, last compiled on 2024-09-20.

Click here for a PDF version of the complete documentation.

WAPT is a software and configuration deployment tool that may be compared to Microsoft SCCM (System Center Configuration Management) (now called MECM (Microsoft Endpoint Configuration Management)), Ivanti UIM (Unified Endpoint Manager), IBM Bigfix, Tanium, OPSI, PDQDeploy, or Matrix42. WAPT exists in two flavors, *WAPT Discovery and WAPT Enterprise*.

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.
- Reduce your need for support by your IT teams, whose reaction times are often long because of their workloads.
- 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.
- No longer tolerate hosts operating in *Administrator* mode.
- No longer tolerate users downloading and running software binaries from their home directory.
- Start applying SRPs (Software Restriction Policies), also known as *Applocker* or WDAC (Windows Defender Application Control) to improve application level IT security.
- 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 installed software configured to work well in the context of your Organization and trust that they will work correctly.

- Give *Users* more autonomy to install software safely and reliably.
- Have better working and more predictable professional systems because of standard software configurations.

CHAPTER

ONE

INTRODUCTION TO WAPT

1.1 For what purpose is WAPT useful?

WAPT installs, updates and removes software and configurations on Windows, Linux and macOS 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.

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

WAPT exists in two versions, **Discovery** and **Enterprise**, both proprietary, the **Community** version having been friendly forked to the Opensource Community.

WAPT is very efficient to address **recurrent Firefox or Chrome 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.

1.2 Security Certification from French Cyberdefense Agency ANSSI

Following its first level security certification obtained on 14 February 2018, WAPT has obtained on 15 march 2018 a higher level certification from ANSSI.

1.3 The genesis of WAPT

1.3.1 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 tools (*OCSInventory* or *WPKG*) can broadcast software but do 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 problems are either too expensive or too inefficient, and they are in every case too complex.

Fig. 1: Security Visa from ANSSI dated 14th of February 2018 for WAPT Enterprise Edition 1.5.0.13

1.3.2 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 should know a scripting language 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 will 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; thereby WAPT local repositories provide signed packages that they can trust to be deployed on their network. Alternatively, they can choose external public repositories providing them the security guarantees that 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 and OU (Organizational Units); it allows to select a large number of hosts to customize their configuration.

CHAPTER

TWO

FUNDAMENTAL PRINCIPLES

2.1 Repository principle

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

Note: Transport protocol used for deploying packages is **HTTPS**.

WAPT packages 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.

Large organizations with remote sites and subsidiaries sometimes require services to be replicated locally to avoid bandwidth congestion (*Edge Computing*).

2.2 Replicated repository

WAPT Enterprise offers the possibility to upgrade remote WAPT 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.

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 remote devices.

With remote repositories, WAPT remains a solution with a low operating cost because high bandwidth **fiber links are not required** 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.
- The WAPT clients connect in priority with the repository that is the closest to them, the local repository.

To learn more about the replicated repositories, visit the documentation on the *replicating a repository*.

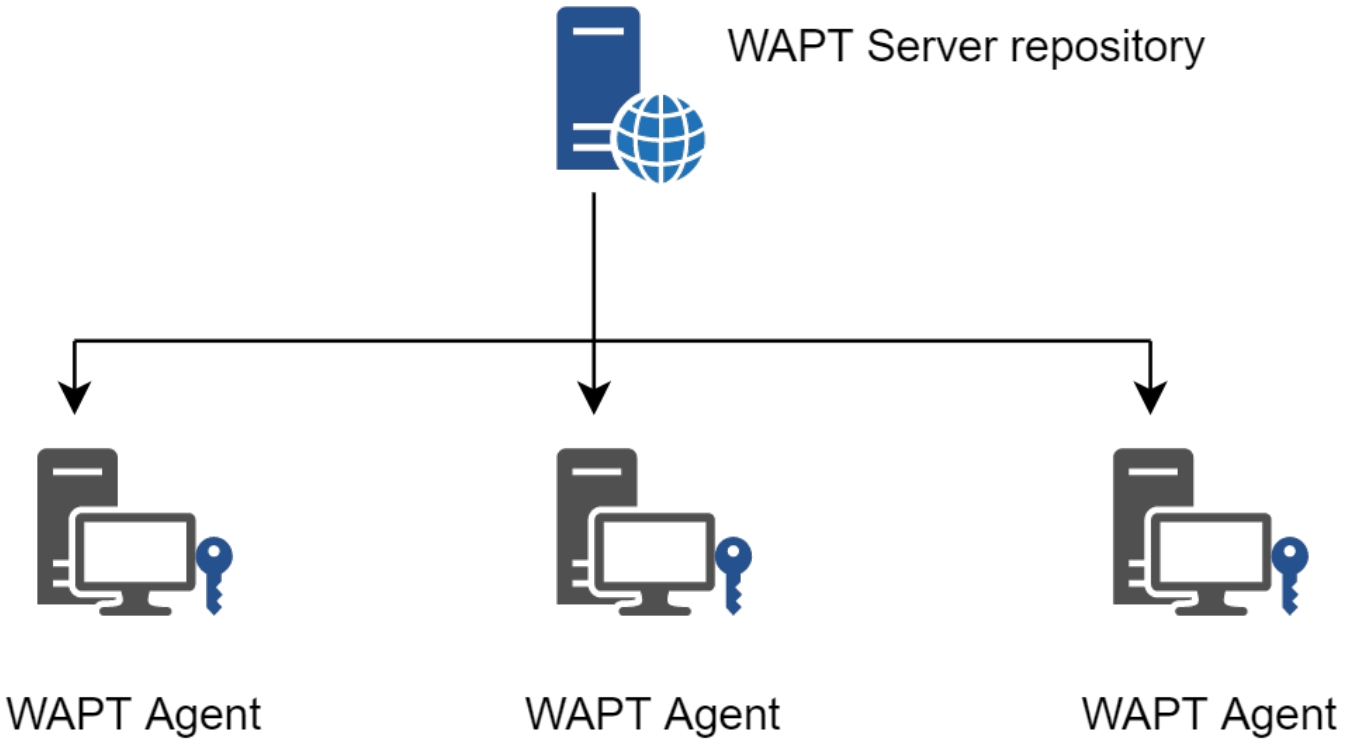


Fig. 1: Replication and multiple repositories

2.3 Packages principle

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:

To learn more about the composition of a WAPT package, visit the documentation on the *structure of a WAPT package*.

2.3.1 Types of WAPT packages

There are 8 types of WAPT packages:

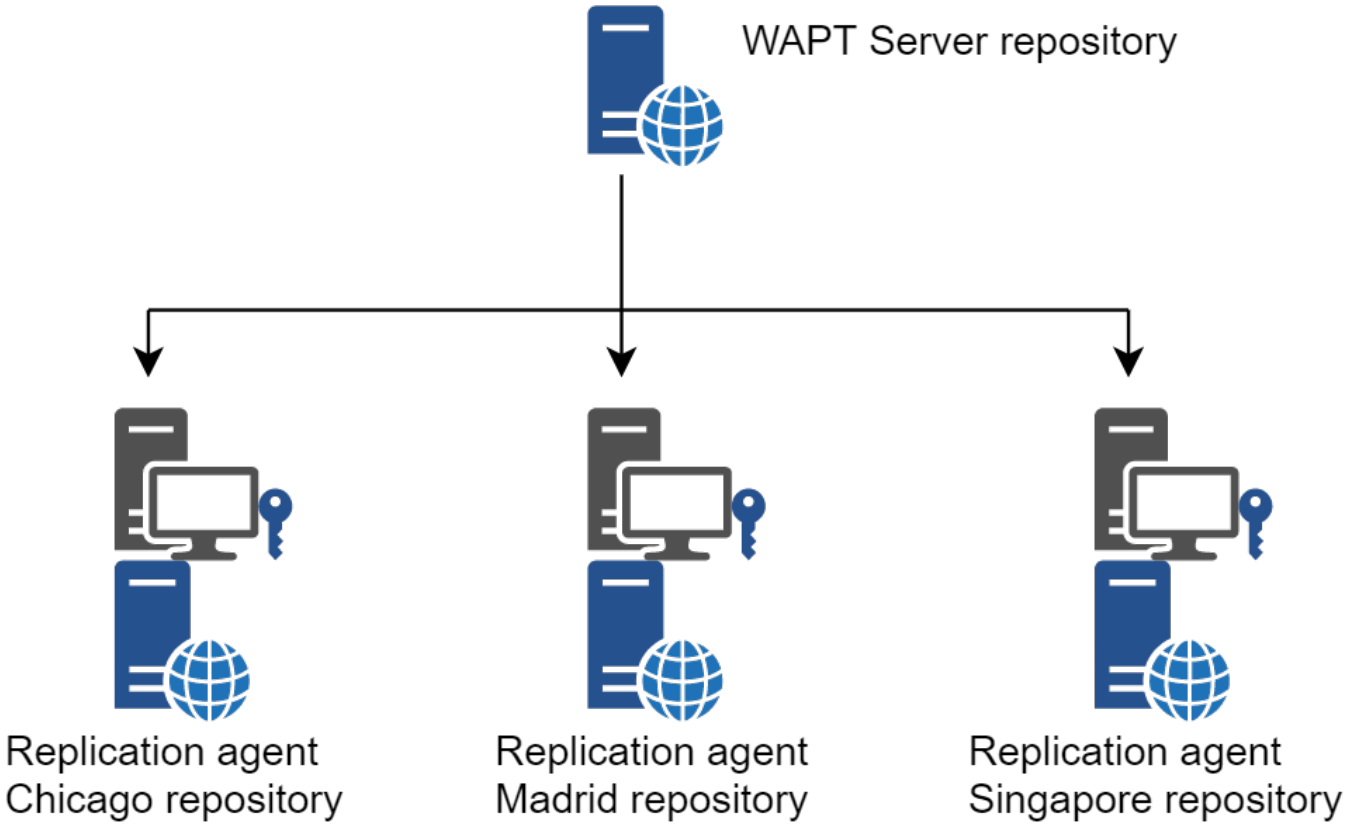


Fig. 2: Replicating WAPT repositories

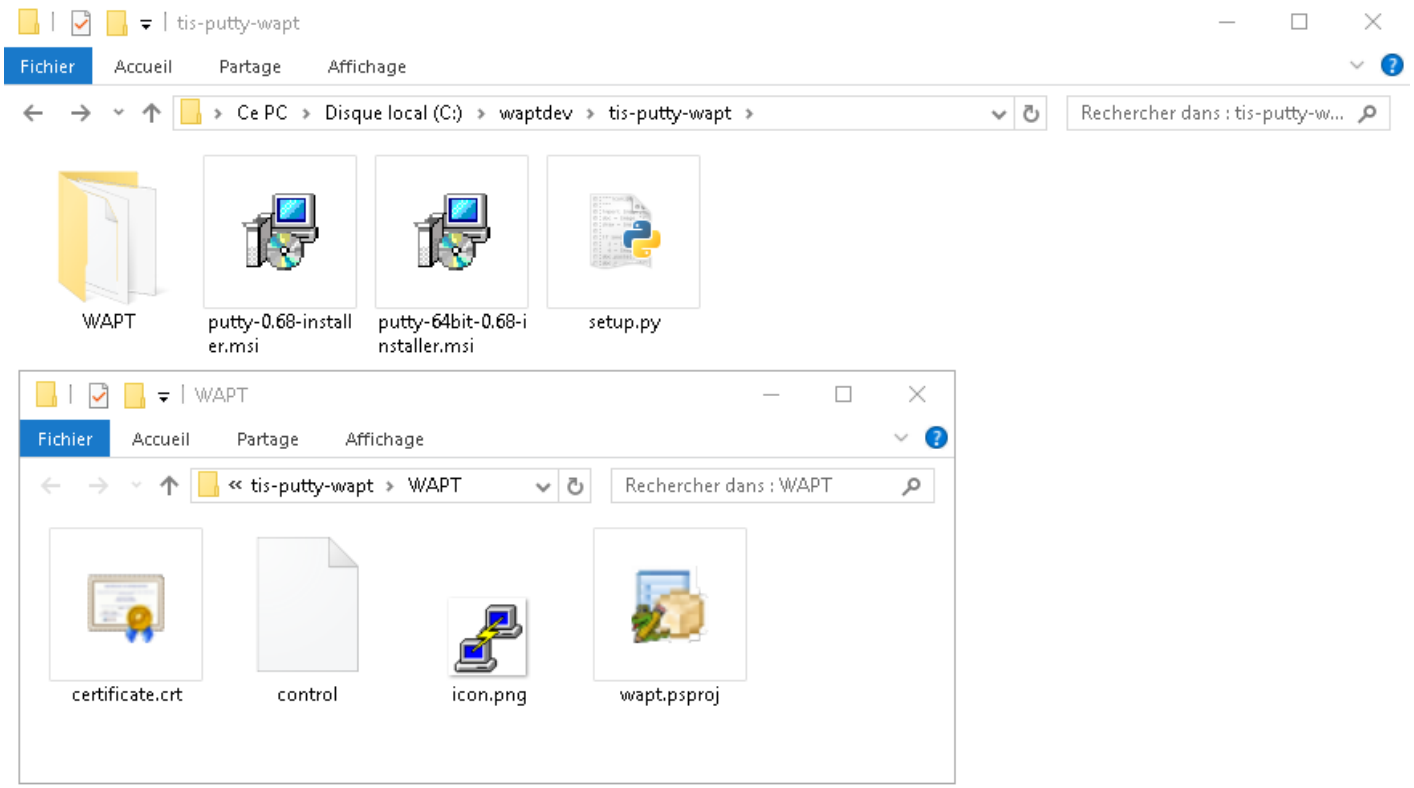


Fig. 3: WAPT package structure shown in Windows Explorer

Fig. 4: 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 of packages.

Each group often correspond to:

- a service in an organization (ex: **accounting**).
- a room, building, etc.

Hint: A host can be a member of several groups.

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 or the *FQDN* of the host.

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

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

unit packages

Unit packages bear the complete name of OU, example: **OU=room1,OU=prod,OU=computers,DC=mydomain,DC=lan**.

By default, each computer looks for the unit packages and then installs the list of associated dependencies.

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

wsus packages

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

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

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

self-service packages

Self-service 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.

Self-service packages are stored in the web directory https://srvwapt.mydomain.lan/wapt/.

profile packages

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 to which the host belongs.
- If a *profile* package has the same name as an 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 corresponding *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 the `wapt-get.ini` configuration file.

---

config packages

config packages are used for modifying the WAPT Agents configuration parameters. This way, it is possible to have a generic WAPT Agent and to customize the WAPT Agents configuration using host profiles. For example, some hosts may need different WAPTWUA rules and some other hosts may need to be set to a **DEV** maturity, etc.

2.4 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 WAPT 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 has been satisfied, the host notifies its status to the WAPT Server. The host status 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.

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 host 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!).

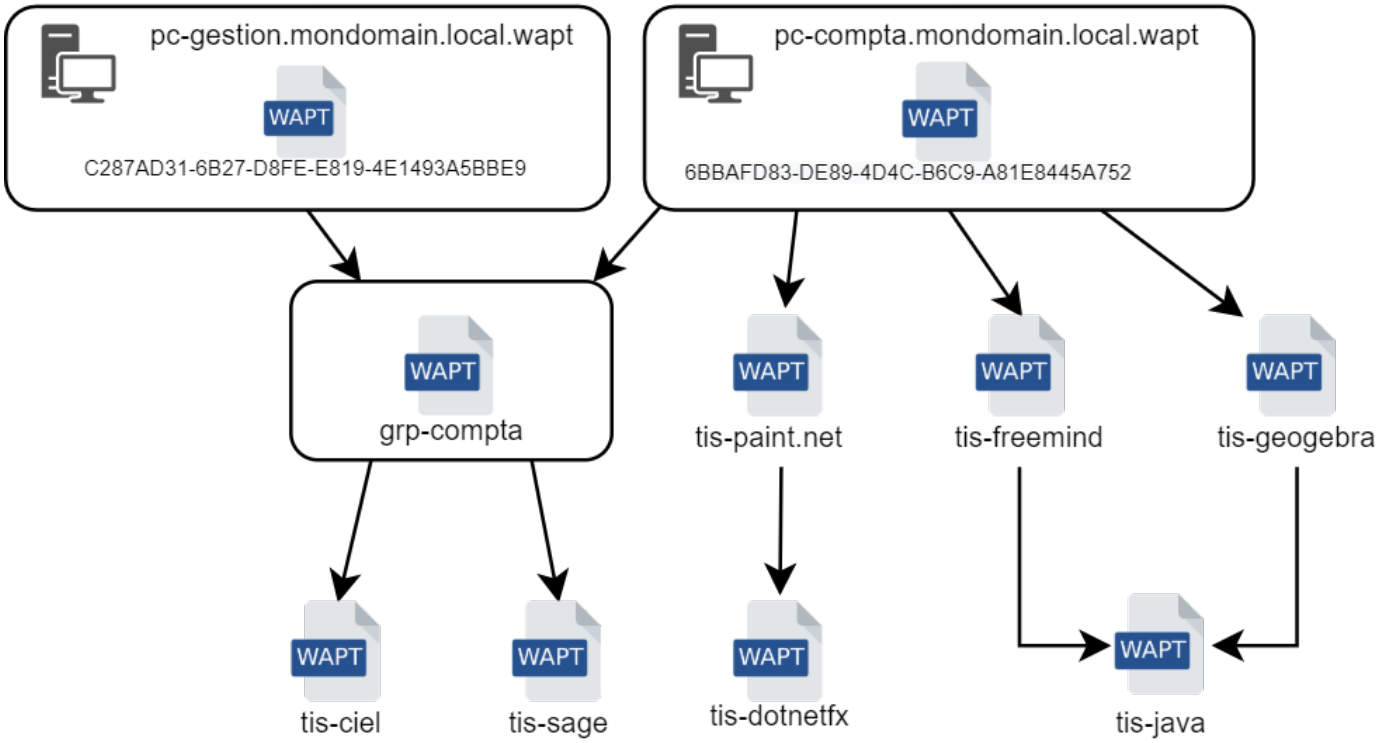


Fig. 5: Conceptual diagram of the dependency mechanism

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 WAPT 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 host.

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`**.

2.5 Private key / Public certificate principle

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.

To work properly, WAPT requires a private key / public certificate 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 certificate** 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 certificates will be stored in the WAPT subdirectory `ssl`. That folder can contain several public certificates.

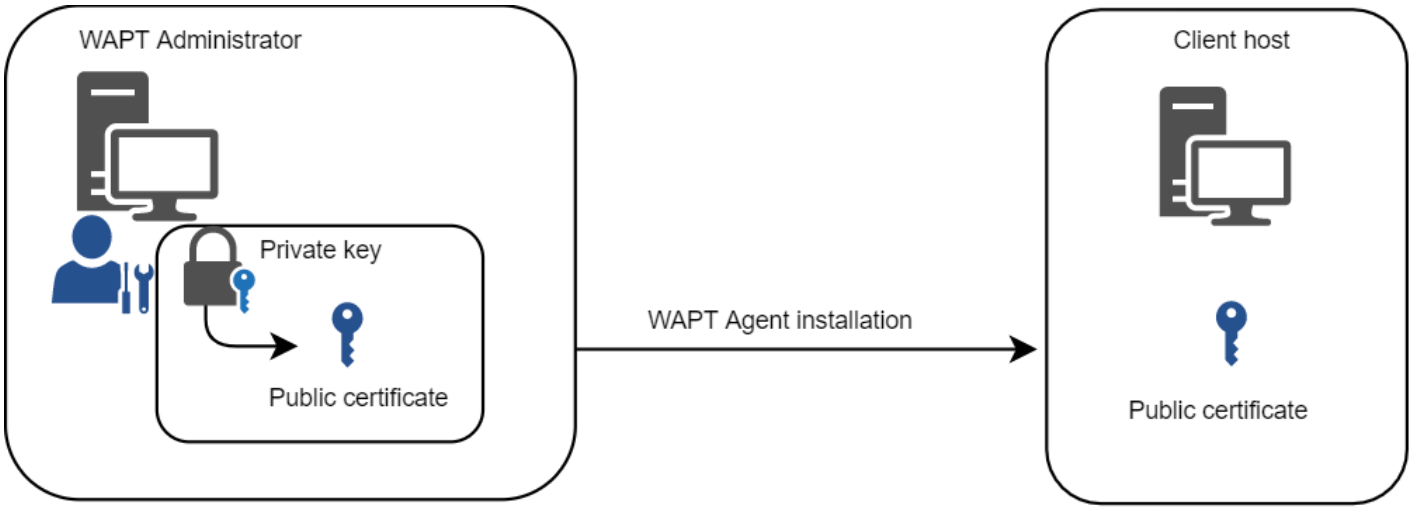


Fig. 6: Private key / public certificate

2.5.1 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 certificates located in `C:\Program Files (x86)\wapt\ssl` on Windows or in `/opt/wapt/ssl` on Linux and macOS, the WAPT Agent will refuse to install the package.

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

2.5.2 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.

The private key is protected with a password by default.

More informations on *generating the Administrator's certificate for signing WAPT packages*.

2.5.3 Differentiating user roles in WAPT

WAPT offers the possibility to differentiate roles based on:

- A PKI (Public Key Infrastructure);
- ACLs (Access Control Lists).

Public Key Infrastructure

Hint: The usage of an existing PKI is possible, the WAPT Console comes with a simple certificate generator.

WAPT works as a CA (Certificate Authority) mode in regards to PKI.

By design, WAPT is capable of generating certificates that may be used as parent keys to generate other public and private child keys.

Therefore the main WAPT Administrator who acts as a may issue certificates for each IT admin so their actions may be identified when they use WAPT.

Child certificates issued from the CA can themselves be configured as:

- Code-signing certificate to allow IT admins to package, sign and deploy WAPT packages containing executable loads (i.e. `setup.py`).
- CA certificate to delegate to other IT admins the right to issue certificates.
- Simple SSL certficate to limit IT admins to only deploying packages containing non executable loads (i.e. configure hosts).

To learn more on generating the Certificate Authority (CA) with WAPT, visit *this documentation*.

Access Control Lists

With WAPT, it is possible to define user access rights using ACL (Access Control Lists).

Each IT technician is identified with his own certificate and rights can therefore be be finely applied on an individual basis.

For exemple, a WAPT Console user may have a `View` right on a host but may not be allowed to click on *Edit hosts*.

To learn more on ACLS in WAPT, visit *this documentation*.

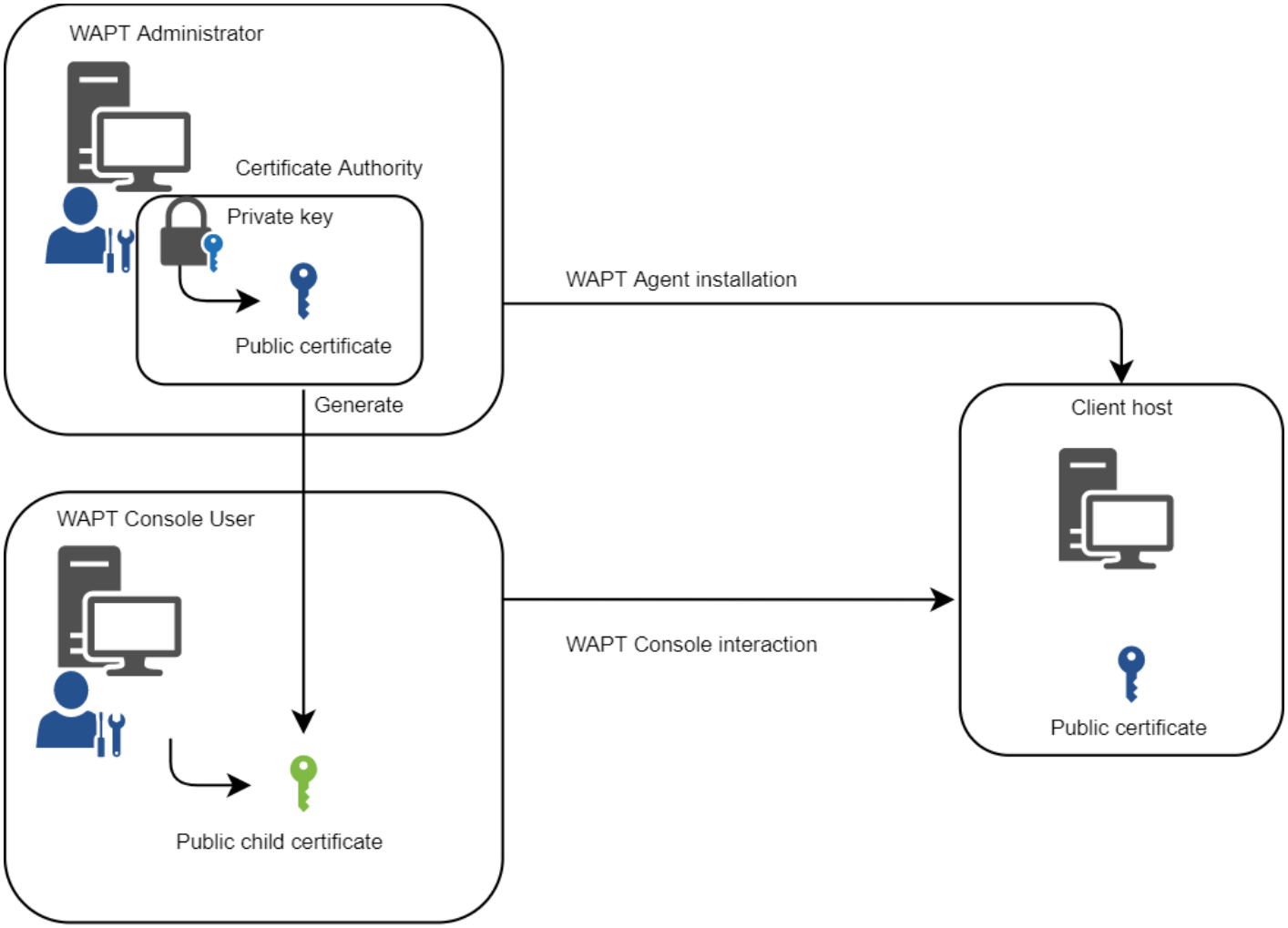


Fig. 7: WAPT user role differentiation

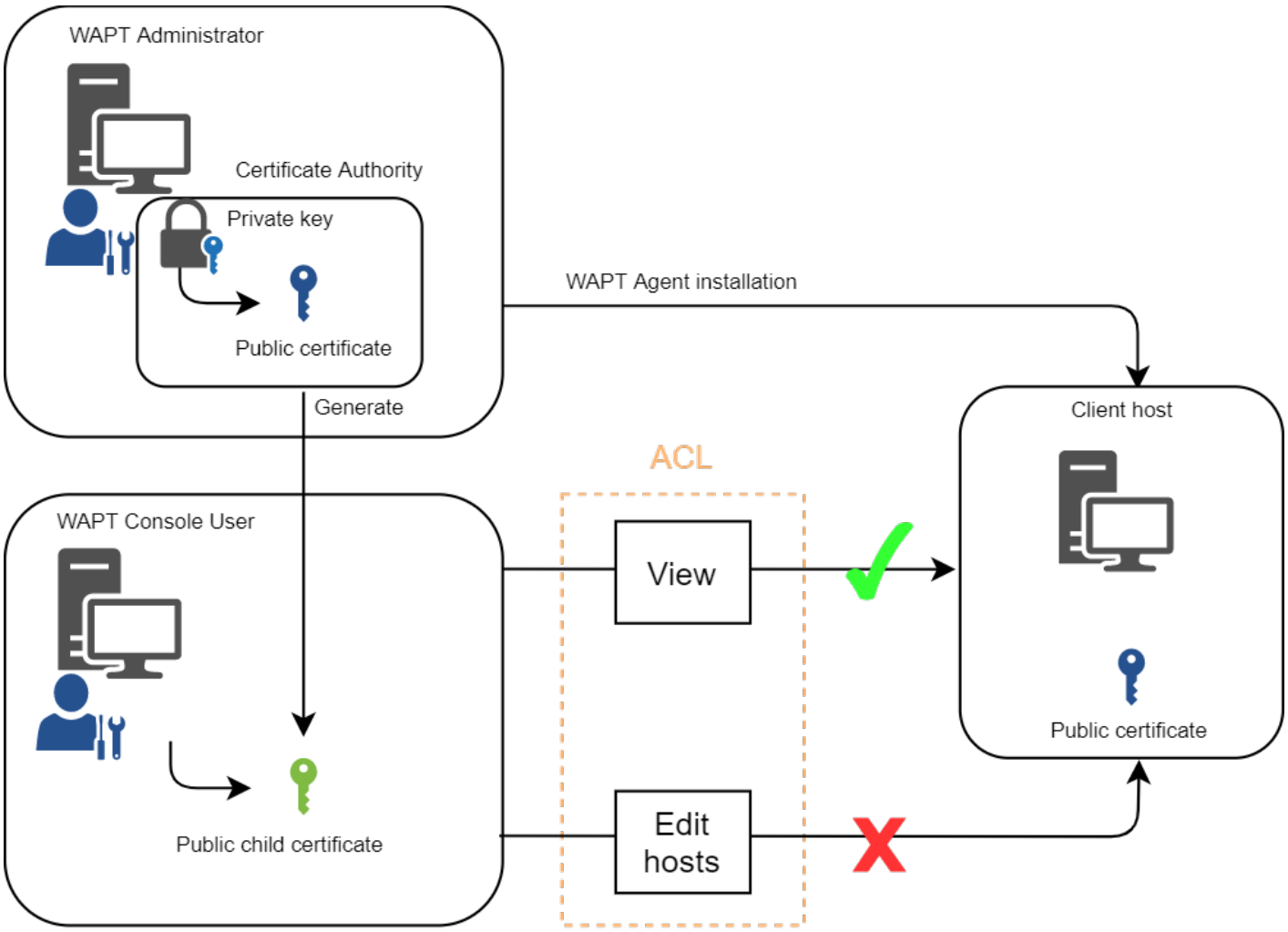


Fig. 8: ACL roles differentiation

CHAPTER

THREE

WAPT MODE OF OPERATION

3.1 Inventory

WAPT keeps a hardware and software inventory of each host.

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

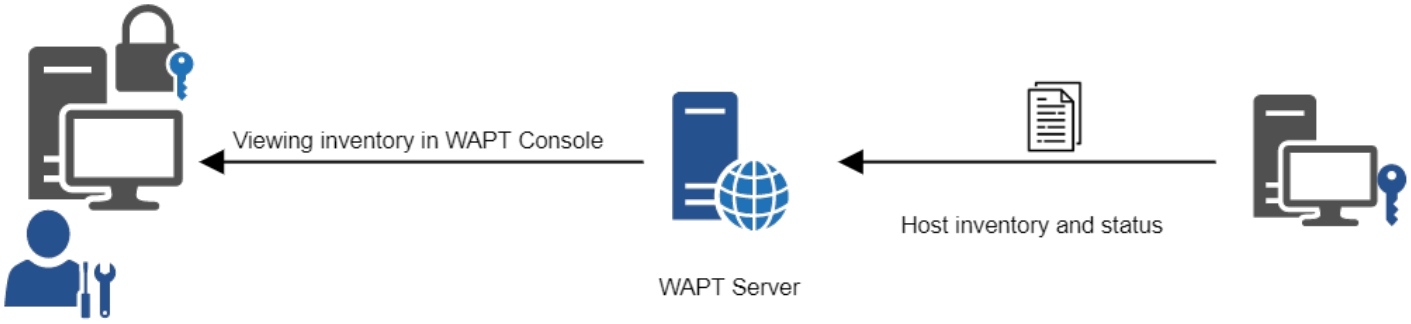


Fig. 1: Inventory feedback mechanism in WAPT

- When first registering with the WAPT Server, the WAPT Agent sends its entire inventory (BIOS, hardware, software) to the WAPT Server.
- When the WAPT Agent updates, the WAPT Agent will report its new inventory status to the WAPT Server.

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

3.2 Information feedback

The WAPT Agents also report back their WAPT package status.

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 WAPT Console.

Status	Reachable	Audit status	Host
ERROR	OK	OK	wsmanage-doc.mydomain.lan
ERROR	DISCO...	WARNI...	client-win11.mydomain.lan

The possibles status of a host in the WAPT Console are:

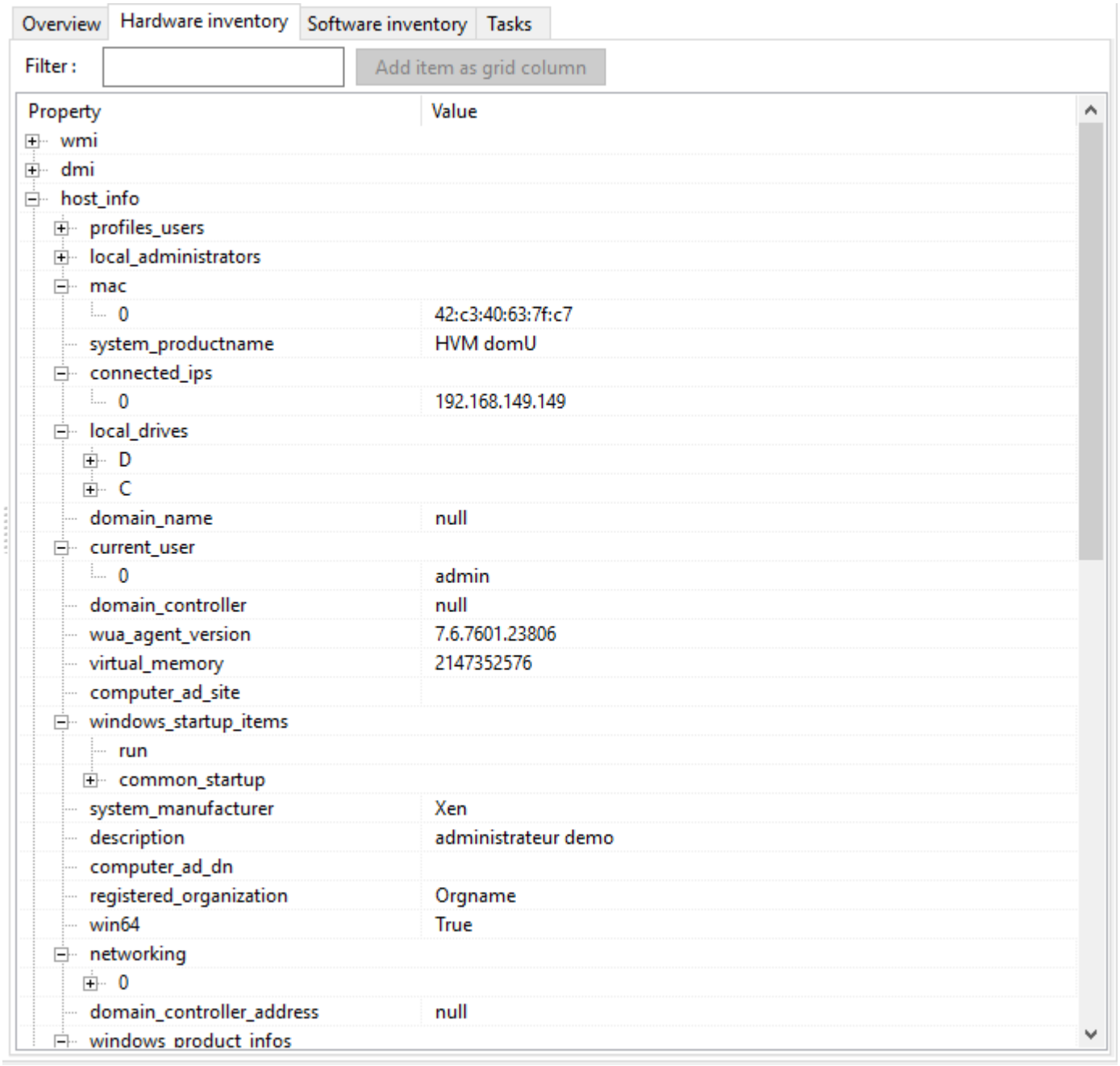


Fig. 2: The inventory in the WAPT Console

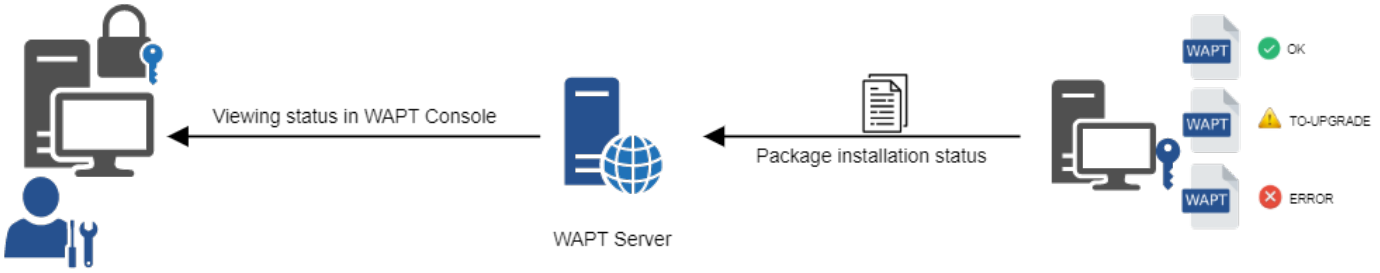


Fig. 3: Flow diagram of the inventory feedback returned to the WAPT Server

Table 1: Available host status in the WAPT Console

Status	Status Icon
OK	
NEED-UPGRADE	
NEED-REMOVE	
ERROR	
NEED-INSTALL	

The *Administrator* can see the packages returned in error in the WAPT 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: WAPT Agents sign their inventory before sending it to the WAPT Server.

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

3.3 WAPT common interactions

3.3.1 update

When an *update* command is launched on a WAPT Agent, 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 the status of the 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.

3.3.2 upgrade

When we launch an *upgrade*, we ask the WAPT Agent to install the packages having a *NEED-UPGRADE* status.

An **`wapt-get update`** **MUST** come before a **`wapt-get upgrade`**, otherwise the Agent will not know whether updates are available.

By default, the WAPT Agent will trigger an **`wapt-get update`**/ **`wapt-get download-upgrade`** at startup; after starting up, **the WAPT Agent will then 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 a **`wapt-get upgrade`** when the computer shuts down. An *Administrator* can force the immediate launch of an **`upgrade`** from the WAPT Console. Alternatively, an end-user may choose to manually launch an *upgrade*. Lastly, a scheduled task may be set up on hosts to launch an *upgrade*.

If the WAPT Server is not reachable when upgrading, the WAPT Agent will still be able to install cached packages. Inventory updates are sent to the WAPT Server when network connectivity returns.

The 5 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.

3.4 WAPT Agent behavior

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 simple steps:

- On triggerring a **`wapt-get update`**, the WAPT Agent downloads *NEED-UPGRADE* or *NEED-INSTALL* packages and stores them in the cache folder.
- On triggerring a **`wapt-get upgrade`**, the WAPT Agent unzips the packages into a temporary folder.
- The `setup.py` content is parsed and stored in the WAPT Agent database located in `C:\Program Files (x86)\wapt\db\waptdb.sqlite`.
- The file `setup.py` is executed and the software is installed from unzipped files.
- In case of success: the downloaded packages and unzipped files are deleted. An **OK** status is returned to the WAPT Server.
- In case of failure: the downloaded packages are kept and the unzipped files are deleted. An **ERROR** status is returned to the WAPT Server.

That behavior is important for understanding the lifecycle of an installed package.

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

- The `setup.py` content is retrieved from WAPT Agent **`sqlite`** database located in `C:\Program Files (x86)\wapt\db\waptdb.sqlite`.
- The WAPT Agent looks up the `UninstallString` in the local database.

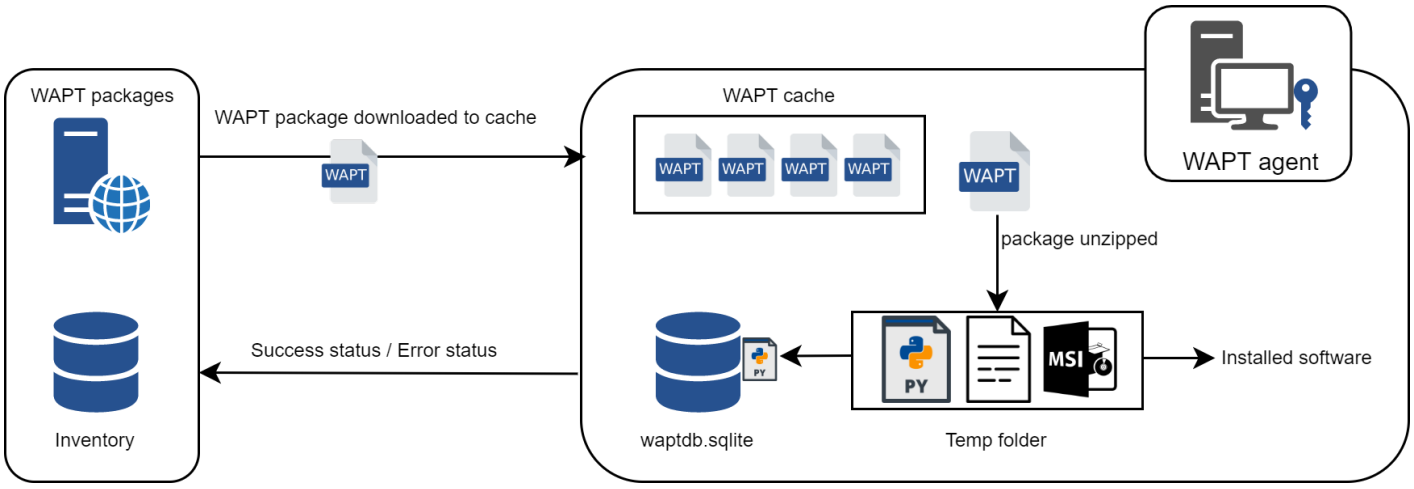


Fig. 4: Flow diagram showing the installation process for a WAPT package

- If defined in the `setup.py` copied into the local database during initial installation of the WAPT package, the **`uninstall()`** function is executed.

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

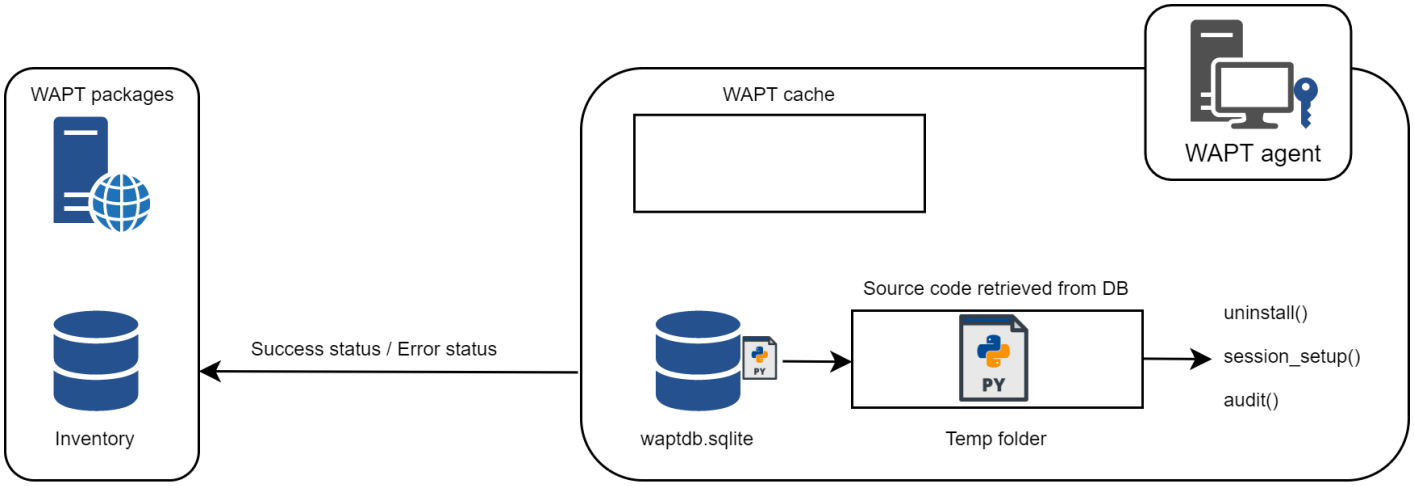


Fig. 5: Flow diagram showing the behavior of WAPT regarding uninstall / session_setup and audit

3.5 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.

Reading the diagram clockwise, a WAPT Administrator will:

- Import packages from an external repository (or create a new package from scratch).
- Test, validate, build and then sign the package.

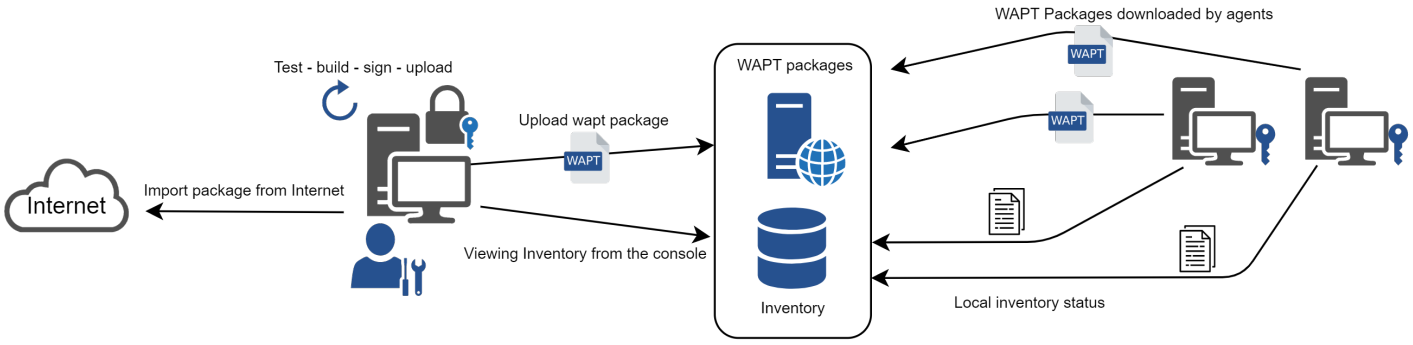


Fig. 6: Flow diagram showing the general operating mode with WAPT

- Upload the package onto the main repository.
- Packages are automatically downloaded by WAPT clients.
- Packages are executed based on the selected method:
  - The *Administrator* forces the **`upgrade`**.
  - The *Administrator* proposes the **`upgrade`** at *User*.
  - A scheduled task launches the upgrade.
  - The *upgrade* is executed when the host shuts down.
  - The *User* chooses the right time for herself (at shutdown or using the *self-service*).
- Inventory information is fed back to the WAPT Server.
- The updated inventory is reported in the WAPT Console.

CHAPTER

FOUR

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.

4.1 Repository role

First, the WAPT Server serves as a web repository.

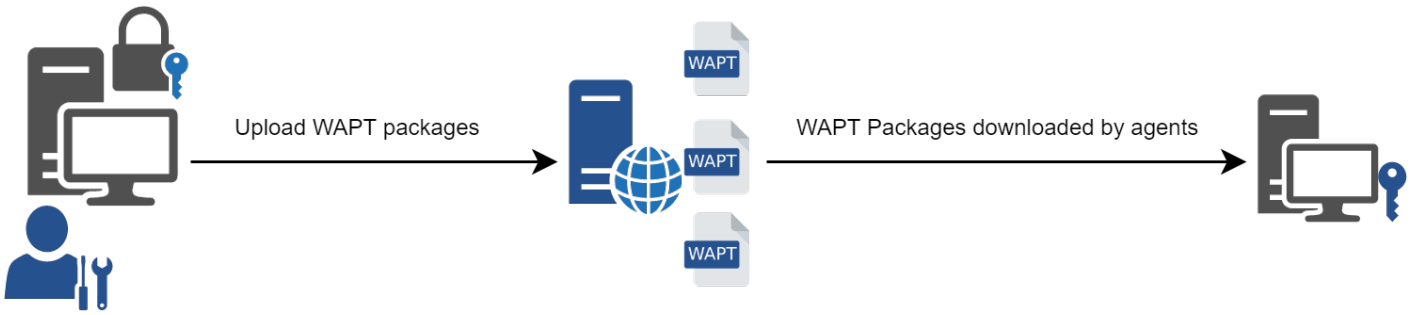


Fig. 1: Flow diagram of the WAPT repository mechanism

- 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.
- The *host* packages are stored in a directory that is not accessible by default (https://srvwapt.mydomain.lan/wapt/wapt-host/).

4.2 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 WAPT inventory Server fails, the inventory will recover by itself from inventory status reports received from the deployed WAPT Agents.

In the **Discovery** version of WAPT, access to inventory data is only possible through the WAPT Console.

WAPT **Enterprise** comes with *reporting* capabilities. In parallel, it is possible to push WAPT inventory to *GLPI* ITSM tool.

4.3 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.

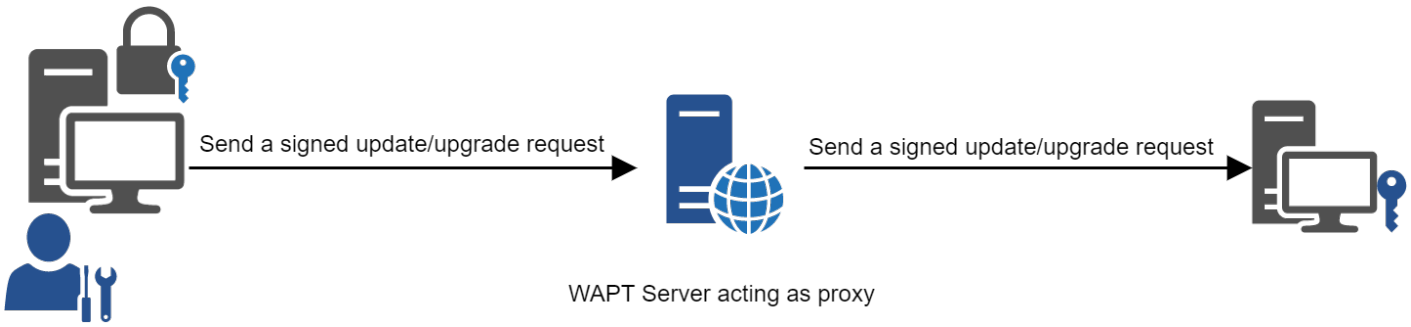


Fig. 2: Flow diagram of the WAPT proxy mechanism

Note: Every action triggered on a WAPT Agent from the WAPT Server are signed with a 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 the documenation on *signing actions relayed to the WAPT Agents*.

CHAPTER

FIVE

WAPT LANGUAGE AND DEVELOPMENT ENVIRONMENT

WAPT is built using the Python language.

> **Attention: With WAPT 2.0, the WAPT internals have switched to python3**.
>
> **WAPT packages MUST NOW follow the new python3 syntax**.
>
> Refer to *this documentation* to help you identify potential problems when switching your existing packages from Python2 to Python3.

Any RAD (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`** for developing WAPT packages for Windows and **`vscode`** for developing WAPT packages for macOS and Linux.

5.1 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).
- And more.

5.2 The power of WAPT

Functions most commonly used with WAPT have been simplified and integrated within libraries called *Setuphelpers*.

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

- **That which was complicated is made simple**.
- **That which was simple is made trivial**.

CHAPTER

SIX

WAPT EDITIONS AND VERSIONS HISTORY

6.1 Currently supported versions

Table 1: Current Software Lifecycle

	February 2023	June 2023	December 2023	June 2024	Second semester of 2024
3.0 Entreprise Discovery					Release 3.0 (To Be Defined)
2.4 Entreprise Discovery		Release 2.4	Security and bugfix maintenance	Security maintenance only	End Of Life
2.3 Entreprise Discovery	Release 2.3	Security and bugfix maintenance	Security maintenance only	End Of Life	

6.2 Not supported versions anymore

Table 2: Old Software Lifecycle

	24 Jan 2020	30 Mar 2021	30 Oct 2021	15 Mar 2022	30 Apr 2022	30 Jun 2022	10 Jan 2023	13 May 2023
2.2 Entreprise Discovery				Release 2.2	Security and bugfix maintenance	Security and bugfix maintenance	Security maintenance only	End Of Life
2.1 Entreprise			Release 2.1	Security and bugfix maintenance	Security and bugfix maintenance	Security maintenance only	End Of Life	
2.0 Entreprise		Release 2.0	Security and bugfix maintenance	Security maintenance only	Security maintenance only	End Of Life		
1.8 Entreprise	Release 1.8	Security and bugfix maintenance	Security maintenance only	Security maintenance only	Security maintenance only	End Of Life		
1.8 Community	Release 1.8	Security and bugfix maintenance	Security maintenance only	Security maintenance only	End Of Support by Tranquil IT, Community support only after[1]			

6.3 Summary of operating principles in WAPT

- **WAPT is agent based to allow no inbound open port** in host's firewalls that initiate a secured bi-directional websocket with the WAPT Server for allowing real-time reporting and actions.
- WAPT works with Trusted Data Gateways using simple task scheduling.
- WAPT works on the principle of smoothly pulling updates and then applying upgrades at a convenient time (works with low / intermittent bandwidth, high latency, high jitter networks).
- WAPT does not require an Active Directory to work (works with Windows Home edition too); however, WAPT will show the host in its Active Directory tree if the host is joined to an AD.
- Methods for deploying WAPT Agent:
  1. Using a GPO (Group Policy Object) or an Ansible script.
  2. Manually after having downloaded the WAPT 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.

[1] WAPT 1.8.2 Community is supported by Tranquil IT until 2022-04-30. After this date, the support will be done by the community only.

- Upgrades may be triggered:
  1. Upon shutdown of the host, this is the standard mode.
  2. By an authorized WAPT Administrator in an emergency (ex: patching critical vulnerabilities running in the wild).
  3. By the user herself at a time she chooses (ex: 24/7 nursing cart unused during breaks 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 asymmetric cryptography.
  2. Authentication of hosts against the WAPT Server using symmetric cryptography on registering.
  3. Confidentiality of the WAPT Server using WAPT deployed client certificates.
  4. Using of ACL to define what an administrator is allowed to view or what actions he is allowed to perform according to his certificate.

6.4 Current feature list as of 2024-09-20

Attention: You may find on the Internet the mention of a GPLv3 **Community** version of WAPT that has been maintained and supported by Tranquil IT up to version 1.8.2, or up to approximately July 2021.

The **Community** version of WAPT has been friendly forked. **Tranquil IT provides no longer any support, nor any maintenance, either free or paid on WAPT =< 1.8.2**. Support and maintenance may be obtained from the operators of the fork at their rates and conditions.

Tranquil IT is the sole author and the full copyright owner of WAPT 1.8.2 and will require from maintainers of friendly forks that they refrain from using the name *WAPT* as the WAPT brand is trademarked and protected by the French INPI (Institut National de la Propriété Intellectuelle).

Table 3: Comparison of features between WAPT versions as of 2024-09-20

Feature	Enterprise	Discovery
Deploy, update and remove software on hosts	✅	✅
Maintenance and support (check footnote for conditions)	Tranquil IT staff[5]	Tranquil IT forum[7]
Licensed under	Proprietary	Proprietary
Limits on number of devices	Depending on the number of devices in your contract	300
Version of Python used in code and WAPT packages	3+ (current)	3+ (current)
Deploy and update **configurations in SYSTEM context**	✅	✅
Deploy and update **configurations in USER context**	✅	✅

continues on next page

Table 3 – continued from previous page

Feature	Enterprise	Discovery
Get a **comprehensive inventory** of hardware, software and applied WAPT packages	✅	✅
Benefit from the **differentiated self-service** (authorized users may install authorized software from authorized WAPT package stores)	✅	❌
Benefit from **simplified Windows Updates** that work better than a standard WSUS (only the required KBs are downloaded 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** for *Edge Computing* scenarii	✅	❌
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	✅	✅
Benefit from **hundreds of Helpers** for simplifying software packaging	✅[3]	✅
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 **reports that you need for your daily sysadmin work or that your organization requires for budgeting decisions**	✅	❌
Authenticate your WAPT *Administrators* against **Active Directory or LDAP**, or their sets of certificates	✅	❌
Benefit from differentiated roles between *Package Developers* and *Package Deployers* so you can **delegate your WAPT powers to the most adequate people** (packagers know security implications, deployers know user needs)	✅	❌

continues on next page

Table 3 – continued from previous page

Feature	Enterprise	Discovery
Benefit from multi-tenant, multi-client mode with ACLs for MSPs (Managed Service Providers) or large multi-departmental or international organizations using an internal, easy to use PKI based mechanism for allowed perimeter	✅	❌
Integration with Mesh Central for simple *screen-sharing* for user support	✅	❌
Continued support for Windows XP in WAPT for factory machine tools, Hospital medical equipment, expensive and hard to replace research instruments, etc	✅[6]	❌
Update packages directly within the WAPT Console with `update_package` function	✅	❌
Integrate WAPT inventory with popular GLPI ITSM (IT Service Management) tool	✅	❌
WADS : operating system image deployment tool integrated within WAPT	✅	❌
Check package with www.virustotal.com	✅	✅[8]
Verified and approved by internationally recognized cybersecurity agency ANSSI  , **WAPT is the only deployment software in the world with this level of certification**	✅	❌
Remote restart and shutdown of client computers	✅	❌
Send html formatted message to connected users	✅	❌
Deploy WAPT configuration packages to easily change the configuration of remote WAPT Agents	✅	❌
Filter newer versions of public WAPT packages directly from the local repository	✅	❌
Support for macOS WAPT Agents	✅	❌
Access to ready-to-deploy WAPT packages or recipes for licensed business software (common business software for industry, medical, office, public collectivities, cybersecurity, etc)	✅	❌

[5] A minimal volume of licenses **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. Forum support is provided without warranty nor delay and may be provided by **Enterprise** or

6.5 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 time-line is promised, stay tuned, we are only promising you that we are working very hard to achieve these objectives.

Feature	Enterprise	Discovery
History of actions done via WAPT for a complete reporting of a hosts software maintenance life-cycle	✅	❌
Authentication of WAPT Administrators using cryptographic tokens (ex: smartcards)	✅	❌
Access to ready-to-deploy WAPT package extensions for simplifying desktop armoring using Applocker or equivalent	✅	❌

6.6 Main functional benefits of the Enterprise version of WAPT

WAPT **Discovery** is designed to let you try WAPT at no cost on a limited perimeter and with limited high-end features.

With WAPT **Enterprise**, you benefit automatically from the base functions included in WAPT to help you deploy, upgrade and remove software and configurations on your Windows, Linux and MacOS devices, from a central WAPT Console, with many more benefits.

WAPT is a *freemium* model. The **Enterprise** version shares the same code base with the **Discovery** version. An activated **Enterprise** license key turns on the following additional functionalities:

- **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*) and with ACLs rigths.

- **ACLs**.

  ACLs are managed by the *SuperAdmin* to authorize or restrict WAPT *Administrators* to viewing informations or performing actions only on a subset of the devices registered with the WAPT Server.

  The identification and the authentication processes rely either on using Active Directory, LDAP or certificates. The authorizations granted to the Administrators are managed in the WAPT Server database. The perimeter of devices on which the rights are granted is defined by the deployed Administrator's certificate.

  This feature is particularly useful for large multi-national Organizations, central administrations with large regional offices or for MSPs wanting to centralize the management of several clients while allowing their end customers to perform some daily management tasks.

Discovery users not affiliated with Tranquil IT.

[3] The Enterprise version embeds more *SetupHelper* functions than the **Community** and **Discovery** versions.

[6] 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 WAPT Agent will of course be excluded from the target of evaluation in future security certifications.

[8] Only for packages on the Tranquil IT certified WAPT store. To benefit from virustotal for your own packages, the Enterprise version is required.

- **Differentiated self-service**.

  WAPT Enterprise allows you to apply lists of allowed packages to user 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.

  This feature is designed to offer *Users* the feeling of freedom and empowerment that they fear to lose in managed environments while allowing CISO to apply strict security rules using such method as SRP (Software Restriction Policies), also known as *Applocker*.

- **WAPT WUA**.

  WAPT allows to manage the Windows Updates on your Windows endpoints.

  WAPT WUA is designed to just work out of the box, be gentle on your storage and preserve your bandwidth for your productive needs.

- **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 host, a dedicated appliance or Virtual Host.

  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.

- **Integration with GLPI**

  GLPI is a popular ITSM solution for ticketing, incident and asset tracking.

  WAPT can now optionally send a minimum set of useful informations to a GLPI server.

6.7 Targeted use cases of WAPT Enterprise

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.
- That value secured and proven solutions in their IT sourcing.

6.8 Description of services available with a WAPT Enterprise contract

6.8.1 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 **Discovery** version. Advanced functions only available in the **Enterprise** version will no longer be accessible and no action other that deleting hosts from the WAPT Console will be allowed until the host count has passed below 300.

6.8.2 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 in **English** and **French**.

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 custom packages may be obtained with prepaid support tickets.

Up to three individuals in your *Organization* may communicate with our direct support.

Note: For more information, contact Tranquil IT sales team.

6.8.3 Price and preferential access to WAPT training

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

Note: For more information, contact the Tranquil IT sales team.

CHAPTER

SEVEN

QUICKSTART - INSTALLING THE WAPT SERVER

7.1 To read beforehand

- The quickstart guide install WAPT Server on a Windows Server. Installing WAPT on a Linux server is the recommended method, unless you are trialing WAPT and you are not familiar with Linux.
- With Wapt Server Windows version, you don't have some features such as Kerberos authentication nor upload voluminous packages. The nginx performance is way less efficient on Windows so actions will be slower than from a Linux Server. Please consider carefully this informations before installing the WAPT Server on a Linux host.
- The installation of the WAPT Server **MUST** be done using a Local Administrator account on the host and **NOT a Domain Administrator** account.
- `Nginx` is the **ONLY** supported web server with WAPT. **Apache or IIS (with or without WSUS) are NOT supported in WAPT**.
- In case of problems when installing WAPT, visit the *Frequently Asked Questions*.

Danger:

- The WAPT Server **MUST NOT** be installed on a computer with services already listening on port 443 (example WSUS with IIS). Port 443 is used by the WAPT Server and MUST be available to WAPT only.
- The WAPT Server **will NOT run** on a x86 version of Windows. It runs only on a fresh Windows version currently supported by Microsoft. The server component of WAPT works just as well on a win10 client VM or a physical host as it does on a Windows server version.

7.2 Installing WAPT server

- Download and execute as administrator `waptserversetup.exe`.
- Choose the language for the WAPT installer and click on *OK* to go on to the next step.
- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).
- Choose the password for the WAPT Server and click on the *Install* to launch the installation, wait for the installation to complete.

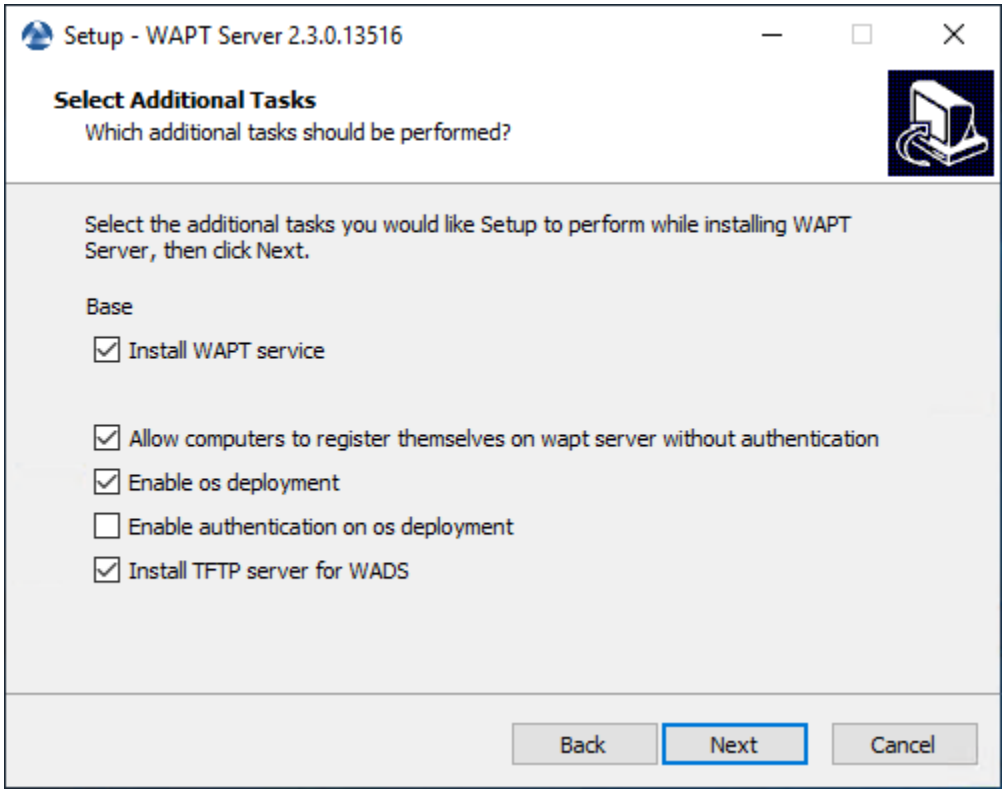
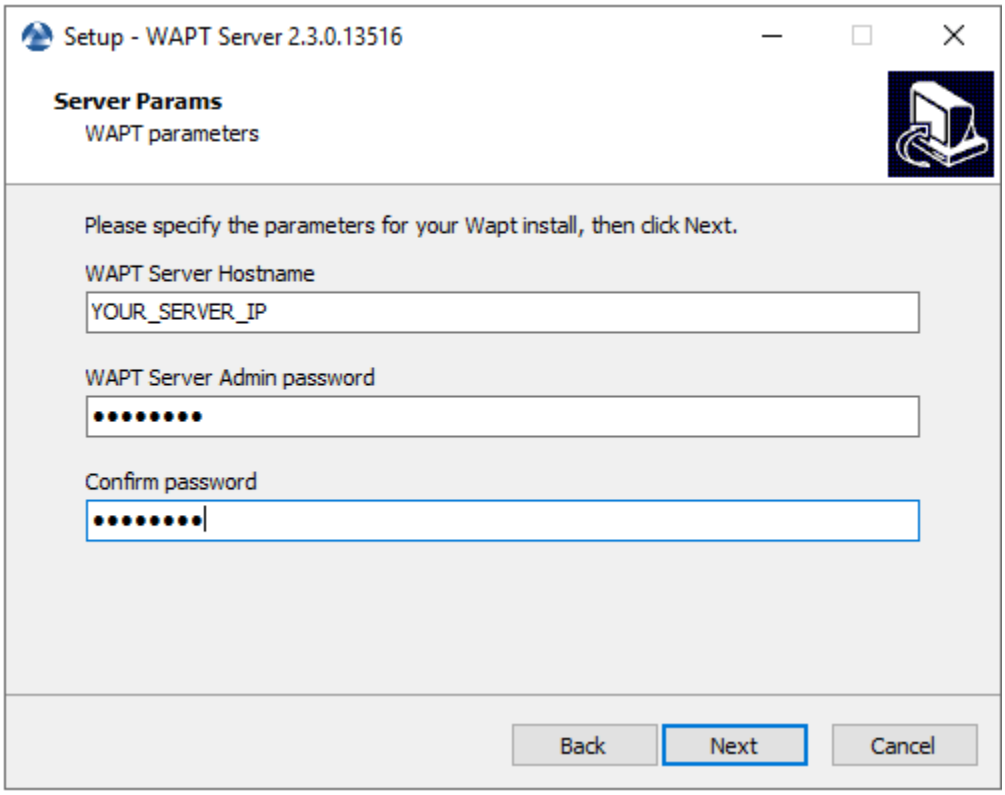


Fig. 1: Choosing the installer options for deploying the WAPT Server



- Click on *Finish* to close the window.

Attention:

- **For security, do not run the WAPT Console or your WAPT package development tool on the WAPT Server.**
- The WAPT Server on Windows **includes the WAPT Agent**. It is not necessary to install the WAPT Agent to manage the WAPT Server on Windows.

Your WAPT Server is now ready. You may now go to the documentation on *Quickstart - Installing the WAPT management Console*.

Hint: If you want to use features such as Kerberos or to deploy voluminous packages, you will have to migrate your WAPT Windows Server to Linux or create a new one from scratch after your tests.

CHAPTER

EIGHT

QUICKSTART - INSTALLING THE WAPT CONSOLE

The WAPT Server having been successfully installed, now we will install the WAPT Console.

- Managing WAPT is done mainly via the WAPT Console installed on the *Administrator*'s workstation.
- It is recommended that the Administrator's computer 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**.

> **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.

To download the `waptsetup.exe` file, point your web browser to your waptserver url https://YOUR_SERVER_IP, then click on the *WAPTSetup* link on the right-hand side of the WAPT Server web page. The WAPT Server home page only provides basic server status information and the download link for the WAPT Console.

8.1 Installing the WAPT Setup on the Administrator's computer

- Start the executable installer as *Local Administrator* on the *Administrator*'s workstation.
- Choose the language for the WAPT installer and click on *OK* to go on to the next step.
- Accept the licence terms and click on *Next* to go to next step.
- Click on *Next* to leave the default WAPT installation folder.
- Click on *Next* with just "Install WAPT service" checked.
- Set up the WAPT Server URL. Can be IP address or DNS Name. Leave the check options as it is.
- Check *Static WAPT Informations* and set:
  - WAPT repository URL: http://YOUR_SERVER_IP/wapt.
  - WAPT Server URL: https://YOUR_SERVER_IP.
- Choose the WAPT repository and the WAPT Server; click *Next*.
- Get a summary of the WAPT Console installation and click *Install* to launch the installation, wait for the installation to complete, then click on *Finish* (leave default options).

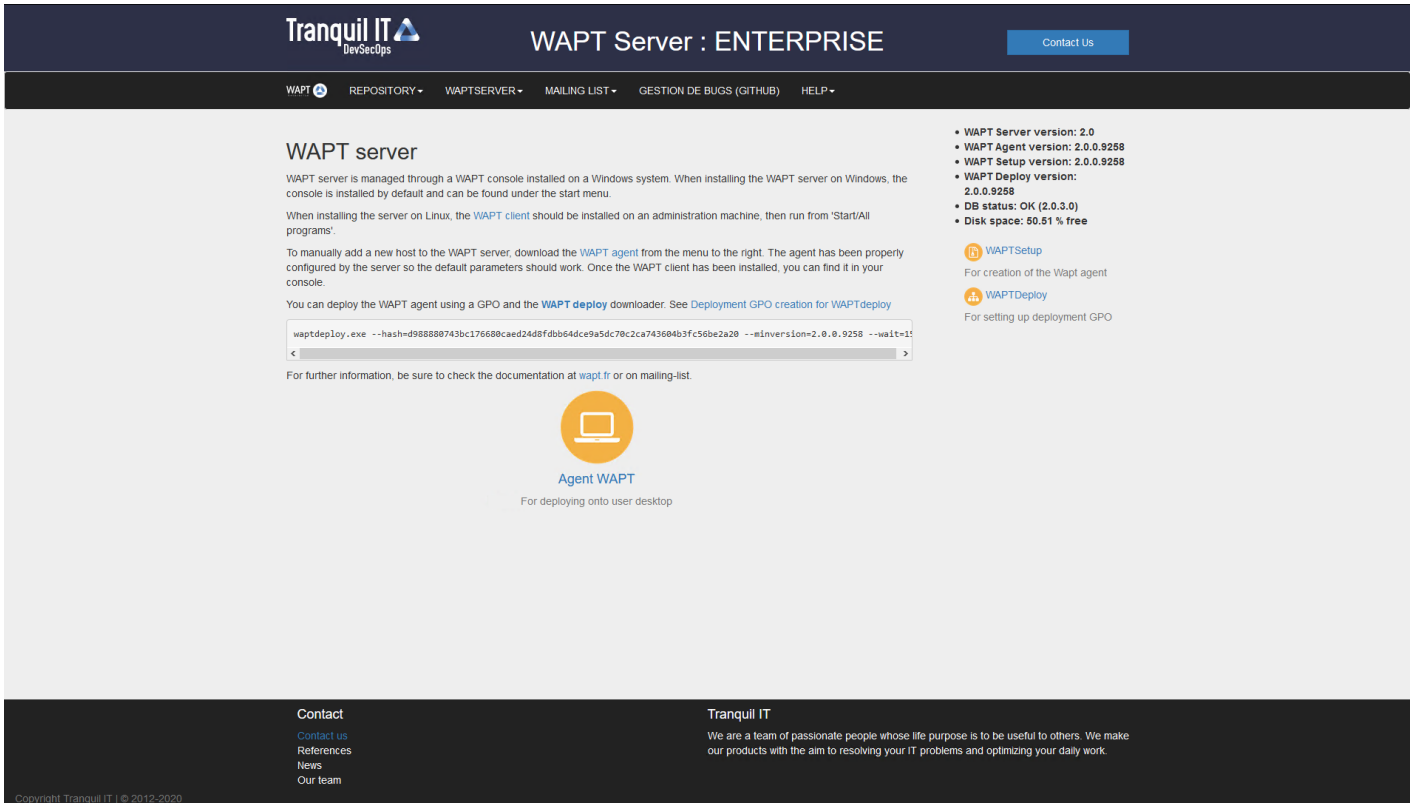


Fig. 1: The WAPT Server interface in a web browser

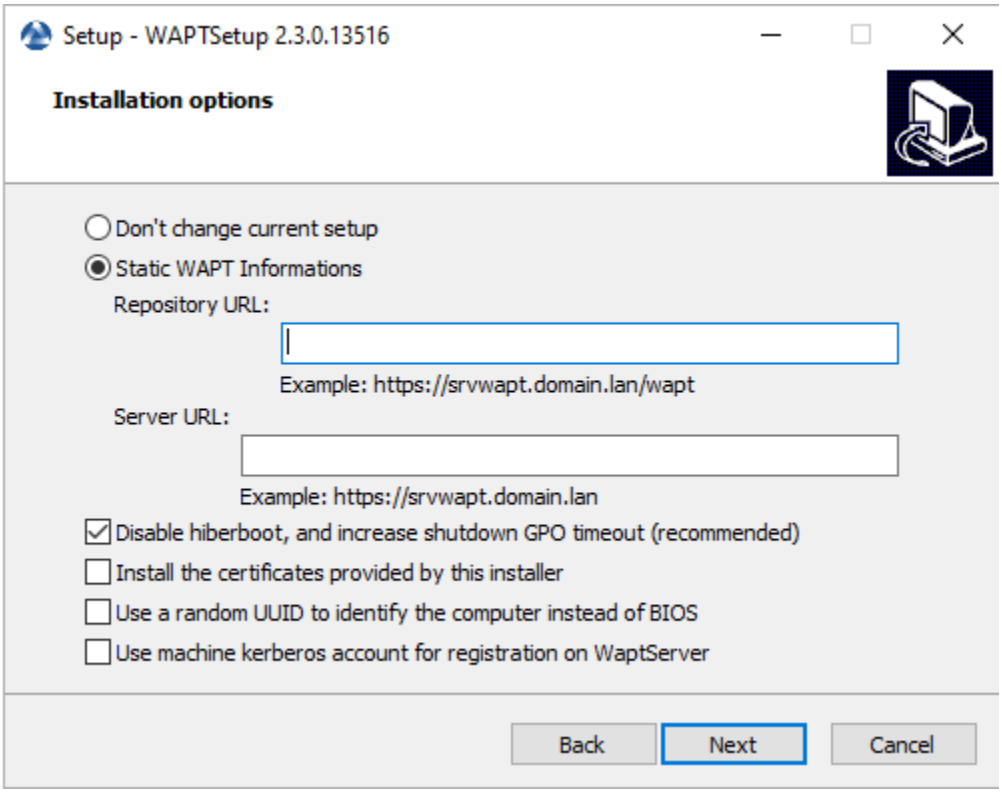


Fig. 2: Choosing the WAPT repository and the WAPT Server

8.1.1 Starting the WAPT Console

- Launch the WAPT Console:
  - By looking for the binary.

    `C:\Program Files (x86)\wapt\waptconsole.exe`
  - Or using the *Start* Menu.
- Log into the WAPT Console with the *SuperAdmin* login and password.

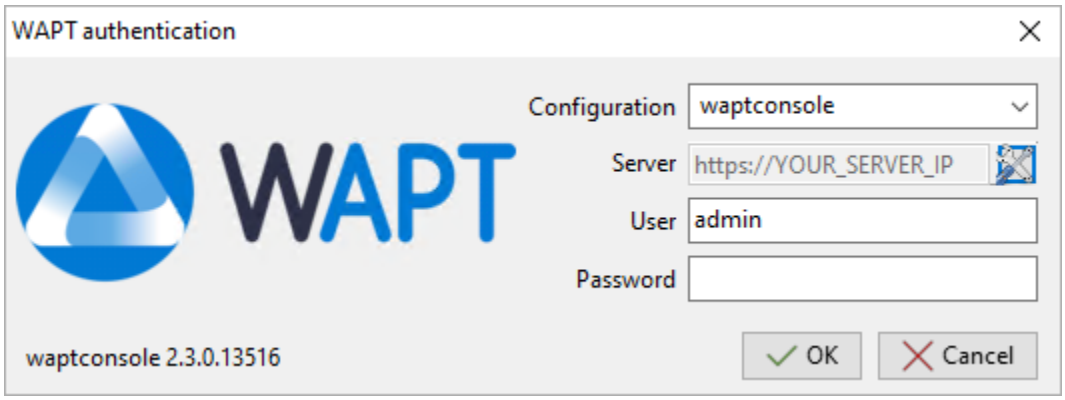


Fig. 3: The WAPT Console authentication window

On first start, you **MUST** start the WAPT Console with elevated privileges. *Right-click on the WAPT Console binary → Start as Local Administrator*.

Note: The recommended size for using the WAPT console is 1920x1080 and the minimum size is 1280x1024.

8.2 Generating the Administrator's certificate for signing WAPT packages

- A message will appear indicating that no personal certificate has been defined.

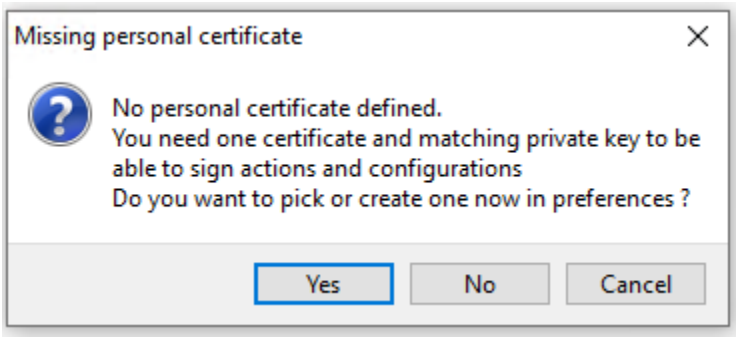


Fig. 4: WAPT personal certificate not found in the WAPT Console

- Select *Yes*

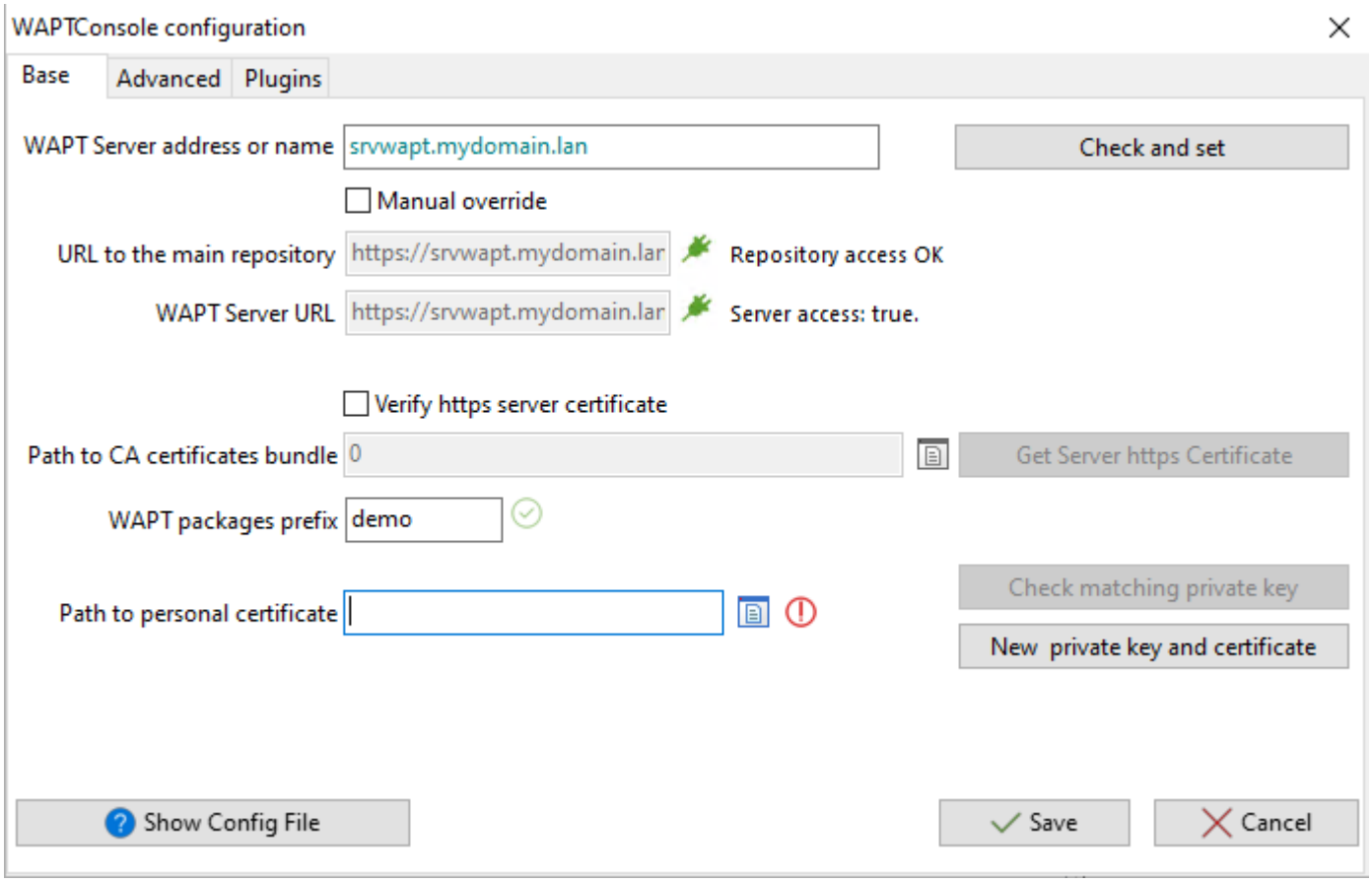


Fig. 5: Window for the basic configuration of the WAPT Console

- Click on *New private key and certicate* and see *create your certificate*.
- In the example, the name of the public certificate signed with the private key is `wapt-private.crt`. This certificate is used to validate the signature of packages before installation. If the public certificate used on the WAPT Console is not derived from the private key used for generating the WAPT Agents, the WAPT Console will not see the WAPT Agents and you will not be able to perform any action on any WAPT Agent.
- In the example, the name of the private key is `wapt-private.pem`. It is located by default in the `C:\private` folder of the *Administrator* workstation and is password protected. It will be used along with the certificate to sign packages before uploading them onto the WAPT repository.

> **Danger:** 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 **MUST NOT** be stored on the WAPT Server.

- Fill the informations to create a self-signed certificate.
- Click on *OK* to go on to the next step.
- Click on *Yes* to copy the newly generated certificate in the folder `C:\Program Files (x86)\wapt\ssl` on Windows or `/opt/wapt/ssl` on Linux or macOS. This certificate will be picked up during the compilation of the WAPT Agent and deployed on the client computers.

8.3 Packet prefix definition

- A message will appear indicating that no package prefix has been defined.
- Select *Yes*
- Set your packages prefix on *WAPT packages prefix*

8.4 Activating a WAPT licence

- A message will appear indicating that no licence has been found, click on yes to activate a licence. To activate the licence, use the `licence.lic` file provided by our sales department.

8.5 Creating the WAPT Agent

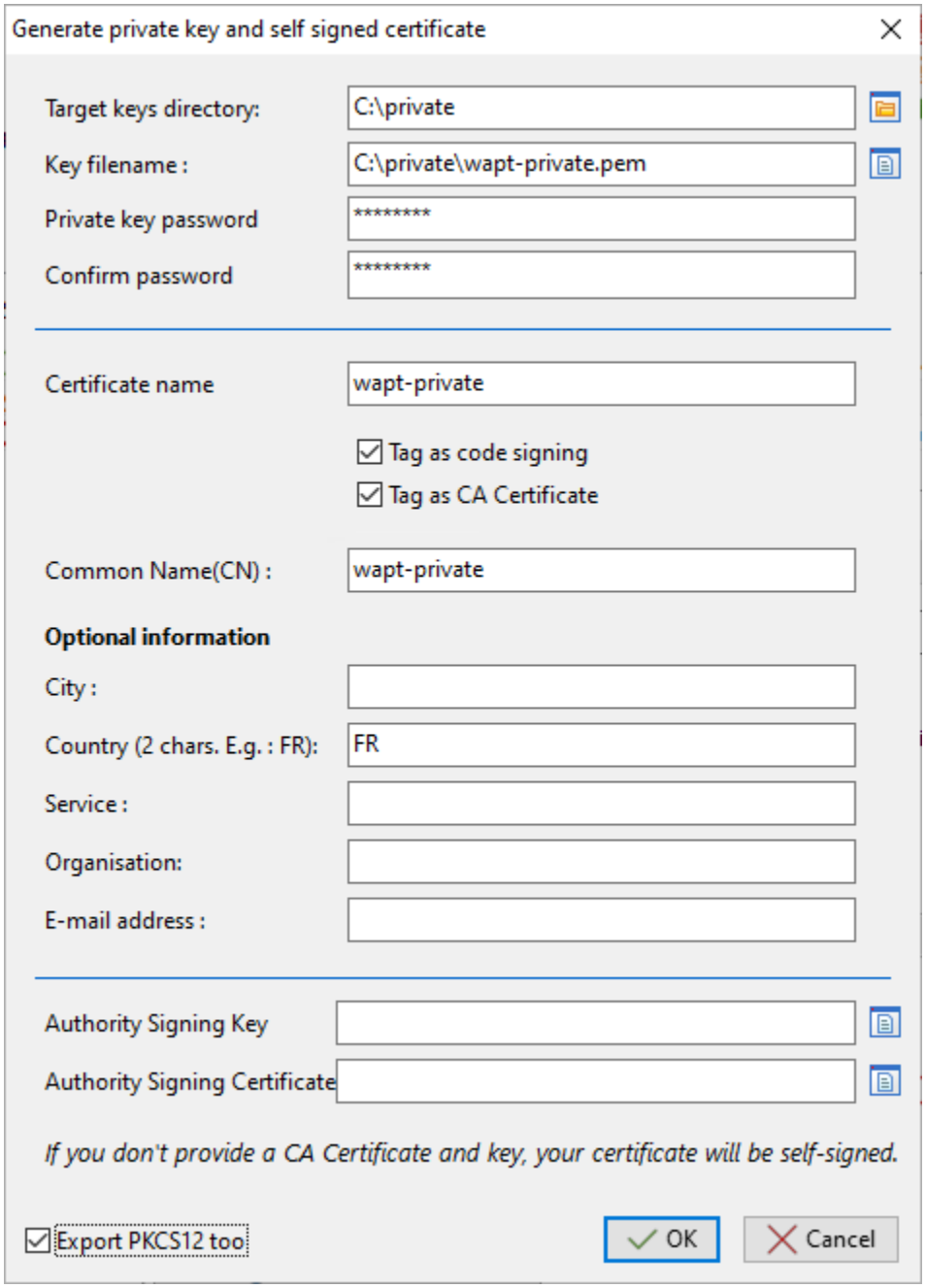
Note: A message may appear indicating that your WAPT Agent version is obsolete or not yet present.

If the *quickstart - administrator's certicate* existing, it is possible to *quickstart - generating new WAPT Agent* by clicking on *Yes*.

Also click on *No* and generate the *administrator's certicate*.

- Fill in the informations that are necessary for the installer.
- 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://YOUR_SERVER_IP/wapt/.


Generate private key and self signed certificate
Target keys directory:
C:\private
Key filename :
C:\private\wapt-private.pem
Private key password
****
Confirm password
****
Certificate name
wapt-private
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 :
Authority Signing Key
Authority Signing Certificate
If you don't provide a CA Certificate and key, your certificate will be self-signed.
Export PKCS12 too
OK
Cancel

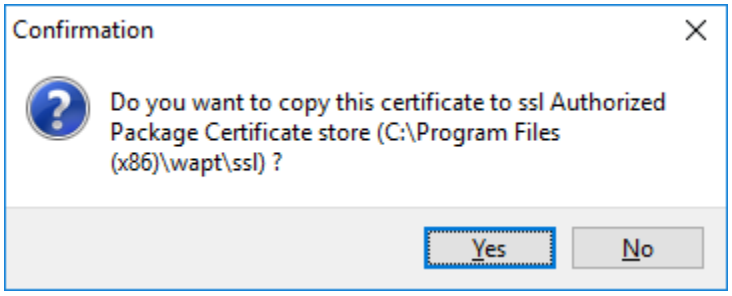


Fig. 6: Dialog box requesting confirmation of the copy of the certificate in the ssl folder in the WAPT Console

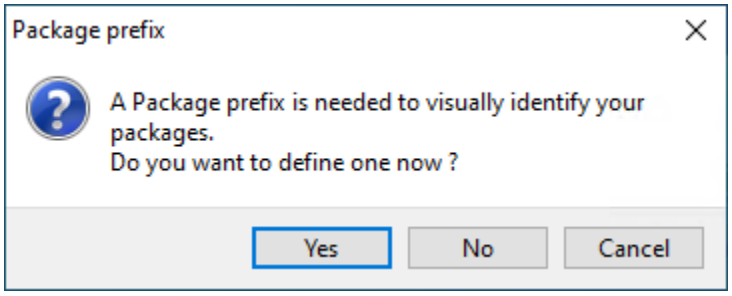


Fig. 7: Dialog box informing that no prefix has been set in the WAPT configuration

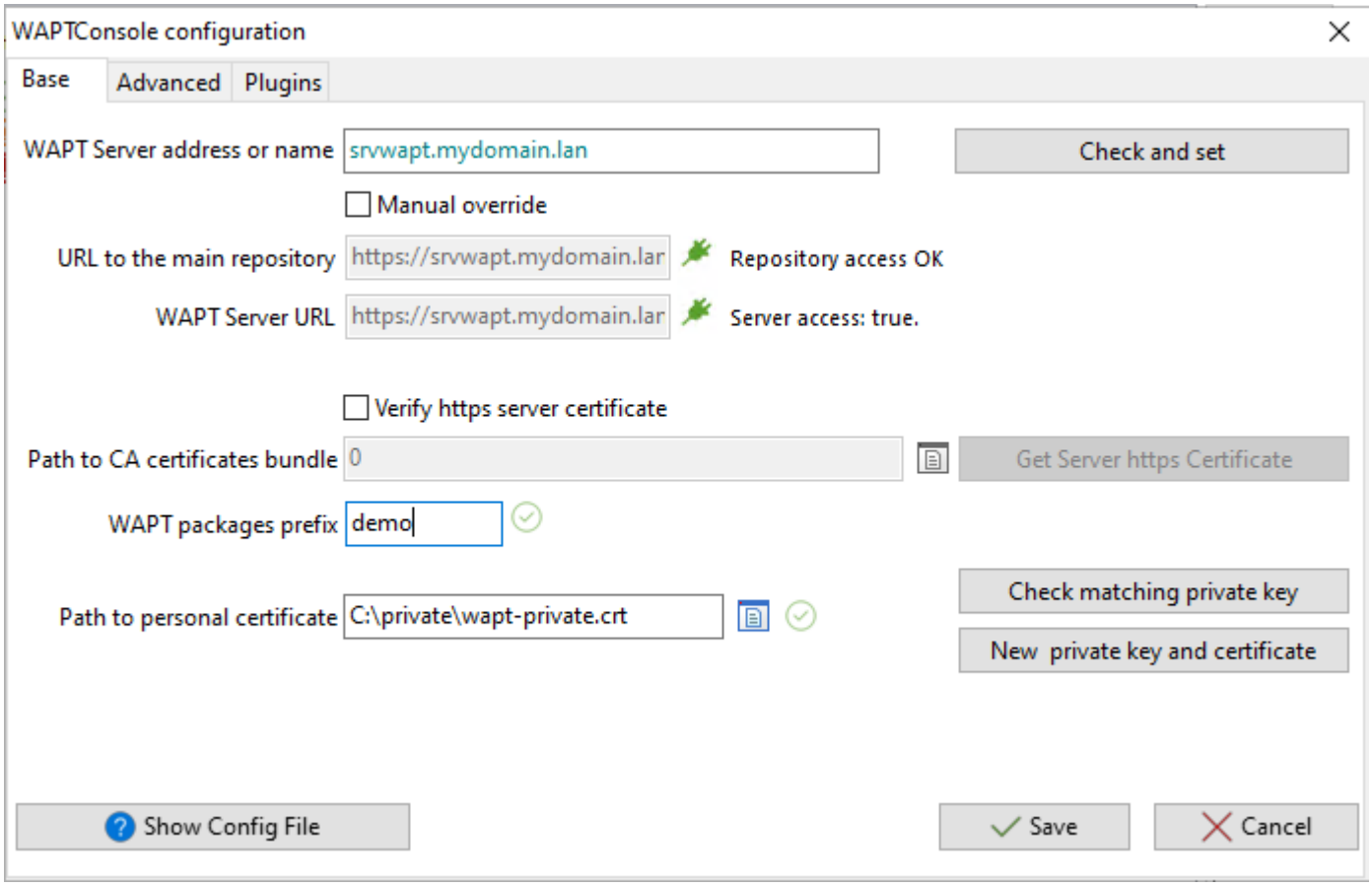


Fig. 8: Window for the basic configuration of the WAPT Console

Fig. 9: Window listing no subscribed WAPT licences in the WAPT Console

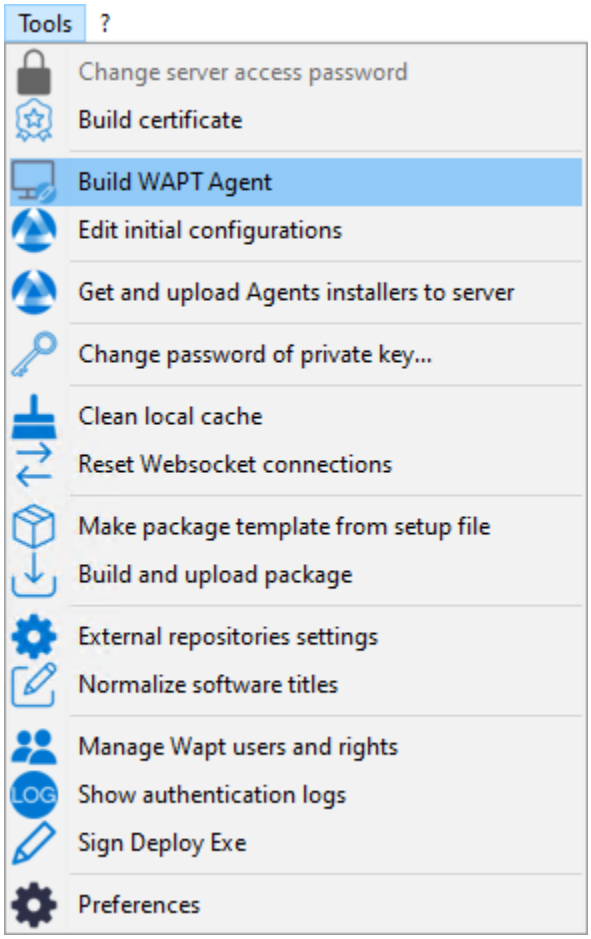


Fig. 10: Generating the WAPT Agent from the WAPT Console

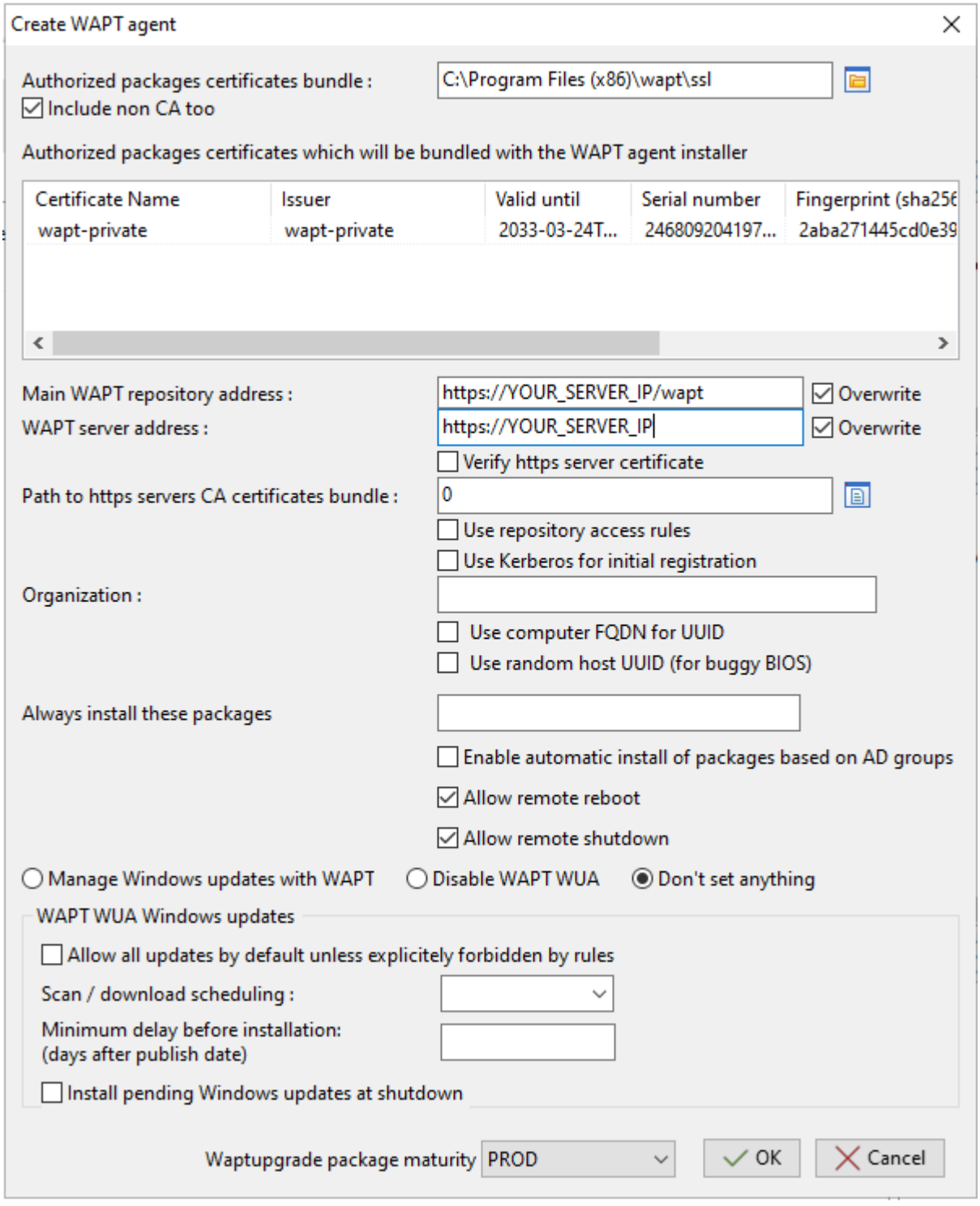


Fig. 11: Filling in the informations on your Organization

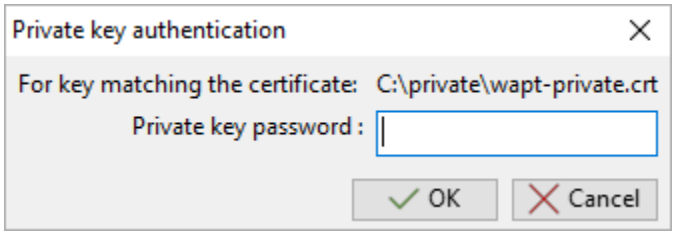


Fig. 12: Providing the password for unlocking the private key

CHAPTER

NINE

QUICKSTART - INSTALLING THE WAPT AGENT

Manually installing the WAPT Agent requires *Local Administrator* rights on the computer.

- Download the WAPT Agent from your WAPT Server then launch the installer. The **waptagent.exe** installer is available at WAPT serveur web home page. The direct download link is for example: https://YOUR_SERVER_IP/wapt/waptagent.exe.
- Choose the language for the WAPT installer and click on *OK* to go on to the next step.
- Accept the licence terms and click on *Next* to go to next step.
- Just click on next until the install button

CHAPTER

TEN

CHECKING WAPT INSTALLATION REQUIREMENTS

10.1 Installation requirements

10.1.1 Naming conventions

You have to take into consideration a few security points in order to extract all possible benefits from WAPT:

- If you are familiar with Linux, we advise you to install WAPT Server directly on Debian or a RedHat based distribution following the security recommendations of French *ANSSI* or the recommendations of your state cyberdefense agency.
- Although the WAPT Server is not designed to be a sensitive asset, we recommend it to be installed on a **dedicated host** (physical or virtual).

Attention: In all steps of the documentation, **you will not use any accent or special characters** 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`.

10.1.2 Hardware recommendations

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

Table 1: Optimal RAM and CPU recommendations for the WAPT Server

Size of the network	CPU	RAM	Server optimization to apply
From 0 to 300 WAPT Agents	2 CPU	2024 Mio	No
From 300 to 1000 WAPT Agents	4 CPU	4096 Mio	Yes
From 1000 to 3000 WAPT Agents	8 CPU	8192 Mio	Yes
From 3000 WAPT Agents onward	16 CPU	16384 Mio	Yes

- 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.
- Finally, we have knowledge of users with WAPT Servers equipped with multiple 10Gbps networking interfaces deploying at full speed massive Catia, National Instruments and Solidworks update packages on their LAN (Local Area Network).

10.1.3 Software recommendations

Operating system

The WAPT Server is available on Linux and Windows:

- For Linux, **Debian 11 and 12**, **Red Hat 7 / 8 and derivatives**, **Ubuntu server LTS 20.04** 64 bit versions are supported.

Note: SELINUX is supported but not mandatory.

- For Windows, WAPT Server can be installed on **Windows Server** 64 bit versions supported by Microsoft (Win2012r2, Win2k16, Win2k19 or Win2k22). Depending on your need, it can also be installed on recent Win10 Pro/Ent version (20H2 or later).

Attention:

- The WAPT Server will only run on **64bit** based systems.
- Install the Server **without** the graphical user interface.
- `Systemd` must be enabled.

Open Ports

Only ports **80** and **443 MUST** be opened to incoming connections as the WAPT framework works with websockets initiated by the WAPT Agents.

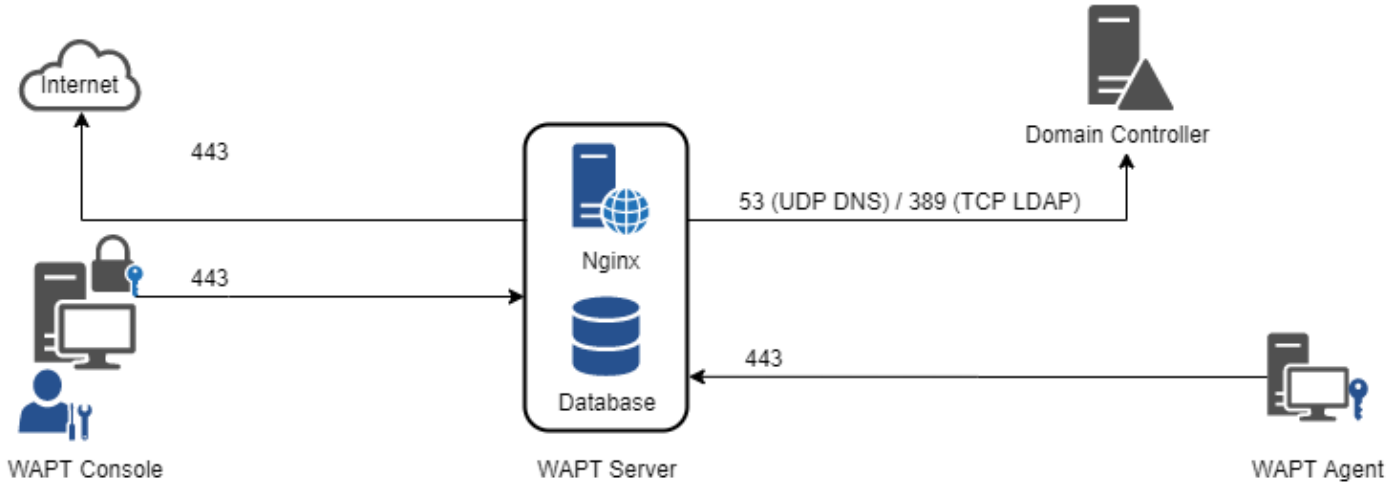


Fig. 1: Data-flow diagram for WAPT

Inbound

Table 2: Inbound ports to open for WAPT to work

Protocol	Port number	Source	Destination	Description
TCP	**80**	All WAPT Agents	WAPT Server	Websocket connection (unsecured) for downloading packages and KB.
TCP	**443**	All WAPT Agents	WAPT Server	Websocket connection for downloading packages and KB.
UDP	**69** Note: tftp uses ephemeral / dynamic ports for data transport. If you have a firewall between the WAPT Server and the fleet of computers, be sure to enable support for tftp conntrack.	**All computers** using *WADS deployment* TFTP method.	WAPT Server	To download the first stage of OS boot files before HTTP becomes available.

Outbound

Table 3: Outbound ports to open for WAPT to work

Protocol	Port number	Source	Destination	Description
TCP	**80**	WAPT Server	Internet	Websocket connection (unsecured) for downloading WAPT packages, `wsusscn2.cab` and KB.
TCP	**80**	WAPT Server	Linux repository (for Linux server) and Tranquil IT repositories ([1])	Uploading of WAPT packages using (unsecured) HTTP.
TCP	**443**	WAPT Server	Linux repository (for Linux server) and Tranquil IT repositories ([?])	Uploading of WAPT packages using (secured) HTTPS.
TCP	**53**	WAPT Server	Domain controller or DNS (Domain Name Service) server	Domain name resolution.
TCP	**389**	WAPT Server	Domain controller or LDAP (Lightweight Directory Access Protocol) server	LDAP authentication to authenticate users with the WAPT Console or the WAPT Self-service.
TCP	**636**	WAPT Server	Domain controller or LDAP server	LDAP authentication.
UDP	**123**	WAPT Server	Domain Controller or NTP (Network Time Protocol) server	NTP to keep time synchronized and kerberos working properly.

10.2 Tips before installing

10.2.1 Configuring the Organization's DNS for WAPT

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

In order to 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.

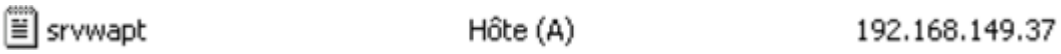
These DNS fields will be used by WAPT Agents to locate the WAPT Server and their WAPT repositories closest to them.

[1] The following DNS names are the Tranquil IT repositories to authorize:

- https://store.wapt.fr
- https://wapt.tranquil.it

10.2.2 Configuring DNS entries in Microsoft RSAT.

- The `A` field **MUST** point to the WAPT Server IP address.



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 a RedHat based distribution*.
- *Install the WAPT Server on Windows* (not recommended for large production networks).

CHAPTER

ELEVEN

INSTALLING WAPT SERVER

11.1 Installing WAPT Server on Debian and Ubuntu

11.1.1 Setting up the server

In order to install a fresh Debian Linux 11 Bullseye or Ubuntu Focal LTS (physical or virtual).

Warning:

- Install **64bit** version.
- Install the Server without the graphical user interface.
- Systemd must be enabled

Attention: The upgrade procedure is different from an initial installation. For an upgrade, please refer to *the documentation on upgrading the WAPT Server*.

Configuring the 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 the 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 WAPT Server.

```
# /etc/hostname of the WAPT Server
srvwapt.mydomain.lan
```

- Configure the `/etc/hosts` file, be sure to put both the *FQDN* and the short name of the WAPT Server.

```
# /etc/hosts of the WAPT Server
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 WAPT 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 of the WAPT Server
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 host with a **`reboot`**.

```
reboot
```

- If it has not already been done, create the *DNS entry* for the WAPT Server in the *Organization*'s Active Directory or DNS server.
- After having rebooted, configure the system locale in English in order to have non-localized logs for easier searching of common errors.

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

- Check whether the machine is properly synchronized with NTP server. If it is not synchronized please refer to the OS documentation to configure **timedatectl**.

```
timedatectl status
```

- Update and upgrade the Operating System and make sure that the Debian default certificate authorities bundle is installed.

```
apt update && apt upgrade
apt install ca-certificates -y
```

- Reboot the WAPT Server.

```
reboot
```

The server is now ready.

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.

11.1.2 Installing the WAPT Server packages

- Update the APT source, retrieve the *.gpg* key from Tranquil IT, then add Tranquil IT's repository.

```
apt install apt-transport-https lsb-release gnupg wget -y
wget -qO- https://wapt.tranquil.it/$(lsb_release -is)/tiswapt-pub.gpg | tee /usr/share/
↪keyrings/tiswapt-pub.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/tiswapt-pub.gpg] https://wapt.tranquil.it/
↪$(lsb_release -is)/wapt-2.4/ $(lsb_release -c -s) main" > /etc/apt/sources.list.d/
↪wapt.list
```

- Then install the WAPT Server packages.

```
export DEBIAN_FRONTEND=noninteractive
apt update
apt install tis-waptserver tis-waptsetup -y
unset DEBIAN_FRONTEND
```

11.1.3 Post-configuring

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.

The post-configuration script rewrites the nginx configuration. A backup file is created when running the postconf in the same directory.

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 (if not defined) 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.The WAPT Server registers all computers that ask to be registered.
  - Choice #2 activates the initial registration based on kerberos(you can activate it later).
  - Choice #3 does not activate the kerberos authentication mechanism for theinitial registering of hosts equipped with WAPT. The WAPT Server will require a login and a password for each host registering with it.

```
WaptAgent Authentication type?

------------------------------------------------------------------------
(x) 1 Allow unauthenticated registration
( ) 2 Enable kerberos authentication required for machines registration.
```

(continues on next page)

(continued from previous page)

```
        Registration will ask for password if kerberos not available
( ) 3 Disable kerberos but registration require strong authentication
-------------------------------------------------------------------------
                                        < OK >          < Cancel >
```

- If you want to use WAPT for OS Deployment, select yes.

```
Do you want to activate os deployment?

        < Yes >        < No >
```

- If you said yes to activate os deployment, postconf will ask you if you want to use a secure authentication in order to deploy your os. It will ask a user/password when you'll try to deploy os.

```
Would you like to activate secure authentication on wads ?

        < Yes >        < No >
```

- Still about wads, if you said yes to the 2 last questions, you'll have a final question :

```
Would you like to mention subnet ip exempt from wads authentication

        < Yes >        < No >
```

If you said yes here too, you'll have to give subnet ip, can be a list for example : 192.168.0.0/24,192.168.1.0/24

- 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.example.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 WAPT Server.

                                    < OK >
```

Listing of post-configuration script options:

Options	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*.

11.2 Installing WAPT Server on a RedHat based distribution

11.2.1 Setting up the RedHat based WAPT Server

In order to install a fresh Redhat or derivative host (virtual or physical) please refer to official documentation.

Warning:

- Install the server without the graphical user interface.

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 the 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 WAPT Server name and the DNS suffix.

- Modify the `/etc/hostname` file and write the *FQDN* of the WAPT Server.

```
# /etc/hostname of the WAPT Server
srvwapt.mydomain.lan
```

- Configure the `/etc/hosts` file, be sure to put both the *FQDN* and the short name of the WAPT Server.

```
# /etc/hosts of the 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 WAPT 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 of the WAPT Server
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 host with a **`reboot`**.

```
reboot
```

- If it has not already been done, create the *DNS entry* for the WAPT Server in the *Organization*'s Active Directory or DNS server.
- After having rebooted, configure the system locale 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 host clock is on time and that SELinux and the firewall are enabled.

```
date
sestatus
systemctl status firewalld
```

- Check whether the machine is properly synchronized with NTP server. If it is not synchronized please refer to the OS documentation to configure **timedatectl**.

```
timedatectl status
```

- Update the distribution and set up the EPEL (Extra Packages for Enterprise Linux) repository.

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

The WAPT Server is now ready.

Attention: The upgrade procedure is different from an initial installation. For an upgrade, please refer to *the documentation on upgrading the WAPT Server*.

11.2.2 Installing the WAPT Server packages

Redhat 9 and derivatives

- Add Tranquil IT's repository.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/redhat9/wapt-2.4/
enabled=1
gpgcheck=1
EOF
```

- Retrieve the `.gpg` key and install the necessary packages.

```
wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/redhat9/RPM-GPG-KEY-TISWAPT-9";␣
↪rpm --import /tmp/tranquil_it.gpg
yum install epel-release -y
yum install tis-waptserver tis-waptsetup cabextract nginx-mod-http-auth-spnego -y
```

- Initialize the PostgreSQL database and activate the services.

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

Redhat 8 and derivatives

- Add Tranquil IT's repository.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/redhat8/wapt-2.4/
enabled=1
gpgcheck=1
EOF
```

- Retrieve the `.gpg` key and install the necessary packages.

```
wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/centos8/RPM-GPG-KEY-TISWAPT-8";␣
↪rpm --import /tmp/tranquil_it.gpg
yum install epel-release -y
dnf module enable nginx:1.20 -y
yum install tis-waptserver tis-waptsetup cabextract nginx-mod-http-auth-spnego -y
```

- Initialize the PostgreSQL database and activate the services.

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

Redhat 7 / CentOS 7 and derivatives

- Add Tranquil IT's repository.

```
cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/centos7/wapt-2.4/
enabled=1
gpgcheck=1
EOF
```

- Retrieve the `.gpg` key and install the necessary 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 -y
yum install tis-waptserver tis-waptsetup cabextract nginx-mod-http-auth-spnego -y
```

- Initialize the PostgreSQL database and activate the services.

```
sudo /usr/pgsql-14/bin/postgresql-14-setup initdb
sudo systemctl enable postgresql-14 waptserver nginx
sudo systemctl start postgresql-14 nginx
```

11.2.3 Post-configuring

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.

The post-configuration script rewrites the nginx configuration. A backup file is created when running the postconf in the same directory.

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 (if not defined) 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.The WAPT Server registers all computers that ask to be registered.
  - Choice #2 activates the initial registration based on kerberos(you can activate it later).
  - Choice #3 does not activate the kerberos authentication mechanism for theinitial registering of hosts equipped with WAPT. The WAPT Server will require a login and a password for each host registering with it.

```
WaptAgent Authentication type?

------------------------------------------------------------------------
(x) 1 Allow unauthenticated registration
( ) 2 Enable kerberos authentication required for machines registration.
```

(continues on next page)

(continued from previous page)

```
        Registration will ask for password if kerberos not available
( ) 3 Disable kerberos but registration require strong authentication
-------------------------------------------------------------------------
                                        < OK >          < Cancel >
```

- If you want to use WAPT for OS Deployment, select yes.

```
Do you want to activate os deployment?

        < Yes >        < No >
```

- If you said yes to activate os deployment, postconf will ask you if you want to use a secure authentication in order to deploy your os. It will ask a user/password when you'll try to deploy os.

```
Would you like to activate secure authentication on wads ?

        < Yes >        < No >
```

- Still about wads, if you said yes to the 2 last questions, you'll have a final question :

```
Would you like to mention subnet ip exempt from wads authentication

        < Yes >        < No >
```

If you said yes here too, you'll have to give subnet ip, can be a list for example : 192.168.0.0/24,192.168.1.0/24

- 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.example.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 WAPT Server.

                                    < OK >
```

Listing of post-configuration script options:

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

Your WAPT Server is now ready. You may go to the documentation on *installing the WAPT Console*.

11.3 Installing WAPT Server on Windows

11.3.1 To read beforehand

- Installing WAPT on a Linux server is the recommended method, unless you are trialing WAPT and you are not familiar with Linux.
- The WAPT Server can not be installed on a computer with services already listening on port 443 (example WSUS with IIS). Port 443 is used by the WAPT Server and **MUST** be available. If port 443 is already occupied by another web service, you should take a look at the Microsoft official documentation for changing the default ports on Windows.
- The WAPT Server **will not run** on a x86 version of Windows and fresh Windows version currently supported by Microsoft. The server component of WAPT works just as well on a win10 client VM or a physical host as it does on a Windows server version.
- The installation of the WAPT Server **MUST** be done using a Local Administrator account on the host and **NOT a Domain Administrator** account.
- **Nginx** is the **ONLY** supported web server with WAPT. **Apache or IIS (with or without WSUS) are NOT supported in WAPT**.
- In case of problems when installing WAPT, visit the *Frequently Asked Questions*.

11.3.2 Installing WAPT server

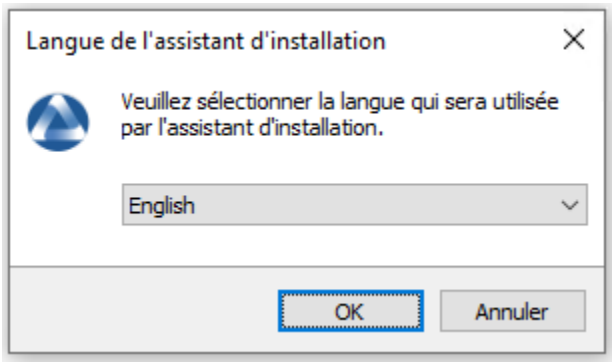
Warning: The post-configuration script rewrites the nginx configuration. If you use a special configuration, save your `nginx.conf` file with the command:

```
copy C:\wapt\waptserver\nginx\conf\nginx.conf C:\wapt\waptserver\nginx\conf\nginx.conf.
↪old
```

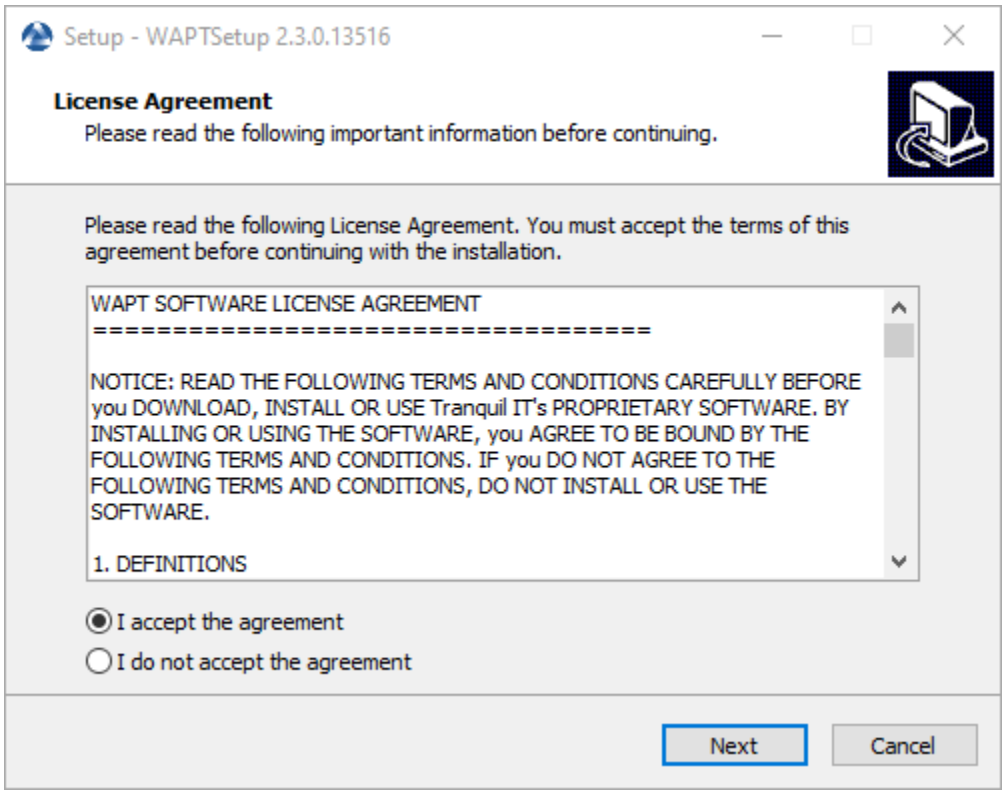
It will be necessary to overwrite the configuration after the post-configuration with the command:

```
copy C:\wapt\waptserver\nginx\conf\nginx.conf.old C:\wapt\waptserver\nginx\conf\nginx.
↪conf
```

- Download and execute .
- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).
- Choose the password for the WAPT Server.

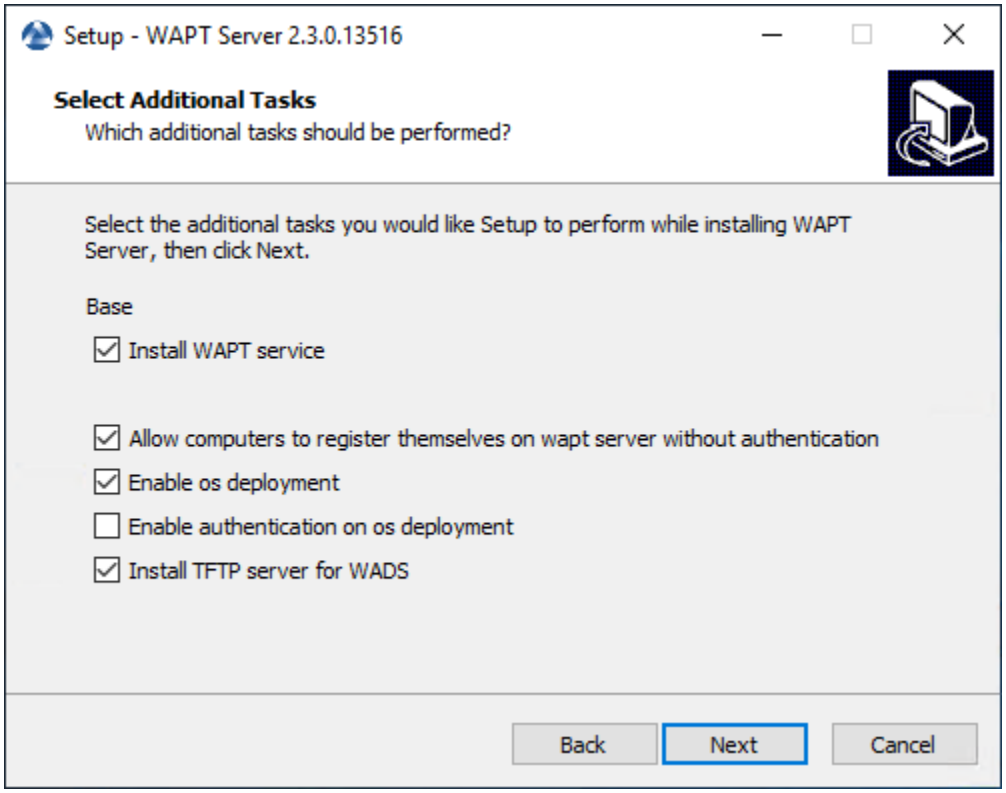
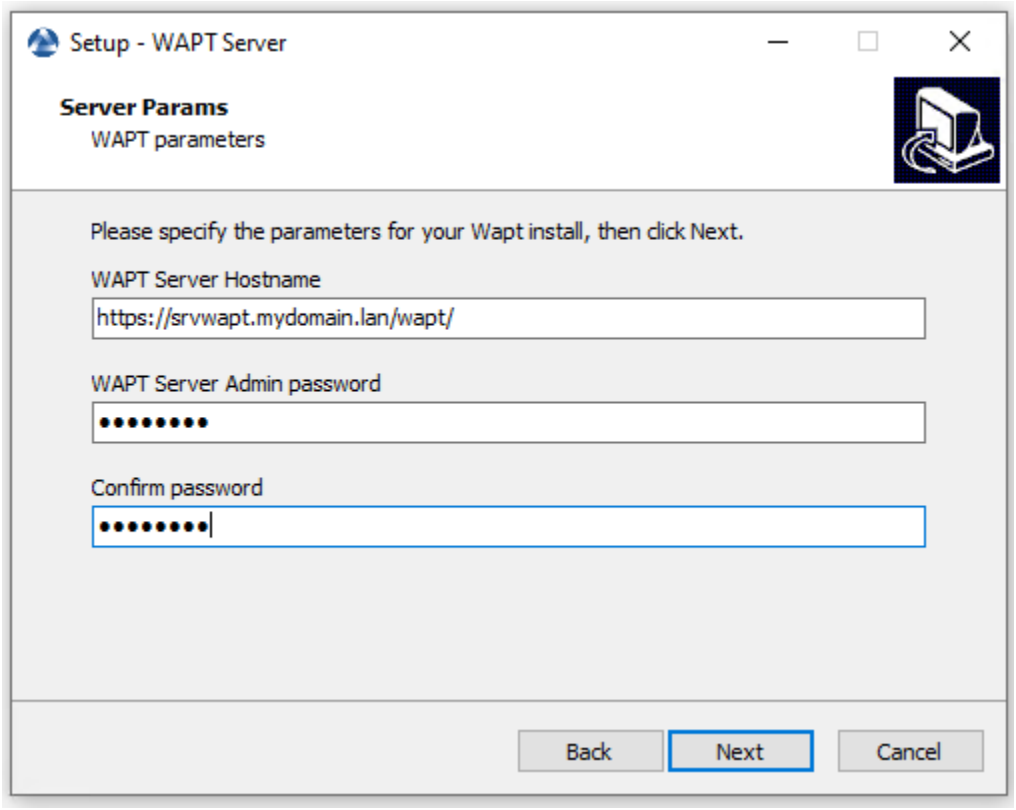
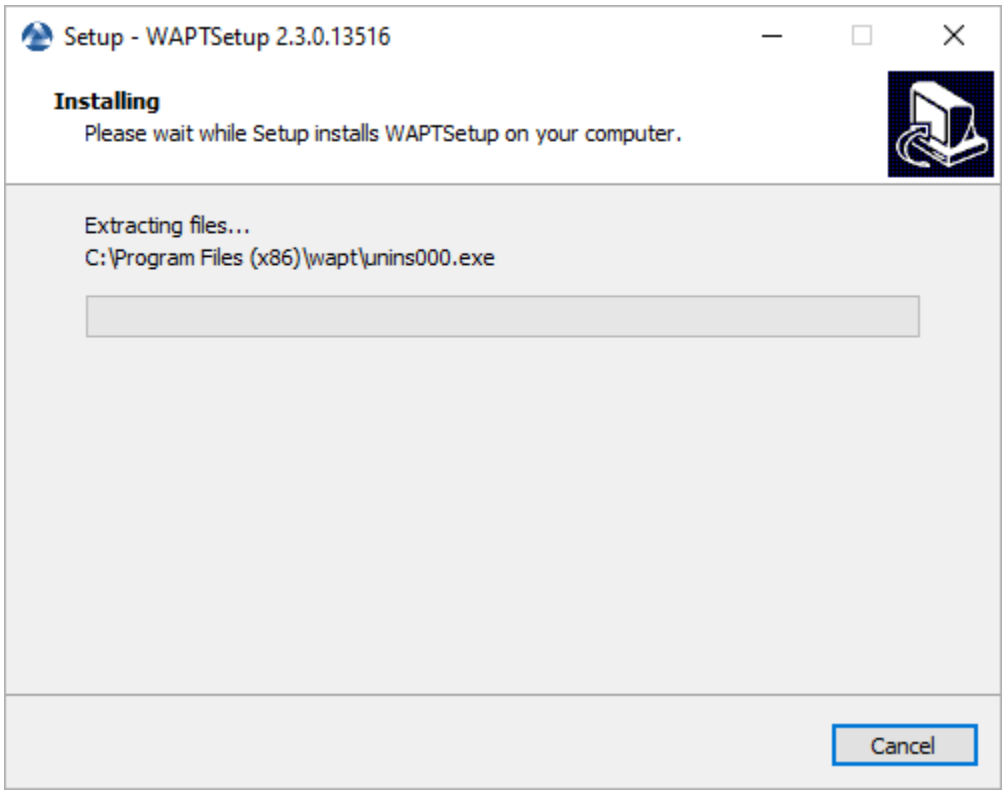


Fig. 1: Choosing the installer options for deploying the WAPT Server



- Click on the *Install* to launch the installation, wait for the installation to complete.



- Click on *Finish* to close the window.

Attention:

- **For security, do not run the WAPT Console or your WAPT package development tool on the WAPT Server.**
- The WAPT Server on Windows **includes the WAPT Agent**. It is not necessary to install the WAPT Agent to manage the WAPT Server on Windows.

Your WAPT Server is now ready. You may now go to the documentation on *Installing the WAPT management Console*.

CHAPTER

TWELVE

UPGRADE WAPT SERVER

If your WAPT Server is a virtual host, take a snapshot of the VM. This way, you will be able to go back easily in the rare case that the update fails.

Attention: After each WAPT Server update, update your *WAPT Console*, then *regenerate* the WAPT Agent.

Before upgrading WAPT Server, please refer to the following upgrading compatibility chart:

Table 1: Available WAPT Upgrade paths

	To WAPT 2.4
From WAPT 1.8.2	✅
From WAPT 2.0	✅
From WAPT 2.1	✅
From WAPT 2.2	✅
From WAPT 2.3	✅

Warning: If upgrading from a version older than WAPT 2.1, the *licence activation* process has changed.

12.1 Switching of WAPT Edition (Community, Discovery, Enterprise)

WAPT Community is no longer supported. If you want to upgrade from WAPT 1.8.2 Community you can upgrade to WAPT Discovery or WAPT Enterprise. Please note that WAPT Discovery is limited to 300 clients.

To upgrade from a WAPT Community setup to WAPT Discovery or Enterprise follow the standard *1.8.2 to 2.4 upgrade documentation*.

The WAPT Server will make the appropriate changes.

To upgrade WAPT Discovery to WAPT Enterprise simply upload a valid *licence* to the WAPT Server from the WAPT Console.

If your Enteprise licence expire, it will fall back on the Discovery Edition. If you are running WAPT Discovery and you have more that 300 client computers in your inventory, the WAPT Console will stop working and will only give you the option to delete computer entries from the inventory. The WAPT Console will return to working condition when the inventory returns below the 300 computer limit.

12.2 Minor upgrade

12.2.1 Upgrading from version 2.4 to latest 2.4

To do a minor upgrade please follow the procedure corresponding to your server operating system.

Debian / Ubuntu

- Update the underlying distribution and upgrade WAPT Server.

```
export DEBIAN_FRONTEND=noninteractive
apt update && apt upgrade -y
apt install tis-waptserver tis-waptsetup -y
unset DEBIAN_FRONTEND
```

- Launch the post-configuration step post-configuration step
- Once completed your WAPT Server is ready.

RedHat and derivatives

- Update the underlying distribution and upgrade WAPT Server.

```
yum update -y
yum install tis-waptserver tis-waptsetup -y
```

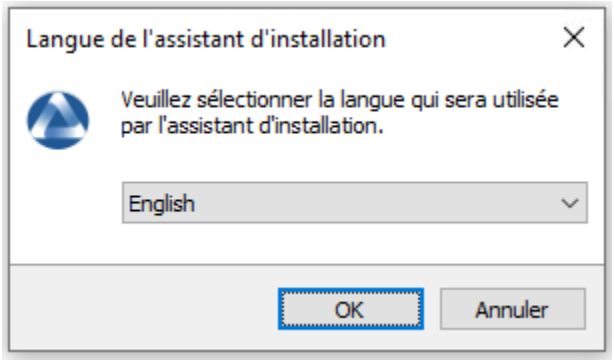
- Launch the post-configuration step post-configuration step
- Once completed your WAPT Server is ready.

Windows

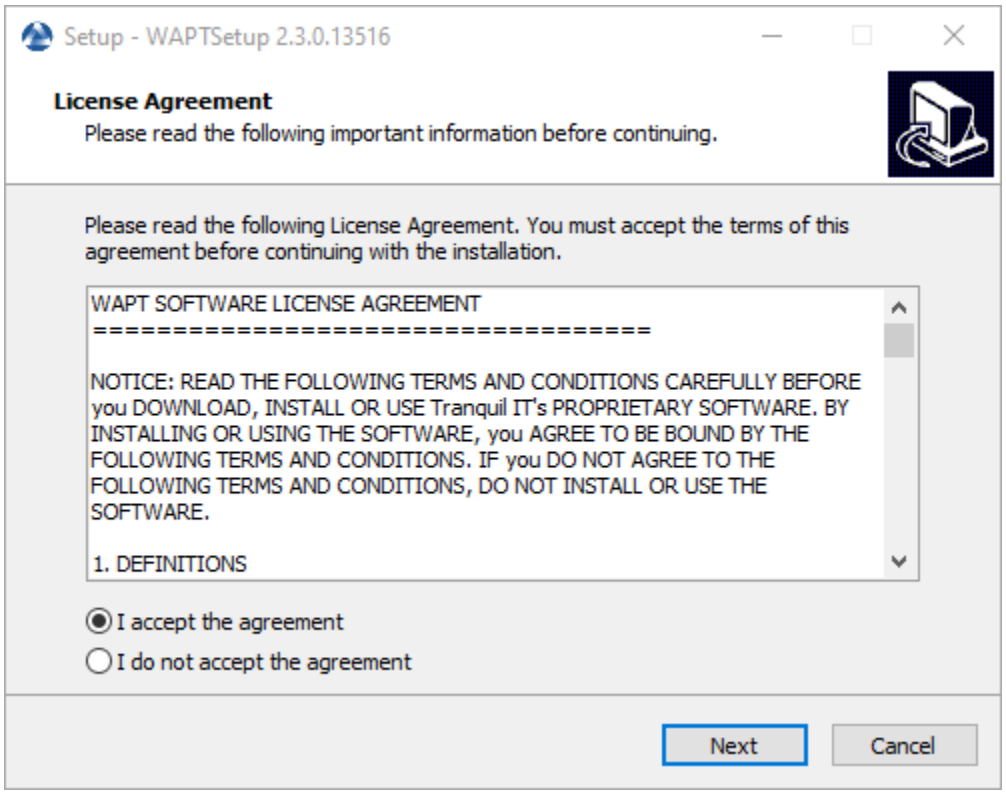
- Download and execute `waptserversetup.exe`.

Attention: The installation of the WAPT Server **MUST** be done using a **Local Administrator** account on the host

- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).
- Do not change the password for the WAPT Server (if not necessary).

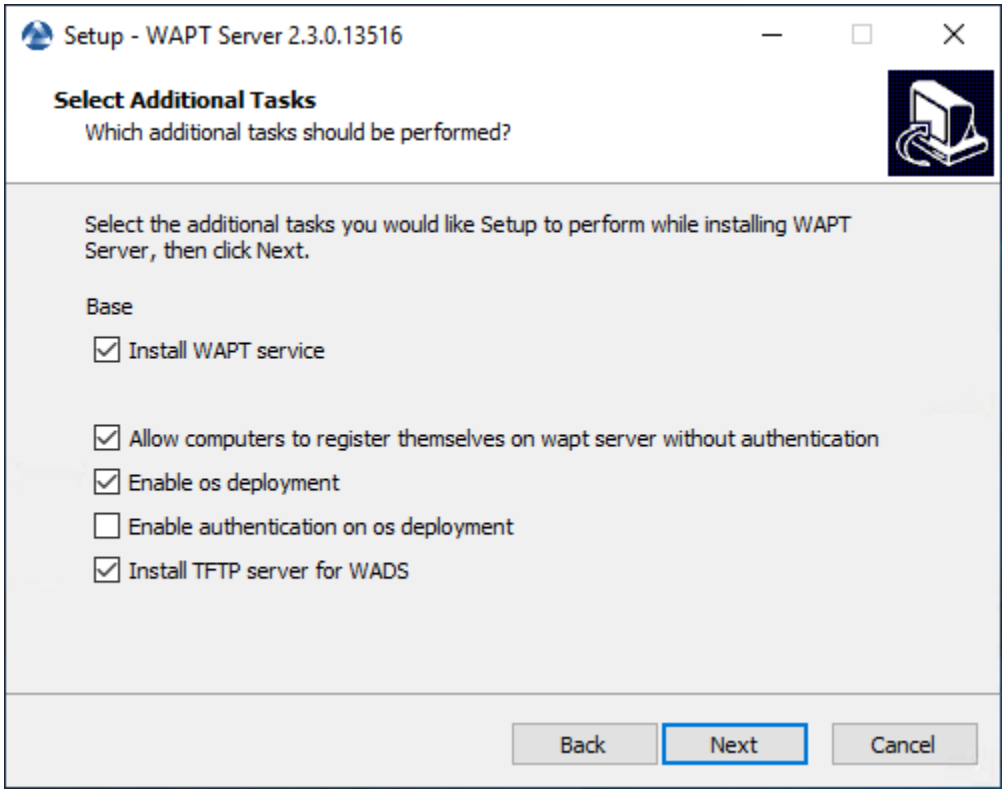
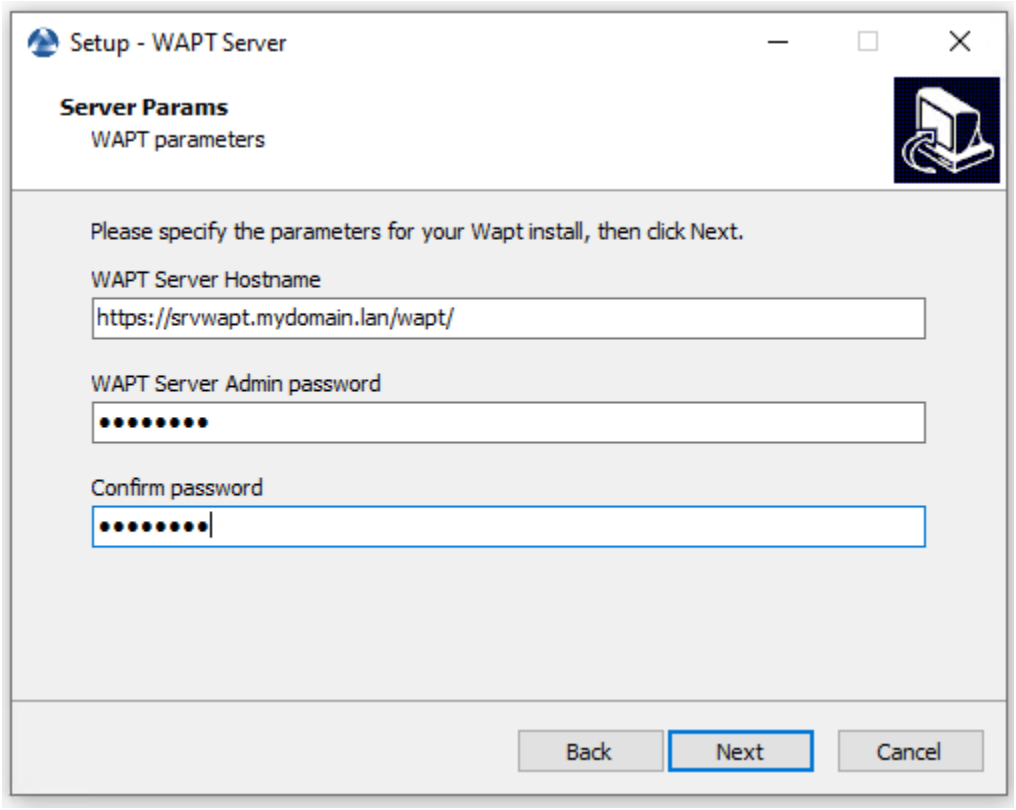
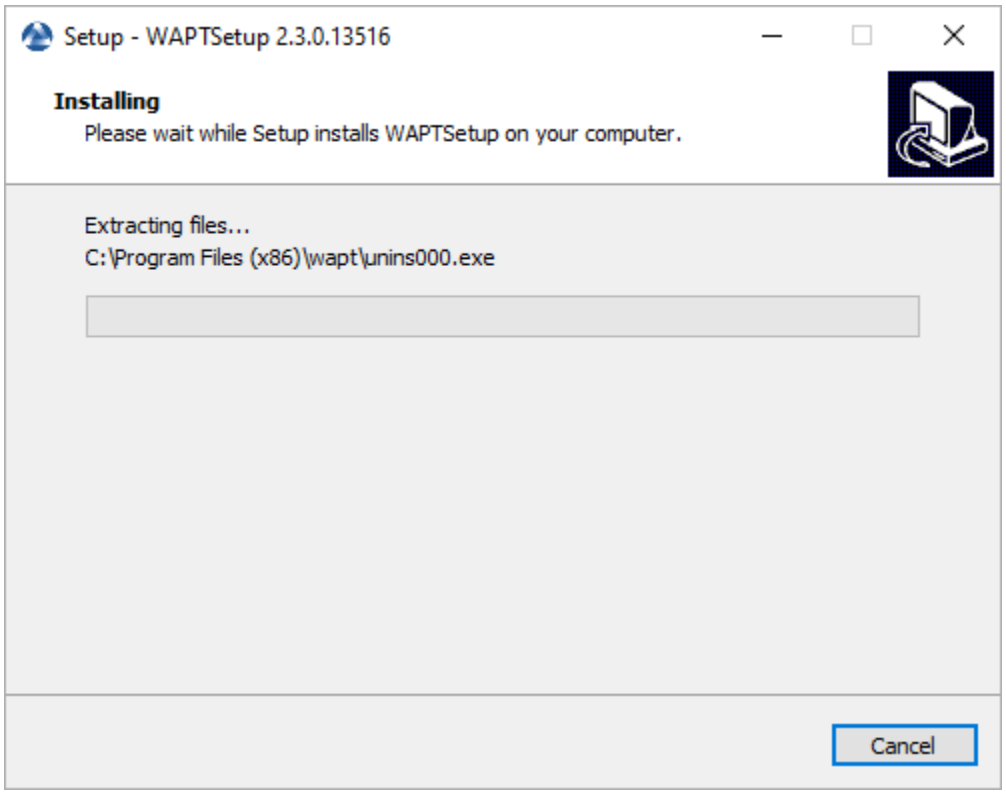


Fig. 1: Choosing the installer options for deploying the WAPT Server



- Click on the *Install* to launch the installation, wait for the installation to complete.



- Click on *Finish* to close the window.

- Once completed your WAPT Server is ready.

12.2.2 Upgrading from version 2.x to 2.4

Note: Before upgrading, ensure that *installation requirements* are met.

WAPT 2.4 needs PostgreSQL 10 or above. If you have upgraded from a older Debian or Ubuntu version with PostgreSQL 9.6, be sure to follow the OS documentation to upgrade PostgreSQL to latest version.

If you are using WAPT WADS, please note that WAPT 2.3 WADS WinPE and WAPT 2.4 WADS WinPE are not compatible and you need to recreate WinPE File using the "upload WinPE" button in the OS Deployment tab.

Indeed, WAPT changed OpenSSL version from 1.1.1 to 3.x.

Debian / Ubuntu

- First of all, update the underlying distributionand install the WAPT Server packages.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
```

- Then the update the package repository and import the GPG key from the repository.

```
wget -O - https://wapt.tranquil.it/$(lsb_release -is)/tiswapt-pub.gpg | apt-key add -
echo "deb https://wapt.tranquil.it/$(lsb_release -is)/wapt-2.4/ $(lsb_release -c -s) main" > /etc/
↪apt/sources.list.d/wapt.list
```

- Update the repository and install the packages.

```
export DEBIAN_FRONTEND=noninteractive
apt update
apt install tis-waptserver tis-waptsetup -y
unset DEBIAN_FRONTEND
```

- Launch the post-configuration step post-configuration step

RedHat and derivatives

- First of all, update the underlying distribution and necessary packages.

```
yum update -y
yum install epel-release redhat-lsb-core -y
```

- Then add or update the package repository WAPT packages and import the GPG key from the repository.

```
RH_VERSION=$(cat /etc/system-release-cpe | awk -F: '{ print $5}')

cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
baseurl=https://wapt.tranquil.it/redhat${RH_VERSION}/wapt-2.4/
enabled=1
gpgcheck=1
```

(continues on next page)

(continued from previous page)

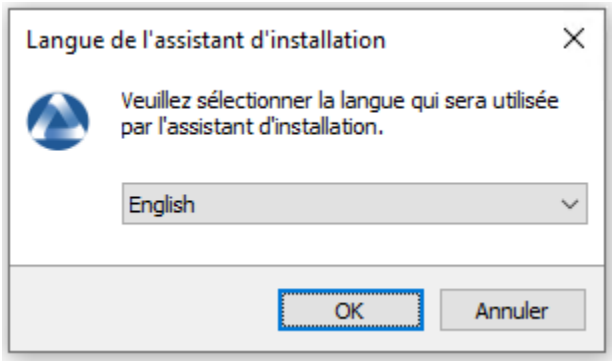
```
EOF

wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/redhat${RH_VERSION}/RPM-GPG-KEY-TISWAPT-$
↪{RH_VERSION}"; rpm --import /tmp/tranquil_it.gpg
```

- And finally upgrade the WAPT Server.

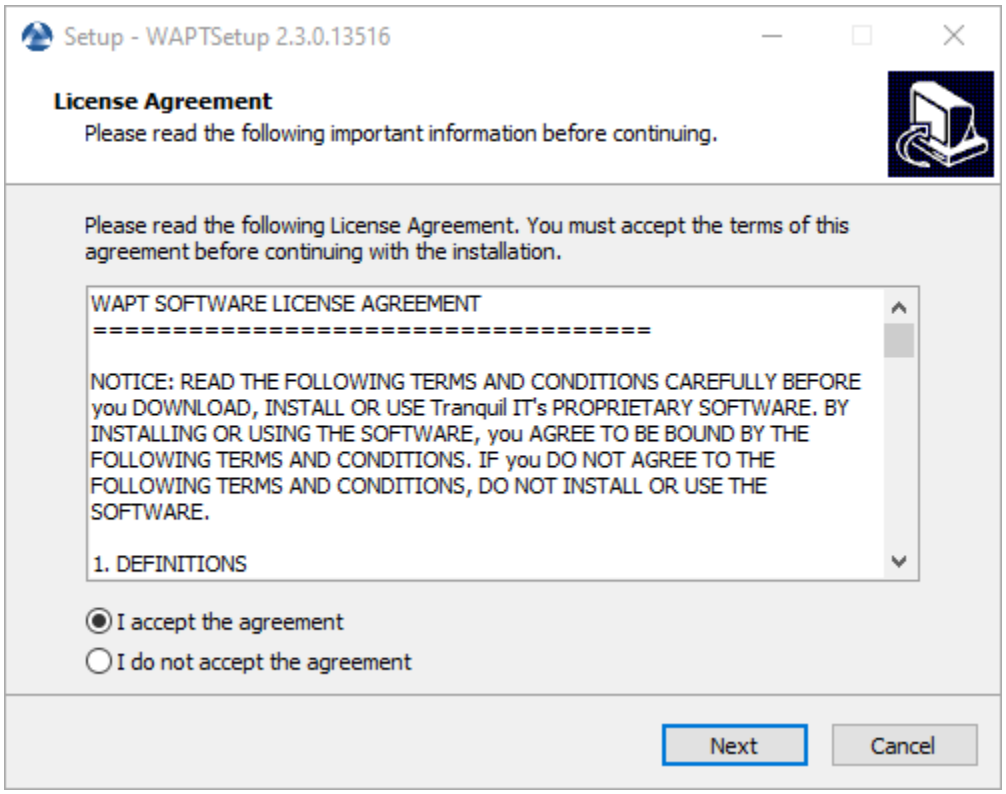
```
yum install tis-waptserver tis-waptsetup cabextract -y
```

Windows

- Download and execute .
- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- If old folder installation found, this message appear. Click on *Yes* to go on to the next step.

- Select additional task if needed.
- Change the WAPT Server password if needed, then press *Next*.

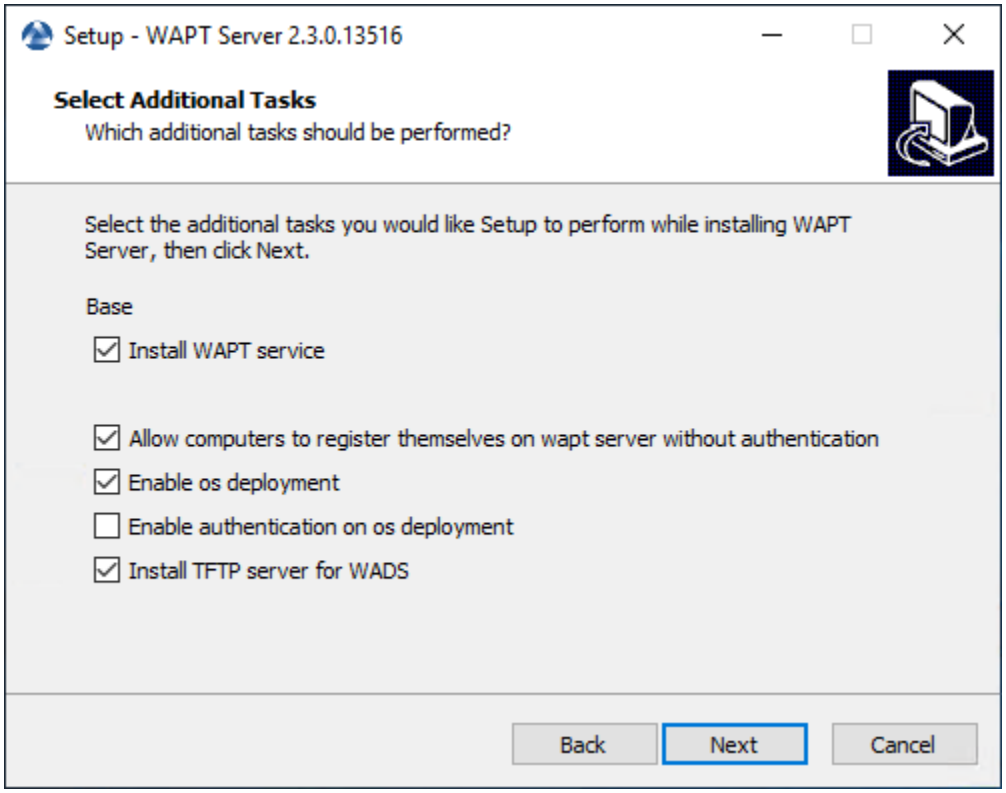
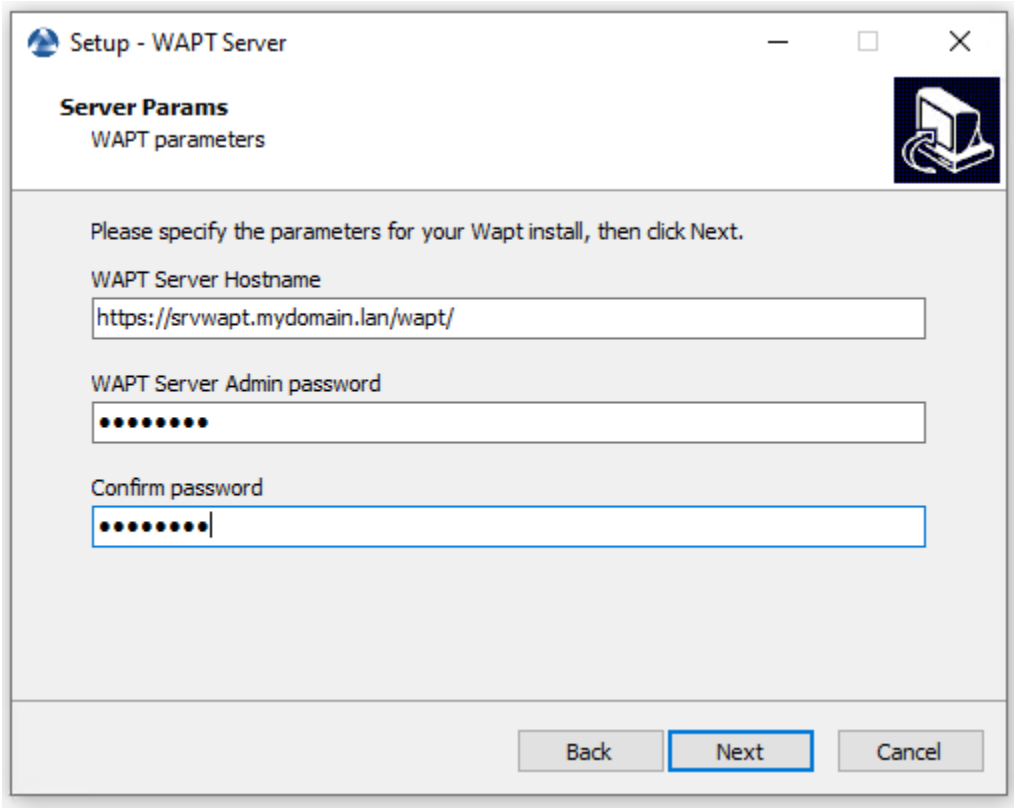
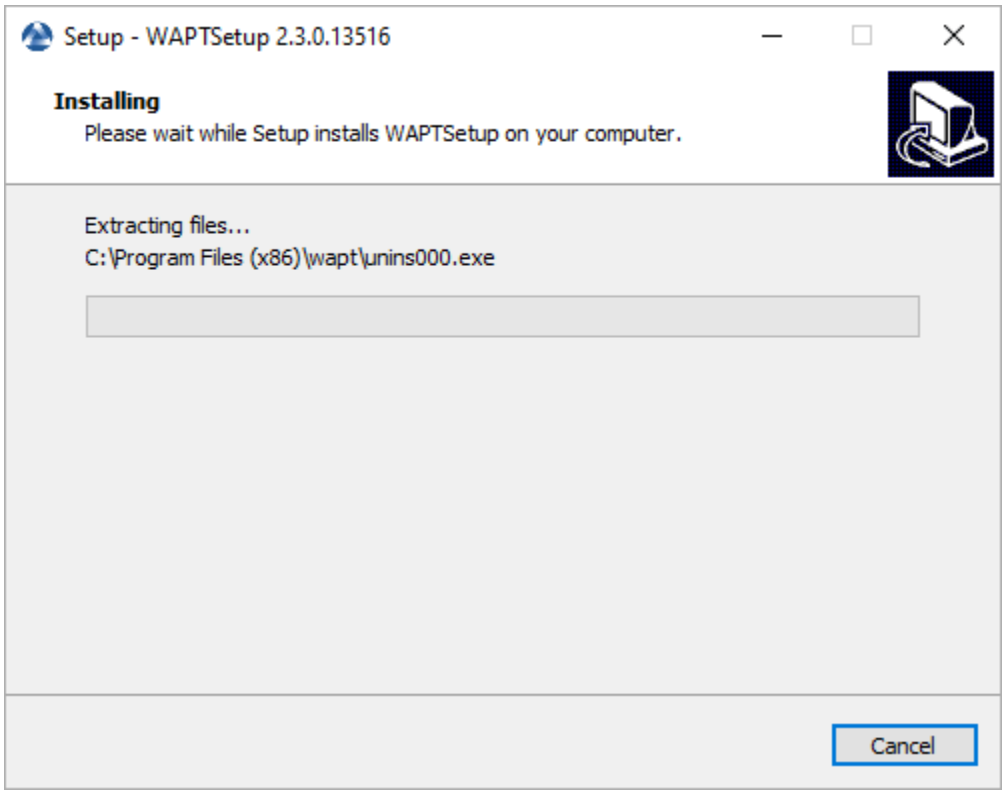


Fig. 2: Choosing the installer options for deploying the WAPT Server



- Click on the *Install* to launch the installation, wait for the installation to complete.



- Click on *Finish* to close the window.

Attention: **DO NOT** use the WAPT Console on the WAPT Server. **DO NOT** install nor run your WAPT package development tools on the WAPT Server.

The WAPT Server on your Windows server or workstation is ready.

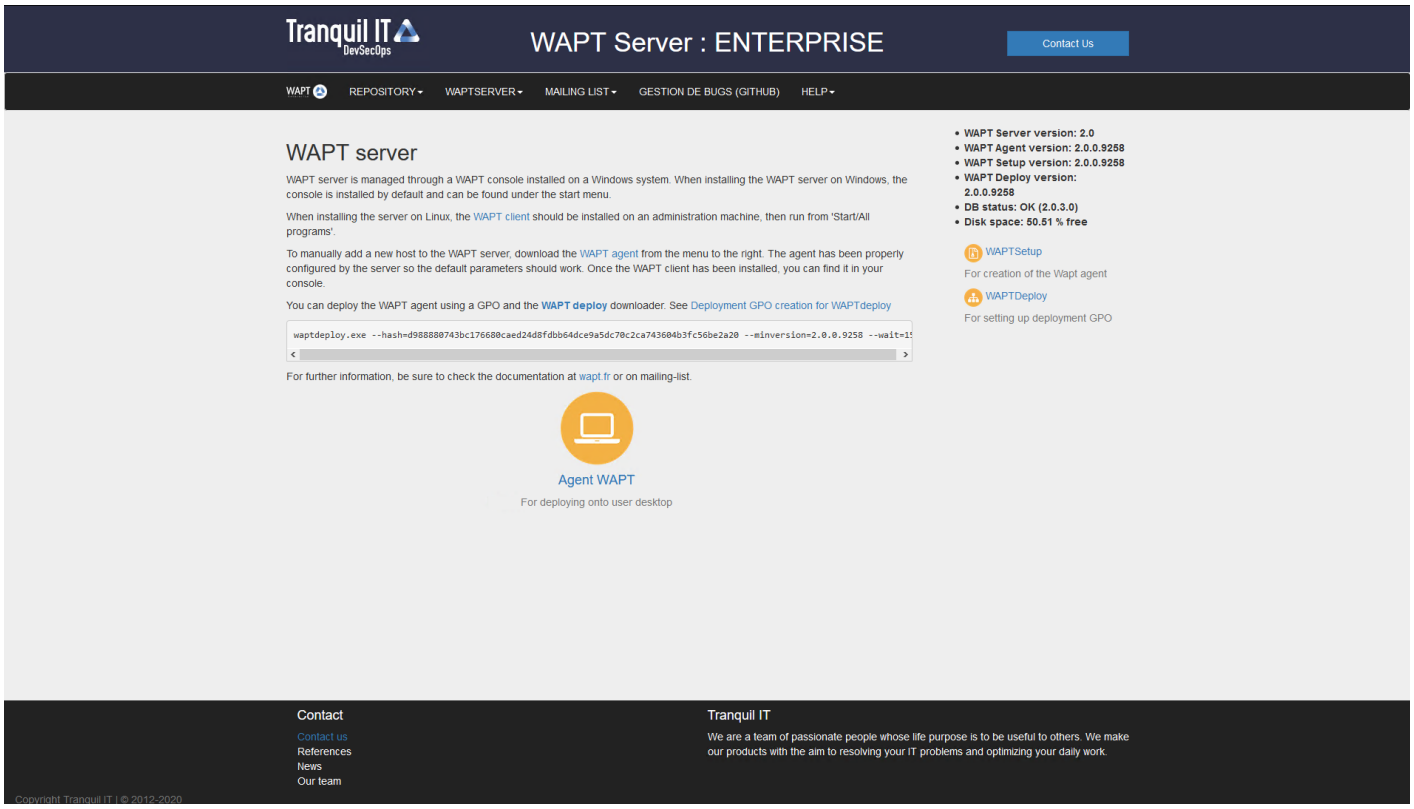


Fig. 3: The WAPT Server interface in a web browser

Your WAPT Server is now ready. You may now go to the documentation on *Installing the WAPT management Console*.

12.3 Upgrading from version 1.8.2 to 2.4

The changement between WAPT 1.8.2 (**not possible from a older version, please upgrade to 1.8.2 beforehand**) and 2.4 are numerous. First of all, 1.8.2 was in Python2, we changed to Python3. A lot of new features are available too (essentially in Enterprise version).

12.3.1 Before upgrading

Ensure that *installation requirements* are met.

Backup your WAPT Private and Public certificates which allows you to deploy your WAPTAgent and packages. Usually, it is located in `C:\private` on your computer where the WAPT Console is set. If you don't remember what this key is please refer to the section about *generating the Administrator's certificate for signing WAPT packages* for better understandment.

In this documentation, your WAPT certificate's name will be **wapt-private.crt**.

12.3.2 Upgrading

Debian / Ubuntu

Note: If you are running on Debian9 Stretch, you have first to upgrade to Debian10 or Debian11 before upgrading to WAPT 2.x. **The WAPT Server 2.x is not available for Debian9**.

It is even recommended to upgrade to Debian 11 Bullseye. In this case, upgrade from Debian 9 => Debian 10 => Debian 11.

- First of all, update the underlying distribution and install the WAPT Server packages.

```
apt update && apt upgrade -y
apt install apt-transport-https lsb-release gnupg
```

- Then update the package repository and import the GPG key.

```
wget -O - https://wapt.tranquil.it/$(lsb_release -is)/tiswapt-pub.gpg | apt-key add -
echo "deb https://wapt.tranquil.it/$(lsb_release -is)/wapt-2.4/ $(lsb_release -c -s) main" > /etc/
↪apt/sources.list.d/wapt.list
```

- Update the repository and install the packages.

```
export DEBIAN_FRONTEND=noninteractive
apt update
apt install tis-waptserver tis-waptsetup -y
unset DEBIAN_FRONTEND
```

- Launch the post-configuration step post-configuration step.

RedHat and derivatives

- First of all, update the underlying distribution and necessary packages.

```
yum update -y
yum install epel-release -y
```

- Then add or update the package repository WAPT packages and import the GPG key from the repository.

```
RH_VERSION=$(cat /etc/system-release-cpe | awk -F: '{ print $5}')

cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name=WAPT Server Repo
```

(continues on next page)

(continued from previous page)

```
baseurl=https://wapt.tranquil.it/redhat${RH_VERSION}/wapt-2.4/
enabled=1
gpgcheck=1
EOF

wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/redhat${RH_VERSION}/RPM-GPG-KEY-TISWAPT-$
↪{RH_VERSION}"; rpm --import /tmp/tranquil_it.gpg
```

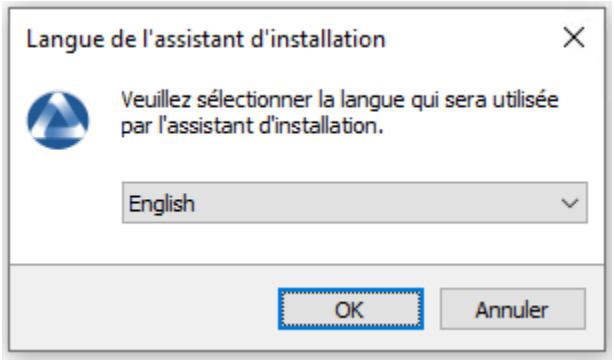
- And finally upgrade the WAPT Server.

```
yum install tis-waptserver tis-waptsetup cabextract -y
```

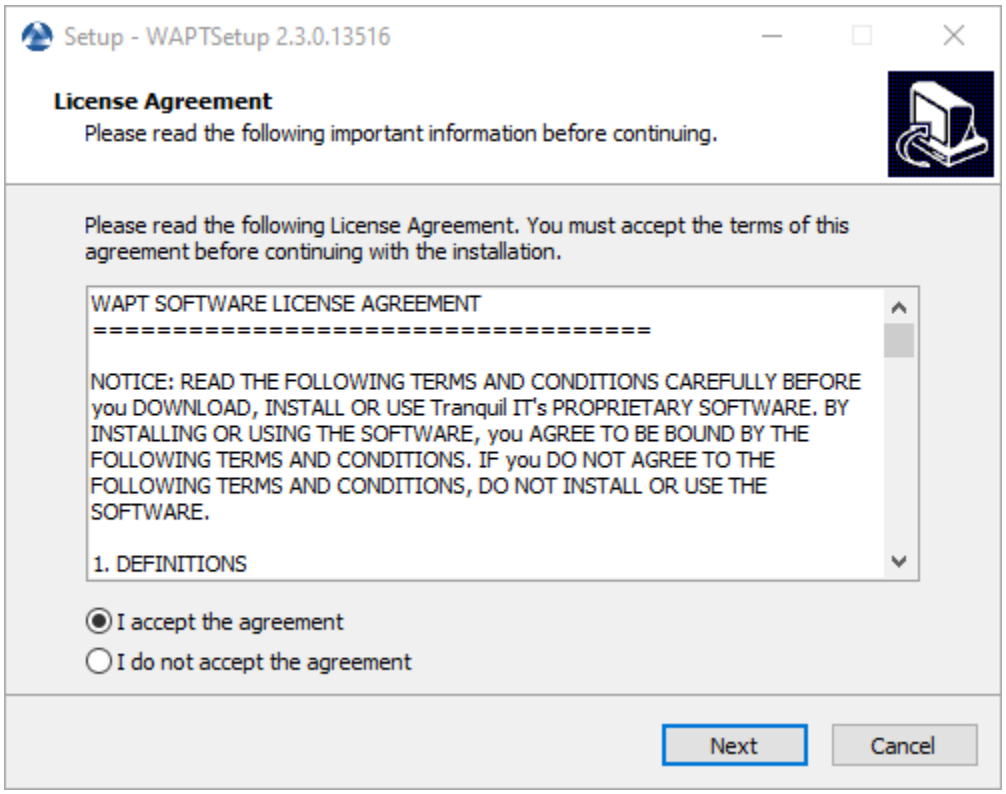
- Launch the post-configuration step post-configuration step.

Windows

- Download and execute .
- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- If an old folder installation is found, this message appears. Click on *Yes* to go on to the next step.

- Select additional task if needed.
- Change the WAPT Server password if needed, then press *Next*.

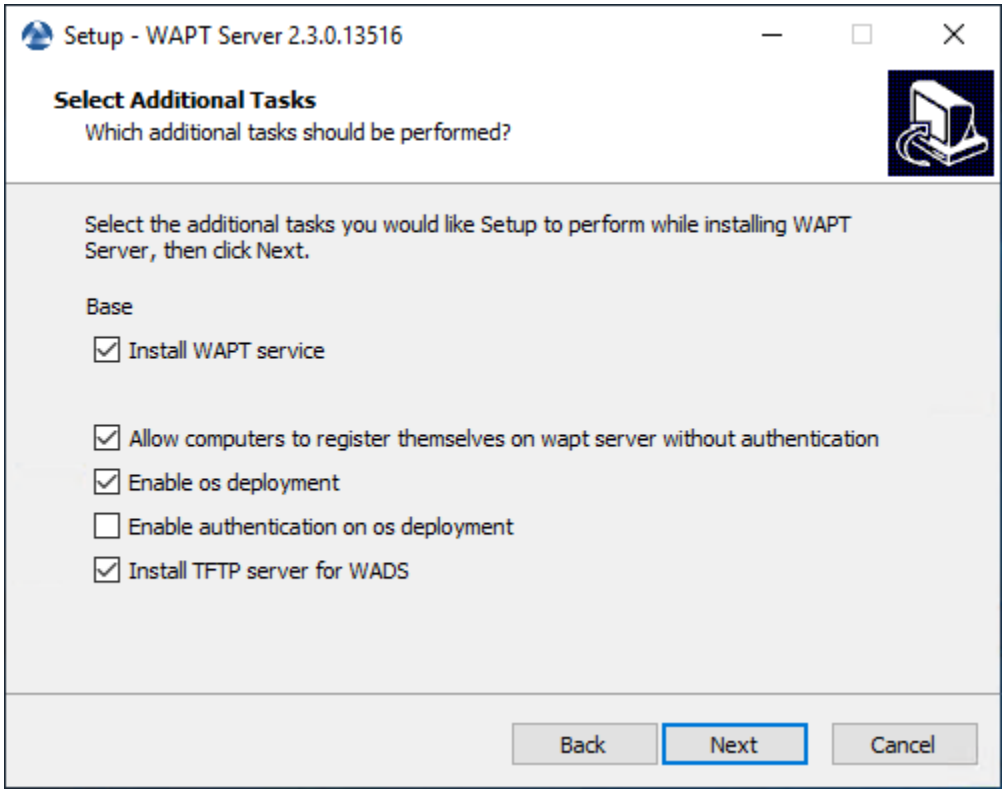
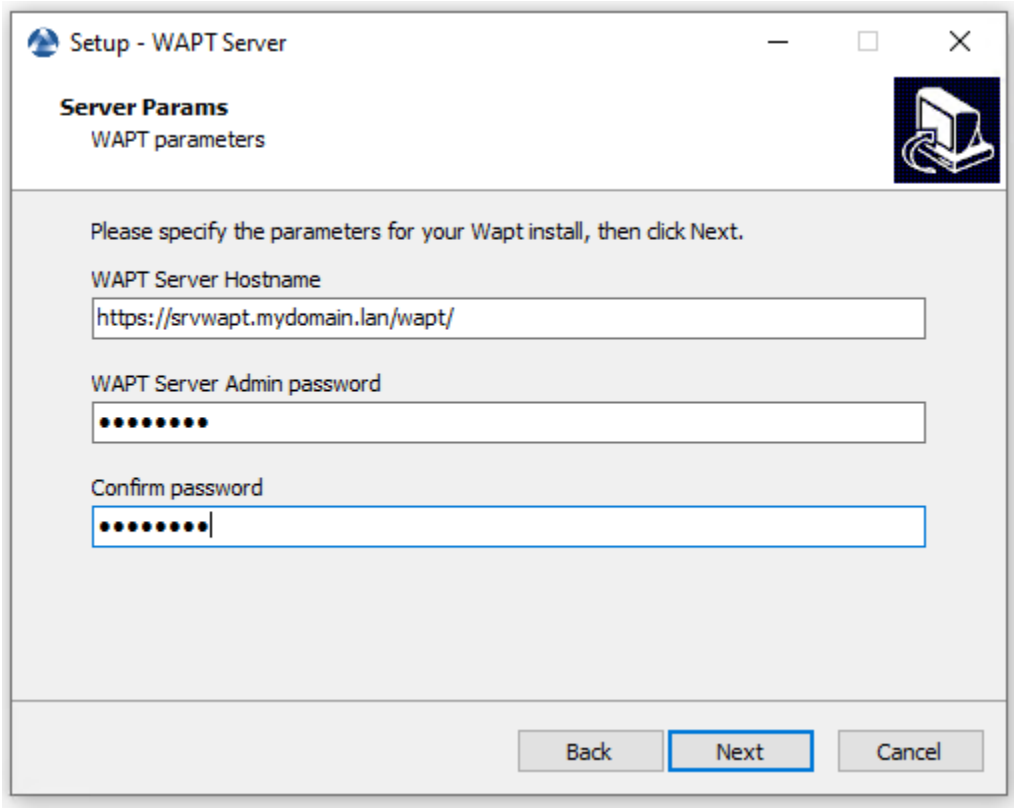
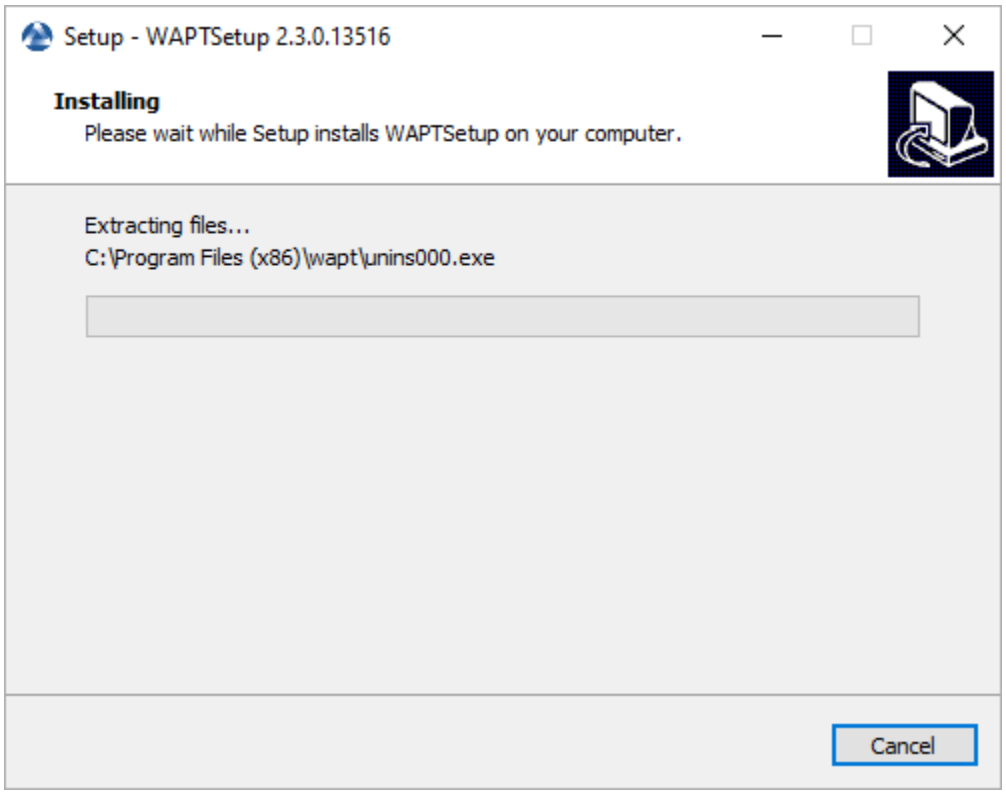


Fig. 4: Choosing the installer options for deploying the WAPT Server



- Click on the *Install* to launch the installation, wait for the installation to complete.



- Click on *Finish* to close the window.

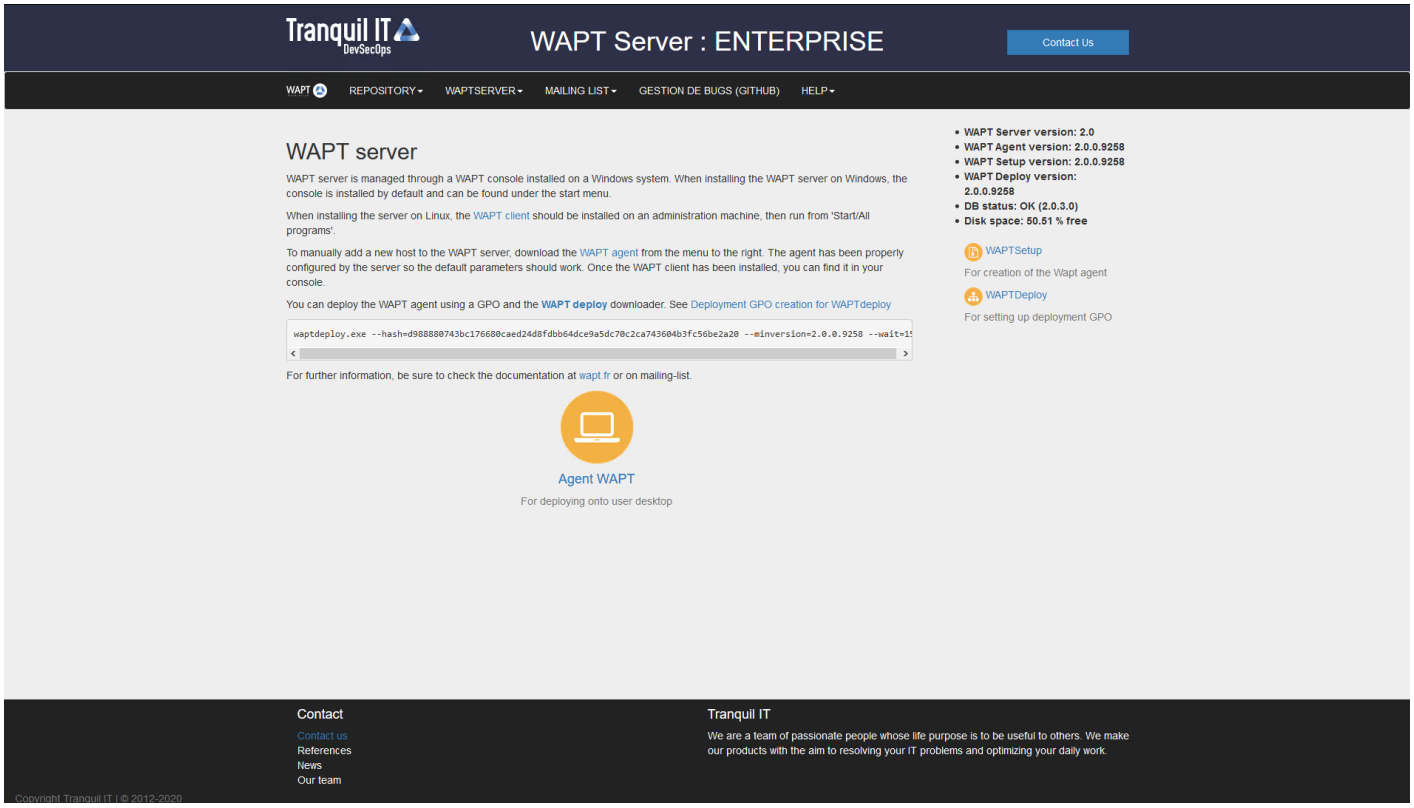


Fig. 5: The WAPT Server interface in a web browser

The WAPT Server on your Windows server or workstation is ready.

Attention: **DO NOT** use the WAPT Console on the WAPT Server. **DO NOT** install nor run your WAPT package development tools on the WAPT Server.

Your WAPT Server is now ready.

12.3.3 The WAPT management Console

To download the `waptsetup.exe` file, point your web browser to your waptserver url https://srvwapt.mydomain.lan, then click on the *WAPTSetup* link on the right-hand side of the WAPT Server web page. The WAPT Server home page only provides basic server status information and the download link for the WAPT Console.

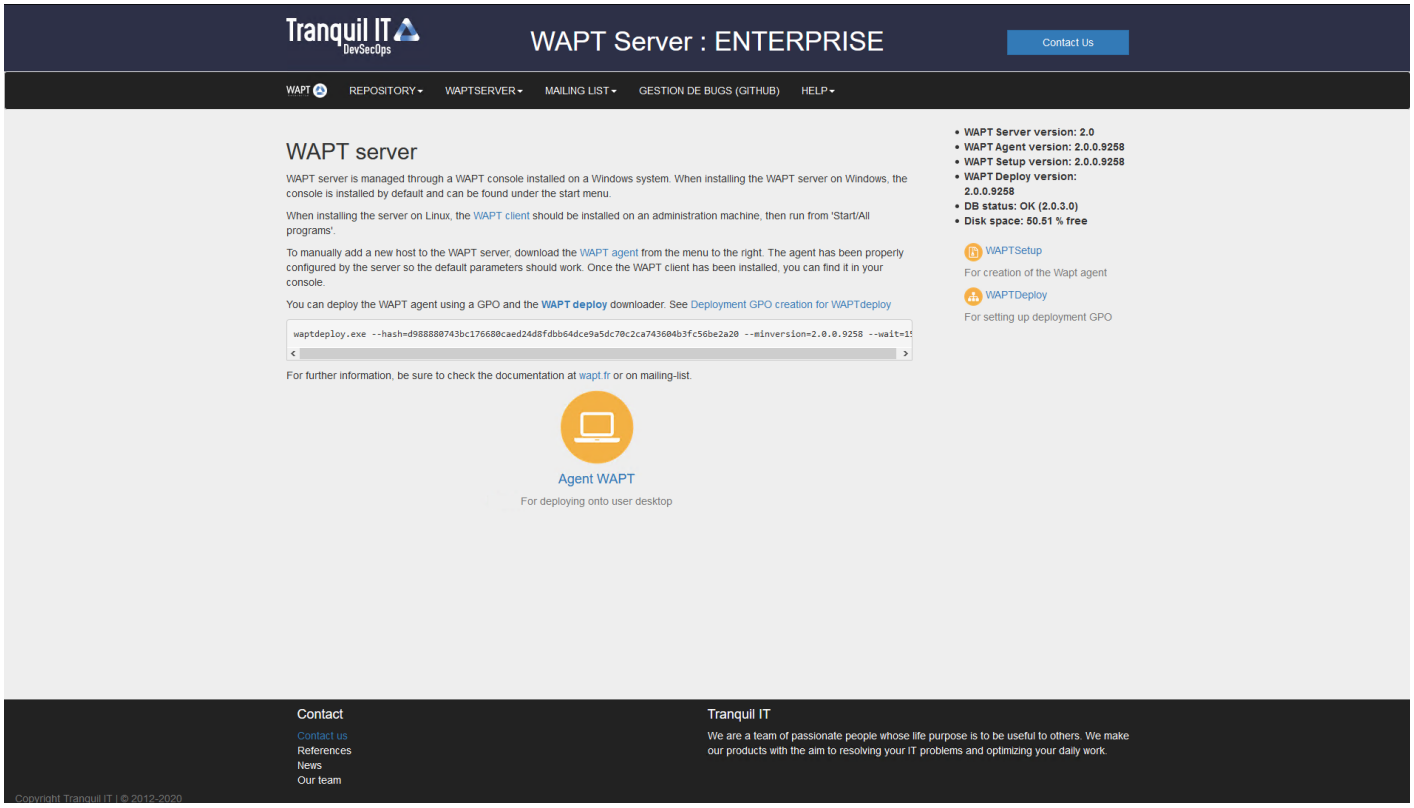
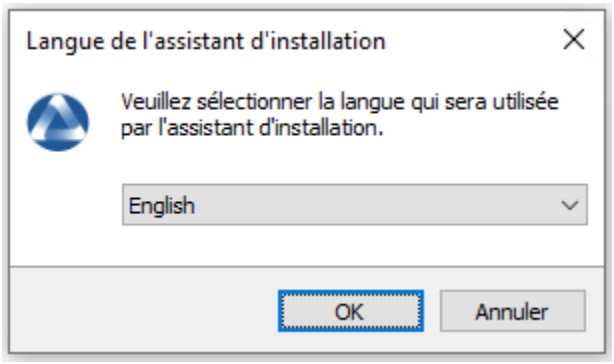


Fig. 6: The WAPT Server interface in a web browser

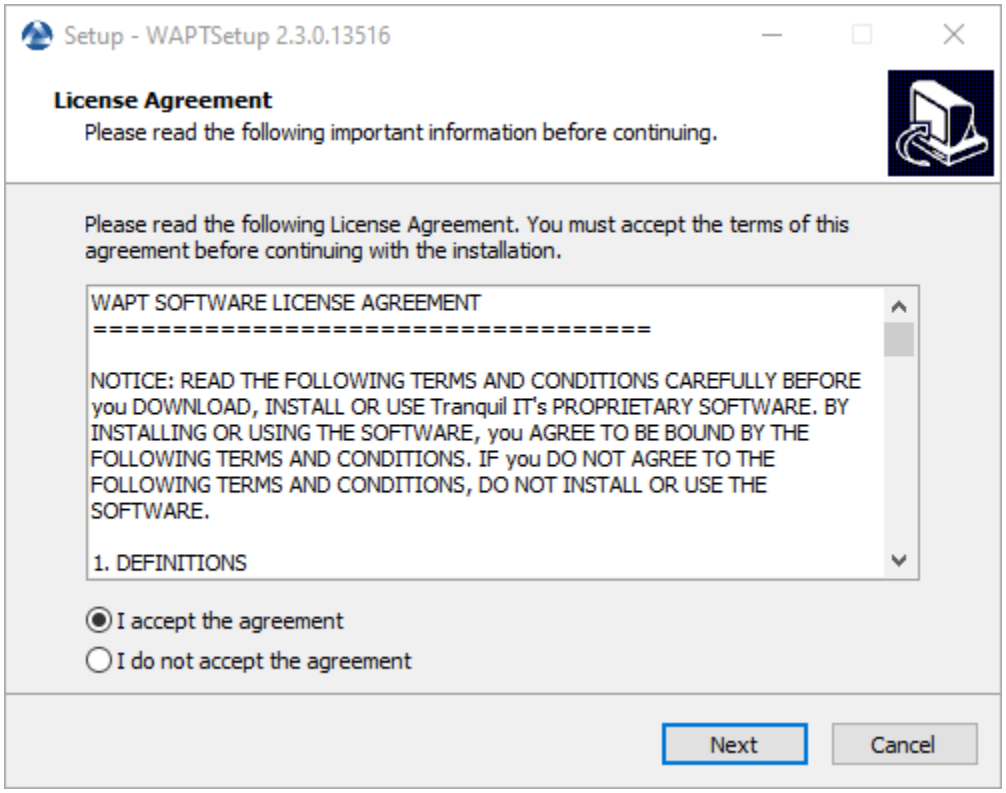
Installing the WAPT Agent on the Administrator's computer

Attention: If **the WAPT Agent** is not compiled and installed on your computer, you need to run de WAPT Agent installer to open and *configure the WAPT Console*.

- Start the executable installer as *Local Administrator* on the *Administrator*'s workstation.
- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).

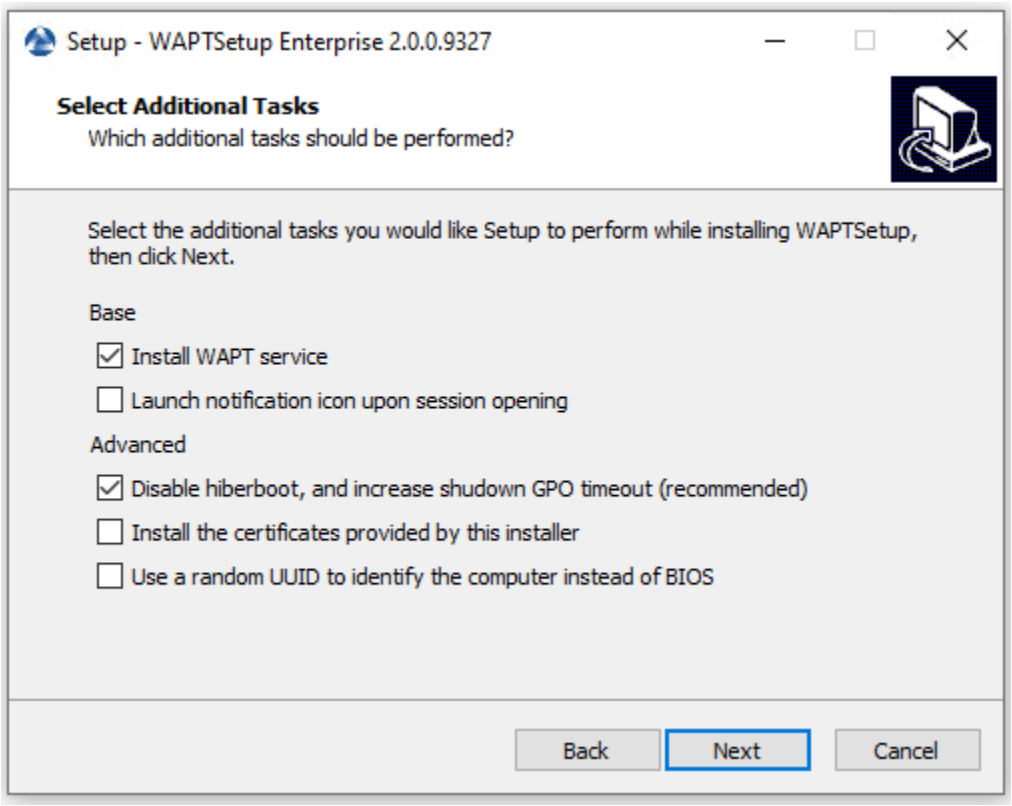


Fig. 7: Choosing the WAPT Agent installer options

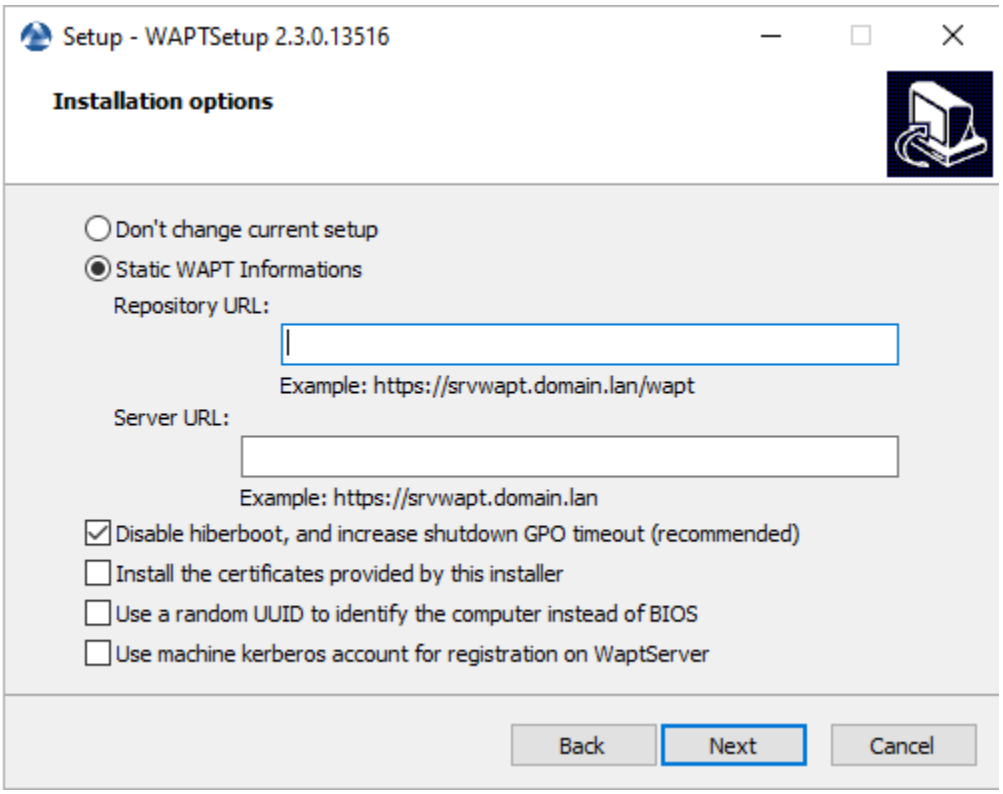
Table 2: Available options of the WAPT Agent installer

Settings	Description	Default value
Install WAPT service checkbox	Enables the WAPT service on this computer.	Checked
Launch notification icon upon session opening checkbox	Launches the WAPT Agent in systray on host startup.	Not checked
Disable hiberboot, and increase shutdown GPO timeout (recommended) checkbox	Disables Windows fast startup for stability, it increases the timeout for the WAPT Exit utility.	Checked
Install the certificates provided by this installer checkbox	Installs Tranquil IT certificate on this computer.	Not checked
Use a random UUID to identify the computer instead of BIOS checkbox	For more information, check the documentation on *BIOS UUID bugs*	Not checked

- Set up the WAPT Server URL.

Fisrt installation

- Check *Static WAPT Informations* and set:
  - WAPT repository URL: http://srvwapt.mydomain.lan/wapt.
  - WAPT Server URL: https://srvwapt.mydomain.lan.

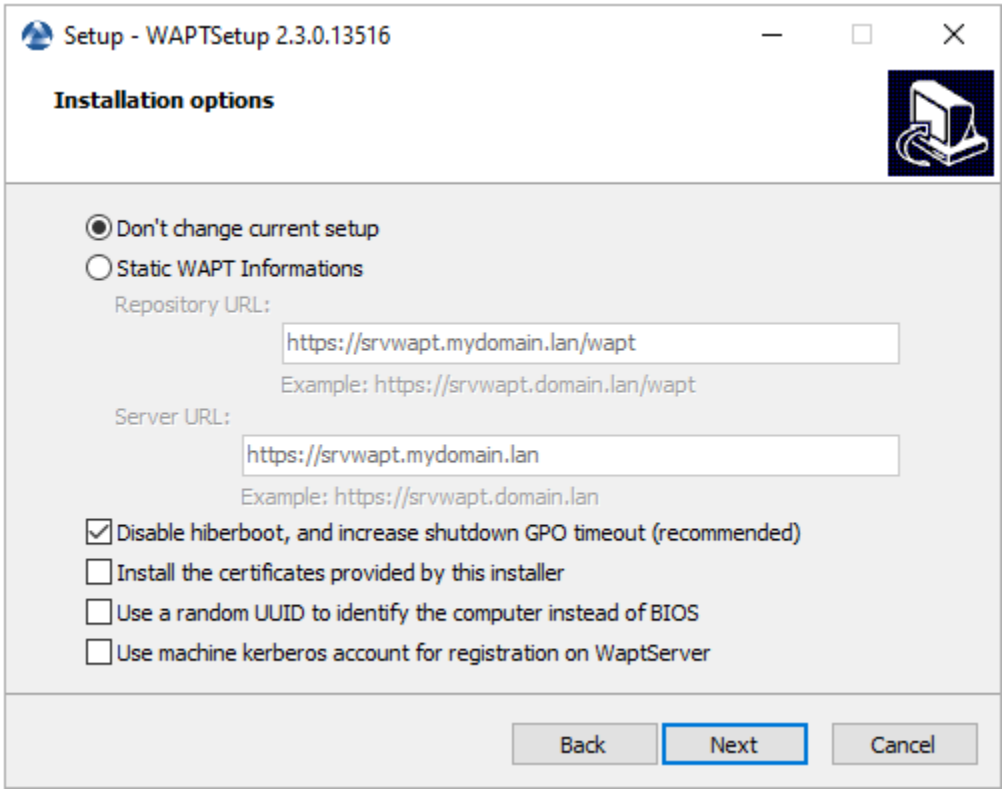


Choosing the WAPT repository and the WAPT Server

- Choose the WAPT repository and the WAPT Server; click *Next*.

Upgrade

- Check *Don't change current setup*, then click *Next*.



The WAPT repository and the WAPT Server are already set

- Get a summary of the WAPT Console installation.
- Click *Install* to launch the installation, wait for the installation to complete, then click on *Finish* (leave default options).

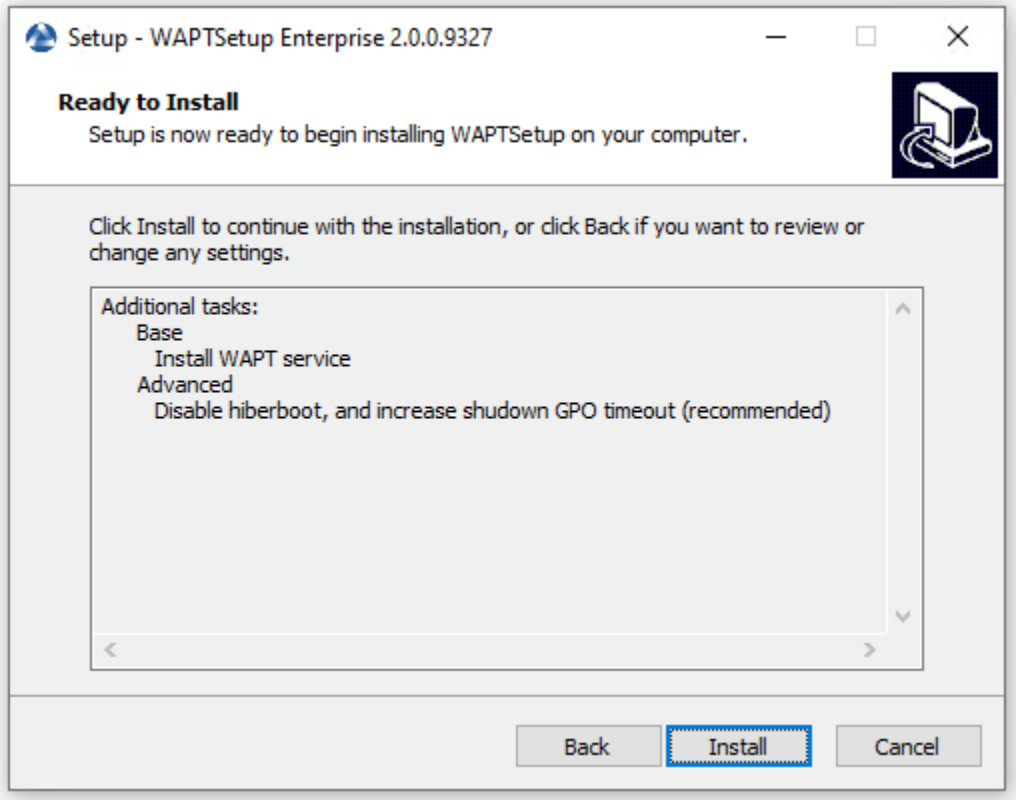
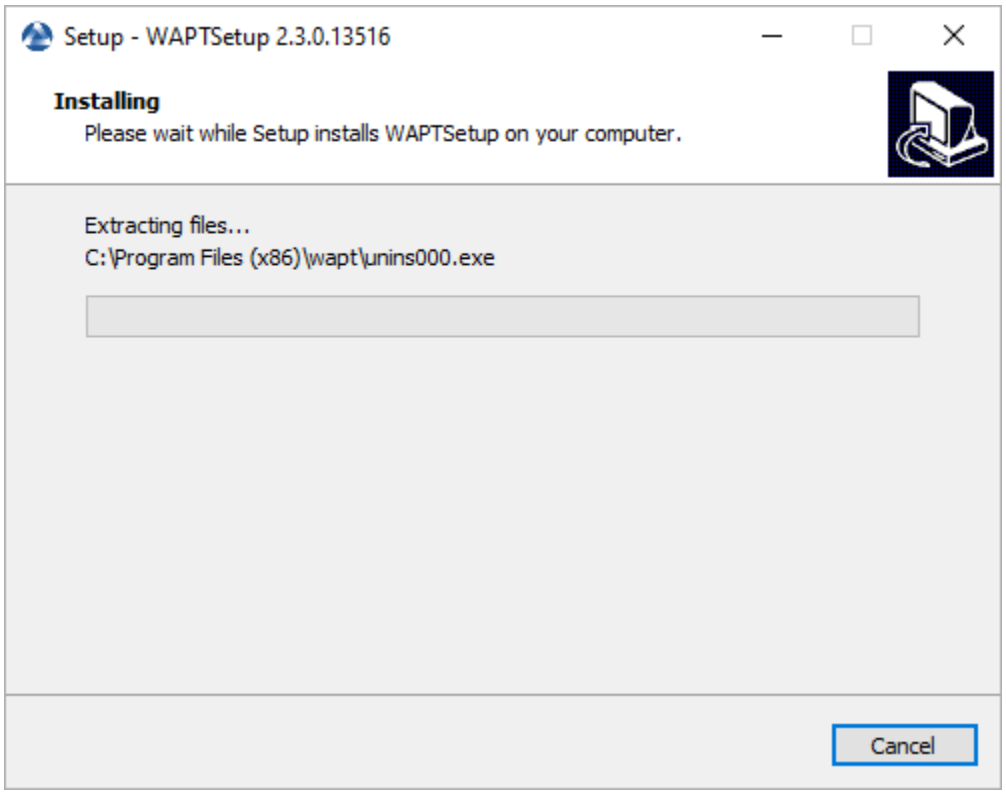
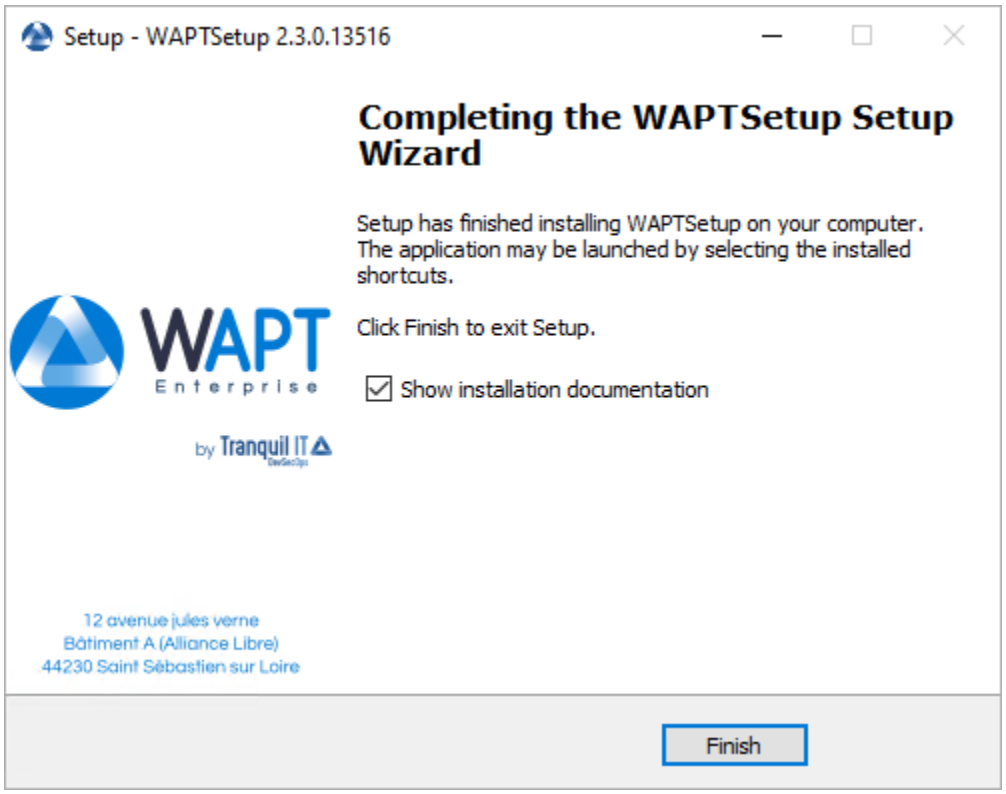


Fig. 8: Summary of the WAPT installation abstract





- Uncheck *Show installation documentation*.

Starting the WAPT Console

- Launch the WAPT Console:
  - By looking for the binary.

    `C:\Program Files (x86)\wapt\waptconsole.exe`
  - Or using the *Start* Menu.

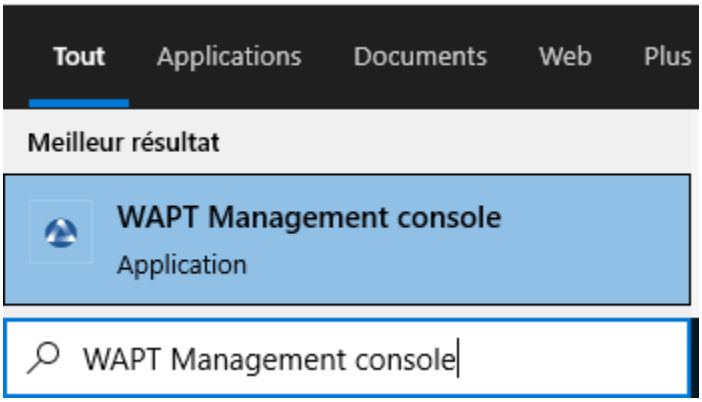


Fig. 9: Launching the WAPT Console from the Windows Start Menu

- Log into the WAPT Console with the *SuperAdmin* login and password.

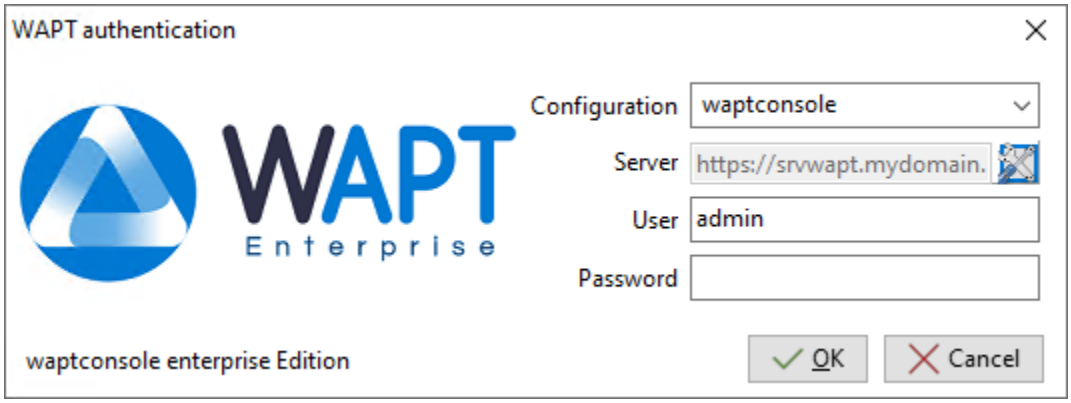


Fig. 10: The WAPT Console authentication window

If you have any issue logging into the WAPT Console, please refer to the FAQ: *Error message when opening the WAPT Console*.

It is recommended to launch the WAPT Console with a Local Administrator account to enable local debugging of WAPT packages.

For Enterprise version, it is possible to authenticate with *Active Directory*.

Danger: After the upgrade, please be sure your certificate (in this documentation **wapt-private.crt**) is still present at your WAPT install location : `C:\Program File (x86)\wapt\ssl`

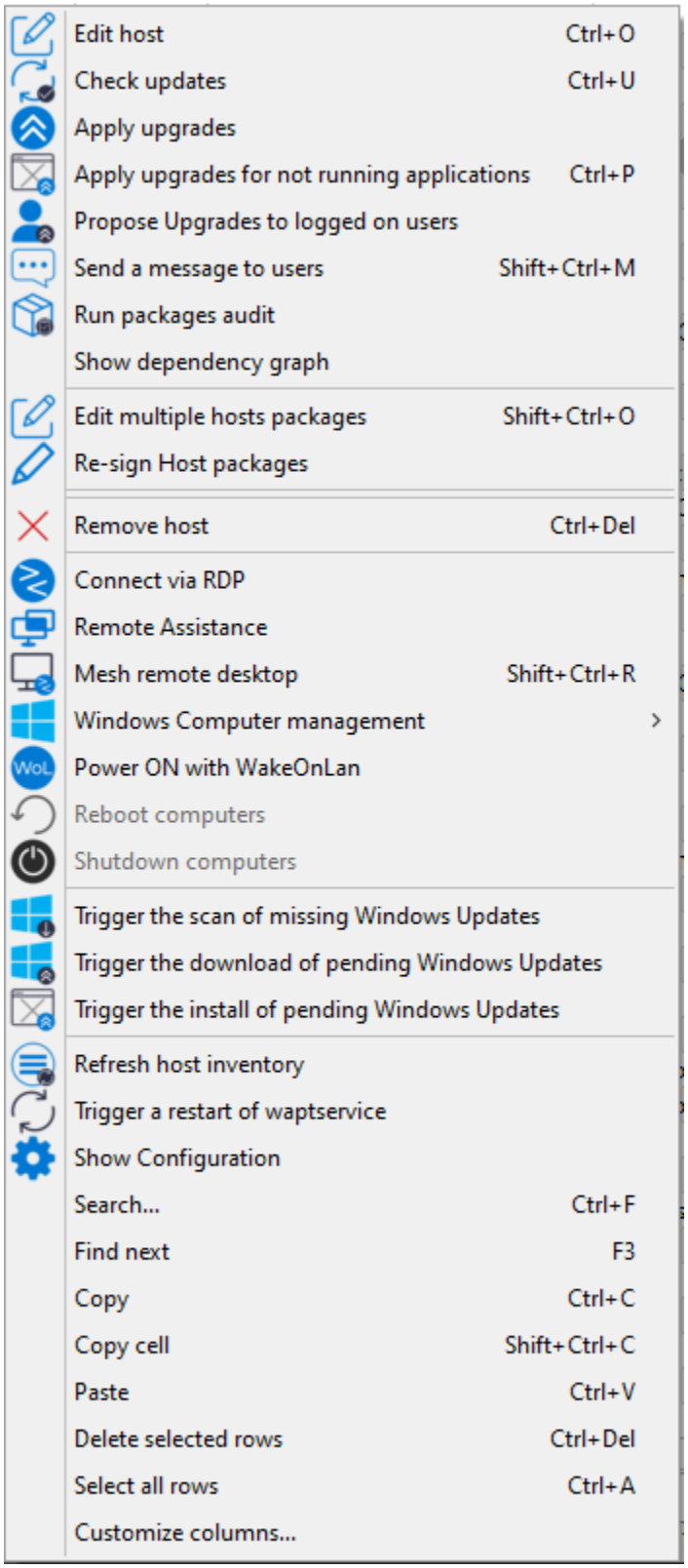
Since you come from WAPT 1.8.2 which was in python2, you'll have to re-sign all your WAPT Packages *using the WAPT Console*, or *using the command line* (**only if you encounter package size issue with the WAPT Console way**).

12.3.4 Re-signing Host packages

This method for re-signing all host packages is useful when the underlying cryptographic method or library changes, as this is the case when upgrading from WAPT 1.8.2 (Python 2.7 based) to WAPT >= 2.0 (Python 3.x based).

Use the Administrator's certificate for re-signing packages.

- Select all host.
- Right-click on the selected hosts.



- Select *Re-sign Host packages*.

- Confirm re-signing the selected hosts.

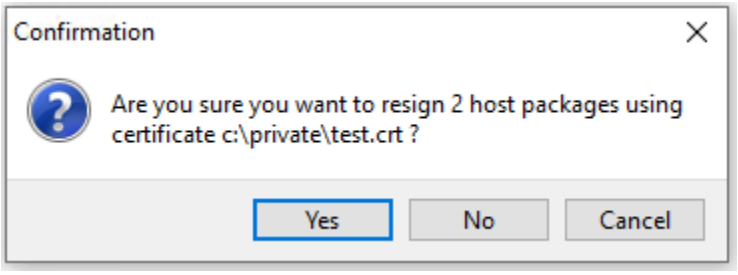


Fig. 11: Modal window for confirming re-signing the selection of hosts

- Then, enter you private key password.

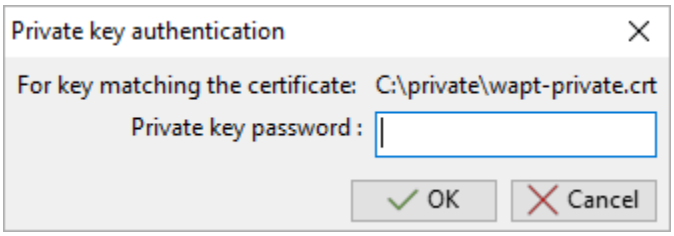


Fig. 12: Entering the password for unlocking the private key in the WAPT Console

- The selected WAPT *host* packages are now all re-signed using the new cryptographic method required with Python3.

12.3.5 Re-signing other types of WAPT package

- Open the repositories in your WAPT Console.
- Select all packages in the repository, then right-click on the selection.
- Select *Re-sign packages*.
- To launch the signature process, click on *Re-sign packages*.
- After processing, which may take some time, all packages will have been re-signed.

Attention: ⓘ | microsoft-office | 16.0.12325.20276-2 | PROD | ERROR | Access violation

If the error **Access violation** appear it may mean that the WAPT package is too big.

You can resign this packages *using the command line*.

And if it's still not working, you can still manually edit the package and visit *this procedure for signing large WAPT packages*.

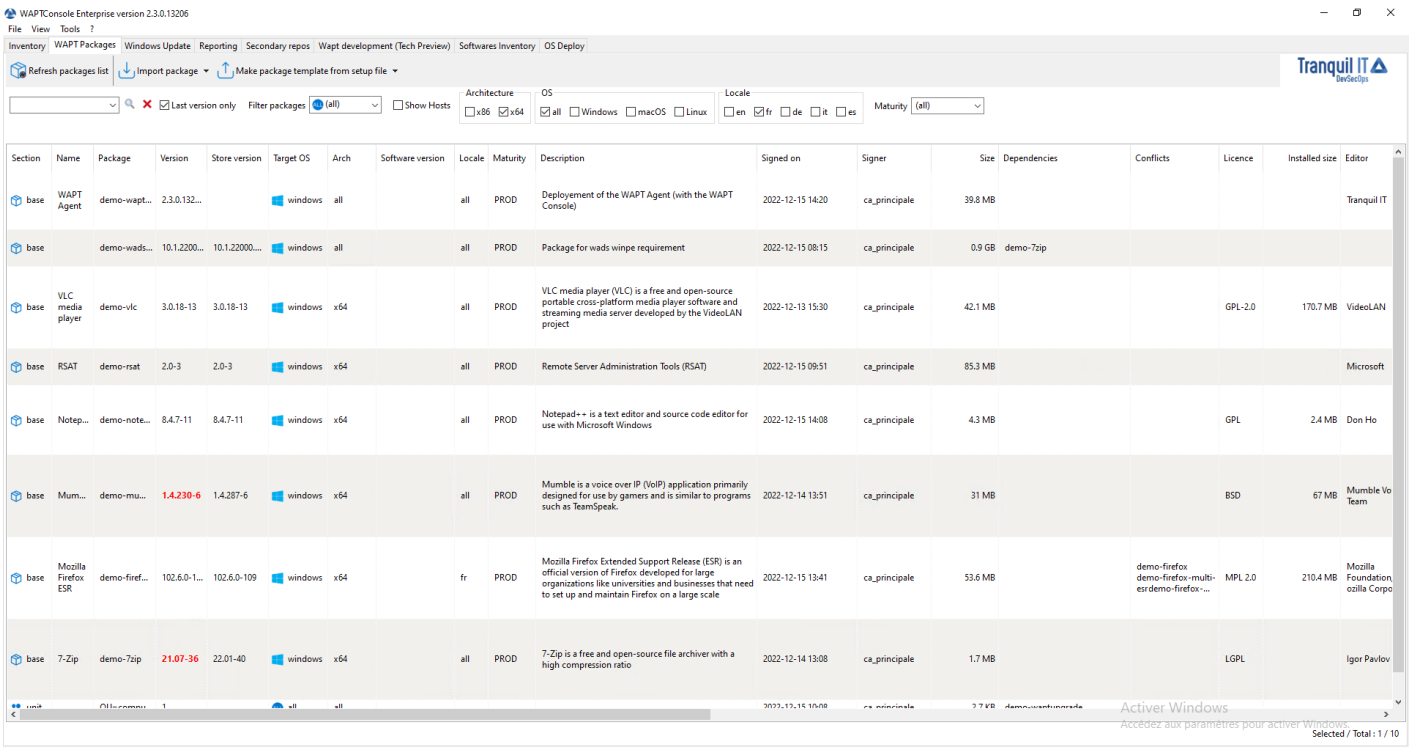


Fig. 13: Window showing the repositories available on the WAPT Console

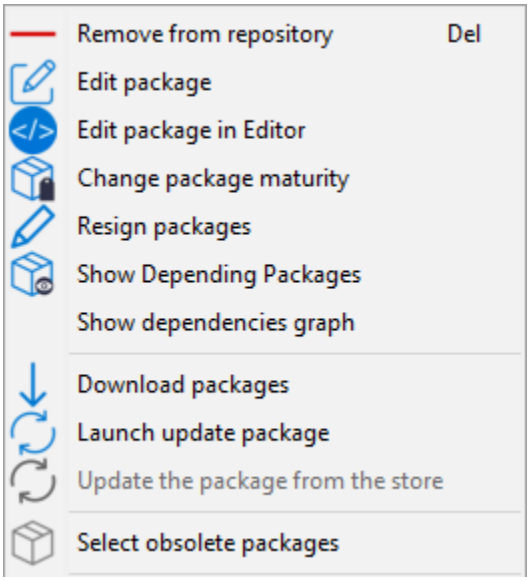


Fig. 14: Menu options for repositories

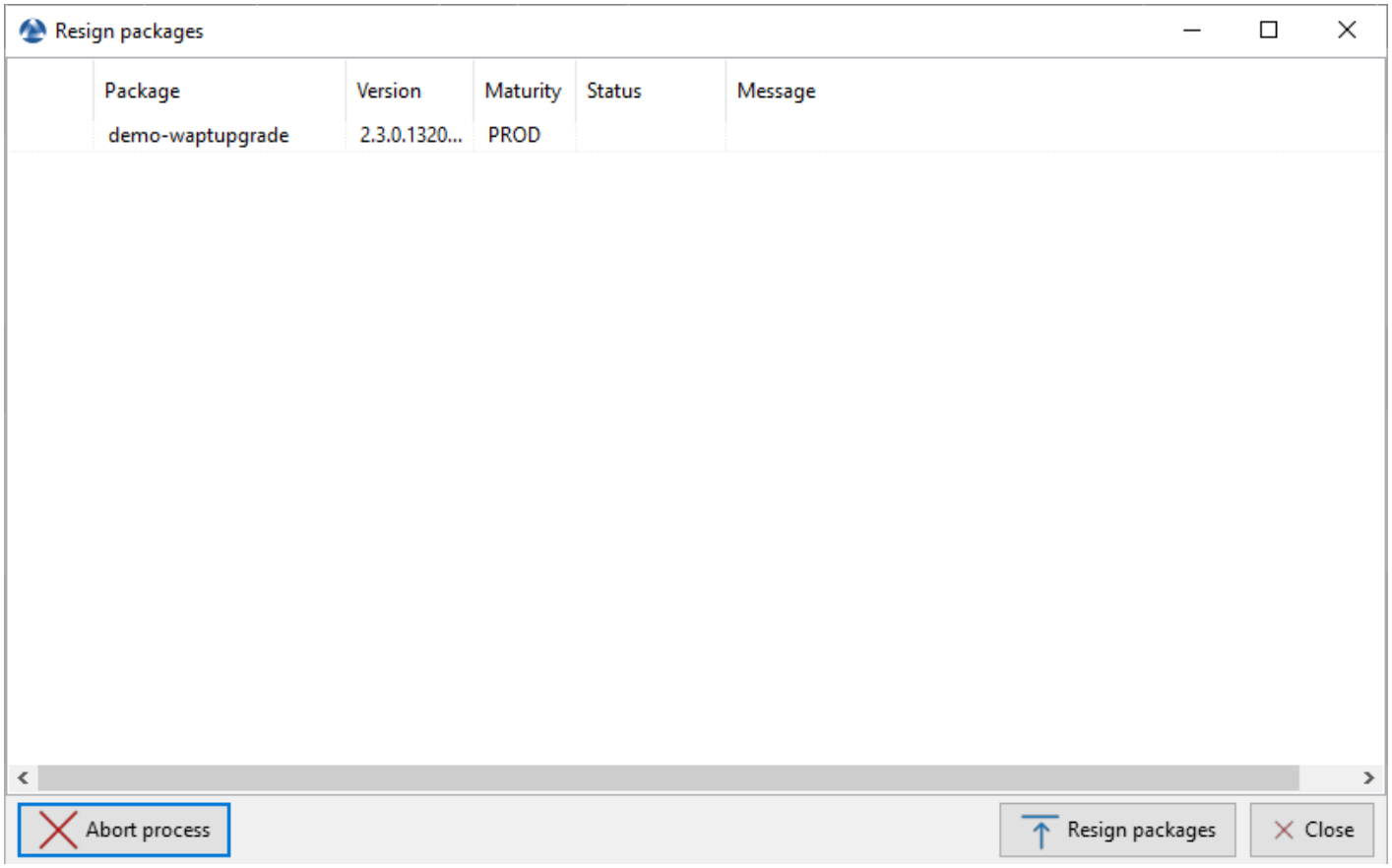


Fig. 15: Window for re-signing WAPT packages

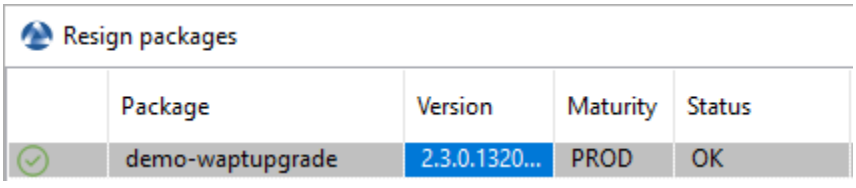


Fig. 16: Signature processing has ended successfully

CHAPTER

THIRTEEN

BACKING UP THE WAPT SERVER

To backup your WAPT Server follow this procedure. Regular backups are recommended.

13.1 Linux

- Stop WAPT related services on the WAPT Server.

```
systemctl stop wapttasks
systemctl stop waptserver
systemctl stop nginx
```

- Backup recursively these directories using a backup tool (ex: **rsync**, **WInSCP**, etc..).

Debian / Ubuntu

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
/var/www/wads/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

Centos / RedHat

```
/var/www/html/wapt/
/var/www/html/wapt-host/
/var/www/html/waptwua/
/var/www/html/wads/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

Hint: If you use Kerberos to authenticate hosts and users, save the `keytab` file too. The `keytab` file is located in the `nginx` folder.

- Backup the PostgreSQL database using the **pg_dumpall** utility (adapt filename with your requirements).

```
sudo -u postgres pg_dumpall > /tmp/backup_wapt.sql
```

- Restart WAPT related services on the WAPT Server.

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

13.2 Windows

- Stop WAPT related services on the WAPT Server.

```
net stop wapttasks
net stop waptserver
net stop waptnginx
```

- Backup the WAPT repository folder on a remote backup destination.

```
C:\wapt\conf
C:\wapt\waptserver\repository\wapt
C:\wapt\waptserver\repository\wapt-host
C:\wapt\waptserver\repository\waptwua
C:\wapt\waptserver\repository\wads
C:\wapt\waptserver\nginx\ssl
```

- Backup PostgreSQL Database with **pg_dump.exe**.

```
"C:\wapt\waptserver\pgsql-14\bin\pg_dumpall.exe" -U postgres -f C:\backup_wapt.sql
```

- Restart WAPT related services on the WAPT Server.

```
net start wapttasks
net start waptserver
net start waptnginx
```

CHAPTER

FOURTEEN

RESTORING THE WAPT SERVER

In case of a complete crash, restart a standard WAPT Server installation on your WAPT Server. Then follow this procedure to restore your data.

14.1 Linux

- Stop WAPT related services on the WAPT Server.

```
systemctl stop nginx
systemctl stop waptserver
systemctl stop wapttasks
```

- Restore the following directories.

Debian / Ubuntu

```
/var/www/wapt/
/var/www/wapt-host/
/var/www/waptwua/
/var/www/wads/
/opt/wapt/conf/
/opt/wapt/waptserver/ssl/
```

Centos / RedHat

```
/var/www/html/wapt/
/var/www/html/wapt-host/
/var/www/html/waptwua/
/var/www/html/wads/
/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!

Warning: Check the encoding before creating the wapt database, if the dumped file is in en_US, your new base has to be en_US.

```
sudo -u postgres psql -c "drop database wapt"
sudo -u postgres psql -c "create database wapt"
sudo -u postgres psql < /tmp/backup_wapt.sql
```

- Apply ownership rights to the restored folders.

Debian / Ubuntu

```
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:www-data /var/www/wads/
chown -R wapt /opt/wapt/conf/
chown -R wapt /opt/wapt/waptserver/ssl/
```

CentOS / RedHat

```
chown -R wapt:nginx /var/www/html/wapt/
chown -R wapt:nginx /var/www/html/wapt-host/
chown -R wapt:nginx /var/www/html/waptwua/
chown -R wapt:nginx /var/www/html/wads/
chown -R wapt /opt/wapt/conf/
chown -R wapt /opt/wapt/waptserver/ssl/
```

- Run a Scanpackages on your repositories.

Debian / Ubuntu

```
wapt-scanpackages /var/www/wapt/
```

CentOS / RedHat

```
wapt-scanpackages /var/www/html/wapt/
```

- Restart WAPT related services on the WAPT Server.

```
systemctl start wapttasks
systemctl start waptserver
systemctl start nginx
```

14.2 Windows

- Stop WAPT related services on the WAPT Server.

```
net start wapttasks
net start waptserver
net start waptnginx
```

- Restore the following directories.

```
C:\wapt\waptserver\repository\wapt
C:\wapt\waptserver\repository\wapt-host
C:\wapt\waptserver\repository\waptwua
C:\wapt\waptserver\repository\wads
C:\wapt\waptserver\conf
C:\wapt\waptserver\nginx\ssl
```

- Apply full rights to the folder `C:\wapt\waptserver\repository` for the "Network Service" group.
- Restore PostgreSQL Database with **`pg_restore.exe`**.

```
"C:\wapt\waptserver\pgsql-14\bin\psql.exe" -f c:\backup_wapt.sql -U postgres
```

- Scan package repositories.

```
wapt-scanpackages "C:\wapt\waptserver\repository\wapt"
```

- Restart WAPT related services on the WAPT Server.

```
net start wapttasks
net start waptserver
net start waptnginx
```

CHAPTER

FIFTEEN

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 API 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.

15.1 API V1

15.1.1 /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 hosts based on their trusted package certs

# Returns:
#     result (dict): {'records':[],'files':[]}
#     query:
#       uuid=<uuid>
#     or
#       filter=<csvlist of fields>:regular expression
# """
```

- List all hosts using the following parameters:

- *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. The display limit is 10000.

```
https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v1/hosts?columns=reachable,computer_
↪fqdn,connected_ips,mac_addresses&limit=10000
```

15.1.2 /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
```

15.1.3 /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

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=UUID&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=UUID&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=UUID&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 Updates installed on a specific host.

```
wsusupdates_data_wapt = wgets('https://%s:%s@%s/api/v1/host_data?uuid=UUID&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
```

15.1.4 /api/v1/usage_statistics

Get usage statistics from the WAPT 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
```

15.2 API V2

15.2.1 /api/v2/waptagent_version

Display **waptagent.exe** version.

```
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
```

15.3 API V3

15.3.1 /api/v3/packages

List packages on the repository, retrieve the `control` files of WAPT packages.

```
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
```

15.3.2 /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
```

15.3.3 /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))
```

15.3.4 /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
```

15.3.5 /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
```

15.3.6 /api/v3/hosts_for_package

List hosts with a specific package installed.

```
hosts_for_package = wgets('https://%s:%s@%s/api/v3/hosts_for_package?package=PACKAGE' % (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
```

15.3.7 /api/v3/host_tasks_status

List tasks on a particular host.

```
host_tasks_status = wgets('https://%s:%s@%s/api/v3/host_tasks_status?uuid=UUID' % (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**.

15.3.8 /api/v3/upload_packages

Todo: Tests

15.3.9 /api/v3/upload_hosts

Todo: Tests

15.3.10 /api/v3/change_password

Change the Admininistrator password [only this account]. The request is a python dictionnary *{}* with keys:

- *user*;
- *old_password*;
- *new_password*.

```
curl --insecure -X POST --data-raw '{"user":"USER","password":"old_password","new_password":"new_
↪password"}' -H "Content-Type: application/json" "https://user:old_password@srvwapt/api/v3/change_
↪password"
```

15.3.11 /api/v3/login

Initialize a connection to the WAPT 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 then POST: https://admin:MYPASSWORD@srvwapt.mydomain.lan/api/v3/get_ad_sites

15.3.12 /api/v3/packages_delete

Delete package with a precise version. The request is a python list *[]*. A list of packages may be given, 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"
```

15.3.13 /api/v3/reset_hosts_sid

Reinitialize all host connections.

For the POST method, the syntax is: `--data-raw` a dictionnary list with `uuids` as keys and the UUIDs of the hosts as values.

```
curl --insecure -X POST --data-raw '{"uuids":["UUID"]}' -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":["UUID#1","UUID#2"]}' -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}
```

15.3.14 /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":["UUID"]}' -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": "computer_fqdn", "mac_
↪addresses": ["mac_addresses"], "uuid": "UUID"}], "success": true, "request_time": null}
```

Hint: If you want several hosts:

```
curl --insecure -X POST --data-raw '{"uuids":["UUID#1","UUID#2"]}' -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": "computer_fqdn#1",
↪"mac_addresses": ["mac_addresses#1"], "uuid": "UUID#1"}, {"computer_fqdn": "computer_fqdn#2",
↪"mac_addresses": ["mac_addresses#2"], "uuid": "UUID#2"}], "success": true, "request_time": null}
```

15.3.15 /api/v3/hosts_delete

```
"""Remove one or several hosts from the WAPT Server database 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":["UUID"],"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/UUID.wapt"], "records": [{"computer_fqdn": "computer_fqdn", "uuid": "UUID"}]},
↪"success": true, "request_time": null}
```

If you do not want to delete from the inventory:

```
curl --insecure -X POST --data-raw '{"uuids":["UUID"],"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": "computer_fqdn", "uuid": "UUID"}]}, "success": true, "request_time
↪": null}
```

15.3.16 /api/v3/trigger_host_action

Todo: Tests

15.3.17 /api/v3/upload_waptsetup

```
# Upload waptsetup

#Handle the upload of customized waptagent.exe into wapt repository

### DOES NOT WORK
#curl --insecure -X POST -H "Content-Type: multipart/form-data" -F 'data=@waptagent.exe' "https://
↪admin:MYPASSWORD@srvwapt.mydomain.lan/upload_waptsetup"
```

15.3.18 /api/v3/ping

Ping shows a general set of informations on a WAPT Server.

```
# https://srvwapt.mydomain.lan/ping
# Lists WAPT Server informations

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))
```

CHAPTER

SIXTEEN

WAPT SERVER ADVANCED CONFIGURATION

The WAPT Server configuration file on GNU/ Linux and macOS systems is found in `/opt/wapt/conf/waptserver.ini` or in `/opt/wapt/waptserver/waptserver.ini`.

The WAPT Server configuration file on Windows is found in `C:\wapt\conf\waptserver.ini`.

Attention: Modification of these files is reserved for advanced users!!

16.1 Default configurations of waptserver file and nginx

16.1.1 Modify the [options] section of waptserver.ini

Several options can be defined in the [options] section.

```
[options]
```

Table 1: Available parameters for the [options] section of `waptserver.ini`

Options (Default Value)	Description	Example
`agents_folder` (default `watpagent in wapt repository`)	Defines where the W
`allow_unauthenticated_connect` (default `None`)	Defines whether web
`allow_unauthenticated_registration` (default `False`)	Allows the initial reg
`allow_unsigned_status_data` (default `False`)	Debug only - Allows
`application_root` (default `None`)	Defines a custom WA
`authentication_logs` (default `True`)	Enables authenticati
`auto_create_waptagent_from_config` (default `False`)	Enables automatic c
`client_certificate_lifetime` (default `3650`)	Defines the host cert
`cleanup_kbs` (default `True`)	Defines whether unu
`clients_read_timeout` (default `5`)	Defines the websock
`clients_signing_certificate` (default `None`)	Defines the host cert
`clients_signing_crl_days` (default `30`)	Defines the host cert
`clients_signing_crl` (default `None`)	Defines the host cert
`clients_signing_crl_url` (default `None`)	Defines the host cert
`clients_signing_key` (default `None`)	Defines the host cert
`client_tasks_timeout` (default `5`)	Defines the maximu

continues on next page

Table 1 – continued from previous page

Options (Default Value)	Description	Example
copy_winpe_x64_in_tftp_folder (default False)	If x64, allows you to	
db_connect_timeout (default 3)	Defines the maximu	
db_host (default None)	Defines the url of the	
db_max_connections (default 90)	Defines the maximu	
db_name (default wapt)	Defines the PostgreS	
db_password (default None)	Defines the passwor	
db_port (default 5432)	Defines the port of t	
db_stale_timeout (default 300)	Defines the database	
db_user (default wapt)	Defines the PostgreS	
default_ldap_users_acls (default view)	Defines the default a	
download_wsusscn2 (default False)	Automatically down	
enable_store (default False)	Enables WAPT Store	
encrypt_host_packages (default False)	Encrypts host packag	
htpasswd_path (default None)	Adds basic authentic	
http_proxy (default None)	Defines the proxy se	
known_certificates_folder (default WAPT /ssl/ folder)	Adds additional kno	
ldap_account_service_login (default None)	Defines the UPN Ac	
ldap_account_service_password (default None)	Defines the user pass	
ldap_auth_base_dn (default None)	Defines the LDAP au	
ldap_auth_server (default None)	Defines the LDAP au	
ldap_auth_ssl_enabled (default True)	Sets SSL authenticat	
ldap_nesting_group_support (default True)	Enables the search o	
ldap_primary_group_ad_support (default True)	Enables the search o	
list_subnet_skip_login_wads (default [])	Lists subnets withou	
login_on_wads (default False)	Enables authenticati	
loglevel (default warning)	Defines the log level	
max_clients (default 4096)	Sets the maximum si	
min_password_length (default 10)	Sets the minimum *S*	
nginx_http (default 80)	Defines the Nginx w	
nginx_https (default 443)	Defines the Nginx w	
optimized_authentication_logs (default True)	If one of the option i	
remote_repo_update_delay (default 1)	Défines the periodici	
remote_repo_websockets (default True)	Enables websocket c	
secret_key (default None)	Defines the random s	
server_uuid (default None)	Defines the WAPT S	
session_lifetime (default 126060)	Defines the maximu	
signature_clockskew (default 300)	Defines the maximu	
token_lifetime (default 43200)	Defines the authentic	
trusted_signers_certificates_folder (default None)	Defines the path to t	
trusted_users_certificates_folder (default None)	Defines the path to t	
use_kerberos (default False)	Enables a WAPT Ag	
use_ssl_client_auth (default False)	Enables *client certifi*	
verify_cert_ldap (default True)	Verifies whether the	
wads_enable (default False)	Enables the WADS f	
wads_folder (default wads folder in wapt repository)	Defines the folder on	
wapt_admin_group_dn (default None)	Defines the LDAP D	
wapt_admin_group (default None)	Defines the sAMAcc	

continues on next page

Table 1 – continued from previous page

Options (Default Value)	Description	Example
wapt_folder (default /var/www/wapt or /var/www/html/wapt or root_dir/waptserver/repository/wapt)	Defines the directory	
wapt_huey_db (default None)	Defines the path to d	
wapt_password (default None)	Defines the *SuperAd*	
waptserver_port (default 8080)	Defines the WAPT S	
wapt_user (default admin)	Defines the *SuperAd*	
waptwua_folder (default wapt_folder + 'wua')	Defines the location	
wol_port (default 7,9)	Defines the list of W	
wapt_bind_interface (default 127.0.0.1)	Defines how to lister	
ipxe_script_jinja_path (default /opt/wapt/waptserver/templates/ipxe-default.j2)	Defines the location	

16.1.2 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;
    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;
    }
```

(continues on next page)

(continued from previous page)

```
  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;
    }
  }
}
```

16.2 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 host. However the Nginx part scales very well and it can takes full advantage of a 10Gbps connection for high load package deployments.

Note: The parameters to be modified below are linked together and should be modified globally and not individually.

16.2.1 Configuring Nginx

Table 2: `nginx.conf` configuration file location

OS Type	File location
Debian / Ubuntu	`/etc/nginx/nginx.conf`
Redhat and derivatives	`/etc/nginx/nginx.conf`
Windows	`C:\wapt\waptserver\nginx\conf\nginx.conf`

In the `nginx.conf` file, modify the `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 `nginx.conf` file:

```
worker_rlimit_nofile 32768;
```

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 WAPT Server 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
```

16.2.2 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 should be 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
```

16.2.3 Configuring the PostgreSQL database

Table 3: `postgresql.conf` configuration file location

OS Type	File location
Debian / Ubuntu	`/etc/postgresql/{version}/main/postgresql.conf`
Redhat and derivatives	`/var/lib/pgsql/{version}/data/postgresql.conf`
Windows	`C:\wapt\waptserver\pgsql{version}_data\postgresql.conf`

A higher number of clients need a higher number of connections to the PostgreSQL database. In the `postgresql.conf` file, you need to increase the following parameter to approximately 1/4 the number of active WAPT Agents.

```
max_connections = 1000
```

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
```

16.3 Using the command-lines for repository management

16.3.1 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.wapt`** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
Uploading packages to https://srvwapt.mydomain.lan
Please get login for https://srvwapt.mydomain.lan/api/v3/upload_xxx:admin
Password:
c:\waptdev\tis-tightvnc.wapt[==============================] 54316019/54316019 - 00:00:17
OK : 1 Packages uploaded, 0 errors
```

16.3.2 wapt-get scan-packages

Hint: This command applies to Windows repositories **ONLY**.

The **wapt-get scan-packages <directory>** command rebuilds a `Packages` file for a WAPT package repository.

The command **wapt-get scan-packages C:waptwaptserverrepositorywapt** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Packages filename: C:\wapt\waptserver\repository\wapt
Processed packages:
  C:\wapt\waptserver\repository\wapt\tis-firefox.wapt
  C:\wapt\waptserver\repository\wapt\tis-tightvnc.wapt
  C:\wapt\waptserver\repository\wapt\tis-7zip.wapt
Skipped packages:
```

16.3.3 wapt-scanpackages

Hint: This command applies to Linux repositories **ONLY**.

The **wapt-scanpackages <directory>** command rebuilds a `Packages` file for a WAPT package repository.

The command **wapt-scanpackages /var/www/wapt/** returns nothing.

16.3.4 Re-signing packages on the WAPT Server using a command line

Use this method if re-signing from the WAPT console method does not complete successfully. These commands are **ONLY** available for WAPT Servers running Linux.

Warning: Before using this method, ensure that your WAPT Server is safe and not under the control of an unauthorized third party entity.

- Copy your `.crt` and `.pem` to `/tmp/` on the WAPT Server using **Winscp** or an equivalent tool.
- It is then possible to re-sign all the packages at once on the WAPT Server with the following commands.

```
wapt-signpackages -d /var/www/wapt-host -c /tmp/wapt_pub_key.crt -k /tmp/wapt_priv_key.pem -s
wapt-signpackages -d /var/www/wapt -c /tmp/wapt_pub_key.crt -k /tmp/wapt_priv_key.pem -s
wapt-scanpackages /var/www/wapt/
```

If the error **Access violation** appears, the reason is that the WAPT package is too voluminous.

Edit the package and check *this procedure to transfert a voluminous package*.

Danger: Remove the *.crt* and *.pem* from `/tmp/` on the WAPT Server or the server will become a sensitive asset.

For more available options, please see the *command line section*.

CHAPTER

SEVENTEEN

ENHANCING THE SECURITY OF YOUR WAPT SETUP - SERVER SIDE

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 WAPT Agents and the WAPT 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)).
- If you are using the **Enterprise** version of WAPT and you operate a large fleet with multiple administrators, you may be interested in knowing how to properly configure and apply the ACLs.

17.1 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.

17.1.1 Configuring the firewall for WAPT Server on Debian / Ubuntu

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
```

17.1.2 Configuring the firewall for WAPT Server on RedHat and derivatives

- 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
```

17.2 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 host with the WAPT Server* and *signing inventory updates*.
- The kerberos authentication will be used only when registering the device.

17.2.1 Installing the kerberos components and configuring krb5.conf file

Debian / Ubuntu

```
apt install krb5-user msktutil libnginx-mod-http-auth-spnego
```

CentOS / RedHat

```
yum install krb5-workstation msktutil nginx-mod-http-auth-spnego
```

Note: Registering with kerberos is not available with a WAPT Server running on Windows.

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` value **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 **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 Mon. 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. If your WAPT server is available from the internet, you should add another servicePrincipalName (SPN) to match with the WAPT public URL. In order to update the keytab file, you must run the 2nd msktutil command every time you add a new SPN.

- Apply the proper access rights to the `http-krb5.keytab` file. If you are with Redhat based OS with selinux, please fix rights with **restorecon**.

Debian / Ubuntu

```
sudo chmod 640 /etc/nginx/http-krb5.keytab
sudo chown root:www-data /etc/nginx/http-krb5.keytab
```

CentOS / RedHat

```
sudo chown root:nginx /etc/nginx/http-krb5.keytab
sudo chmod 640 /etc/nginx/http-krb5.keytab
restorecon -v -R /etc/nginx/http-krb5.keytab
```

17.2.2 Post-configuring kerberos for the WAPT Server

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.

17.2.3 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. If you are with Redhat based OS with selinux, please fix rights with **restorecon**.

Debian / Ubuntu

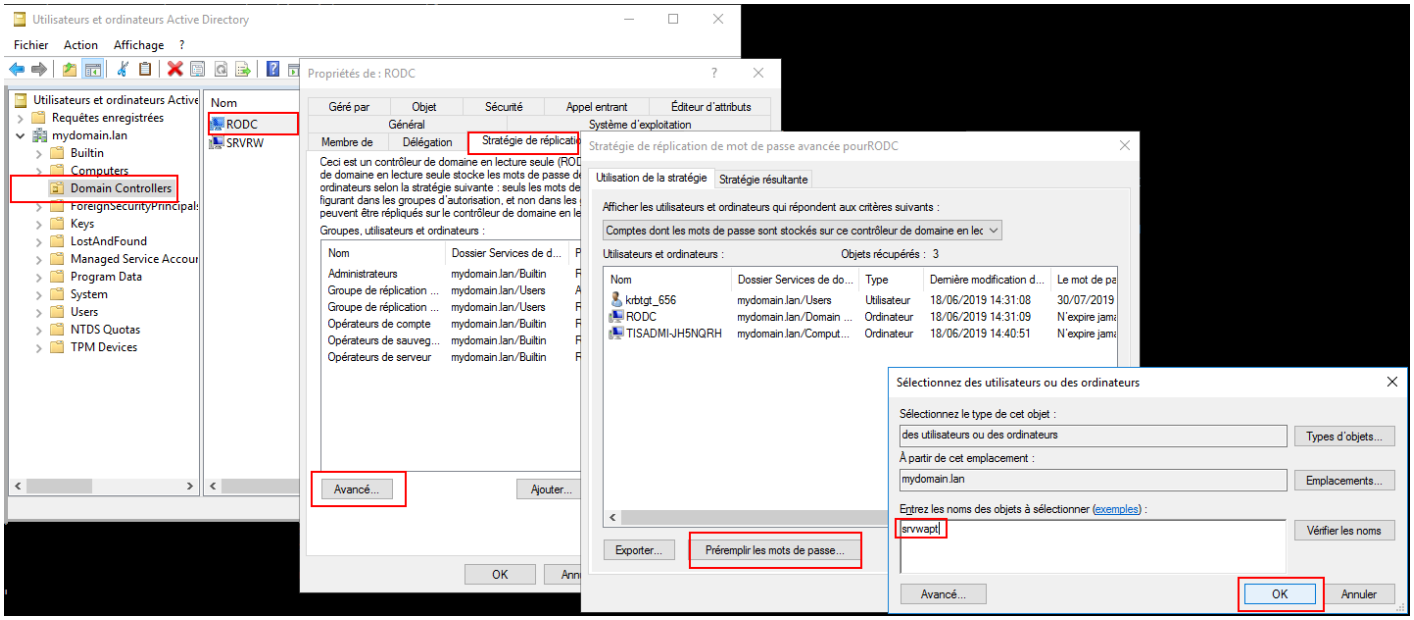
```
sudo chmod 640 /etc/nginx/http-krb5.keytab
sudo chown root:www-data /etc/nginx/http-krb5.keytab
```

CentOS / RedHat

```
sudo chown root:nginx /etc/nginx/http-krb5.keytab
sudo chmod 640 /etc/nginx/http-krb5.keytab
restorecon -v -R /etc/nginx/http-krb5.keytab
```

WAPT Agents 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 relationships

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`:

```
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
```

17.2.4 Debug problems with the kerberos

Attention:

- The WAPT Server address cannot be an IP, Kerberos works well only with DNS.
- In your test, the url used **MUST** be **exactly** the same address as the one indicated in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Did you restart nginx correctly?

```
systemctl restart nginx
```

Check the permissions of the http-krb5.keytab file

```
[root@srvwapt.mydomain.lan]# ls -l /etc/nginx/http-krb5.keytab
-rw-r----- 1 root www-data 921 janv. 4 16:20 /etc/nginx/http-krb5.keytab
```

Is kerberos mode active on my WAPT Agent?

On the Windows host:

- Check in your `C:\Program Files (x86)\wapt\wapt-get.ini` that the `use_kerberos` value is `True`.

```
[global]
use_kerberos=True
```

- If you change the value, do not forget to restart the WAPT service.

```
net stop waptservice
net start waptservice
```

Is Kerberos mode active on my WAPT Server?

On the Linux host:

- Check in your `/opt/wapt/conf/waptserver.ini` that the `use_kerberos` value is `True`.

```
[options]
use_kerberos=True
```

- Check in your `/etc/nginx/sites-enabled/wapt.conf` that this configuration is present.

```
location ~ ^/.*_kerberos$ {

      proxy_http_version 1.1;
      proxy_request_buffering off;
```

(continues on next page)

(continued from previous page)

```
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    # be sure these headers are not forwarded
    proxy_set_header X-Ssl-Client-Dn "";
    proxy_set_header X-Ssl-Authenticated "";

    auth_gss on;
    auth_gss_keytab /etc/nginx/http-krb5.keytab;
    proxy_pass http://127.0.0.1:8080;

}
```

- If one of the two configurations is not present, restart the post-configuration and activate kerberos.

Checking that the keytab file contains the correct url

```
[root@srvwapt.mydomaine.lan]# KRB5_KTNAME=/etc/nginx/http-krb5.keytab klist -k
Keytab name: FILE:/etc/nginx/http-krb5.keytab
KVNO Principal
---- --------------------------------------------------------------------------
   ...
   3 HTTP/srvwapt.ad.mydomain.lan@AD.MYDOMAIN.LAN
   ...
```

Trying to register the host using a system account

To switch to a system account you **MUST** use the **psexe** tool from Microsoft: `psexe`.

- In **cmd** as an Administrator.

```
C:\Users\\xxxxxx\\Downloads\\PSTools\\psexec.exe -accepteula -s -i cmd
```

- In the new **cmd** window, check that you are identified as *System*.

```
C:\WINDOWS\\system32>whoami

NT AUTHORITY\System
```

- Run **register**.

```
wapt-get register
```

Trying an authentication with the keytab from your WAPT Server

- On the Linux host.

```
[root@srvwapt.ad.tranq ~]# ktutil
ktutil: read_kt /etc/nginx/http-krb5.keytab
ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
   1    3                   srvwapt$@AD.TRANQUIL.IT
   2    3                   srvwapt$@AD.TRANQUIL.IT
   3    3                   srvwapt$@AD.TRANQUIL.IT
   4    3                   SRVWAPT$@AD.TRANQUIL.IT
   5    3                   SRVWAPT$@AD.TRANQUIL.IT
   6    3                   SRVWAPT$@AD.TRANQUIL.IT
   7    3               host/srvwapt@AD.TRANQUIL.IT
   8    3               host/srvwapt@AD.TRANQUIL.IT
   9    3               host/srvwapt@AD.TRANQUIL.IT
  10    3 HTTP/srvwapt.ad.tranquil.it@AD.TRANQUIL.IT
  11    3 HTTP/srvwapt.ad.tranquil.it@AD.TRANQUIL.IT
  12    3 HTTP/srvwapt.ad.tranquil.it@AD.TRANQUIL.IT
ktutil: quit
[root@srvwapt.ad.tranq ~]# kinit -k -t /etc/nginx/http-krb5.keytab srvwapt\$@AD.TRANQUIL.IT
[root@srvwapt.ad.tranq ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: srvwapt$@AD.TRANQUIL.IT

Valid starting       Expires              Service principal
05/02/2021 19:06:05  06/02/2021 05:06:05  krbtgt/AD.TRANQUIL.IT@AD.TRANQUIL.IT
  renew until 06/02/2021 19:06:05
```

Verifying that you are successfully obtaining a Kerberos ticket

Attention: Always execute commands in system account (see previous point)!

```
klist purge
klist get http/srvwapt.ad.mydomain.lan
```

You should get (in your language):

```
C:\Windows\System32>klist get http/srvwapt.ad.mydomain.lan

LogonId est 0:0x13794d
Un ticket pour http/srvwapt.ad.mydomain.lan a été récupéré.

Tickets mis en cache : (2)

#0> Client : sfonteneau @ AD.MYDOMAIN.LAN
```

(continues on next page)

(continued from previous page)

```
  Serveur : krbtgt/AD.MYDOMAIN.LAN @ AD.MYDOMAIN.LAN
  Type de chiffrement KerbTicket : AES-256-CTS-HMAC-SHA1-96
  Indicateurs de tickets 0x40e00000 -> forwardable renewable initial pre_authent
  Heure de démarrage : 2/4/2021 15:51:07 (Local)
  Heure de fin :   2/5/2021 1:51:07 (Local)
  Heure de renouvellement : 2/11/2021 15:51:07 (Local)
  Type de clé de session : AES-256-CTS-HMAC-SHA1-96
  Indicateurs de cache : 0x1 -> PRIMARY
  KDC appelé : srvads.AD.MYDOMAIN.LAN

#1> Client : sfonteneau @ AD.MYDOMAIN.LAN
  Serveur : http/srvwapt.AD.MYDOMAIN.LAN @ AD.MYDOMAIN.LAN
  Type de chiffrement KerbTicket : AES-256-CTS-HMAC-SHA1-96
  Indicateurs de tickets 0x40a80000 -> forwardable renewable pre_authent 0x80000
  Heure de démarrage : 2/4/2021 15:51:07 (Local)
  Heure de fin : 2/5/2021 1:51:07 (Local)
  Heure de renouvellement : 2/11/2021 15:51:07 (Local)
  Type de clé de session : AES-256-CTS-HMAC-SHA1-96
  Indicateurs de cache : 0
  KDC appelé : srvads.AD.MYDOMAIN.LAN
```

If that does not work, check in your Active Directory that the `serviceprincipalname` attribute on the computer account of the WAPT Server has this value: `HTTP/srvwapt.mydomain.lan`.

Check that it works with Firefox

Note: You need to first configure Firefox for kerberos authentication.

- Type **`about:config`** in the URL bar in your Firefox.
- Edit `network.negotiate-auth.trusted-uris`, and add the url of the WAPT Server: `srvwapt.mydomain.lan`.
- You can now visit the url: https://srvwapt.mydomain.lan/add_host_kerberos.
- If the authentication does not work, then the WAPT Server will return a 403 error message.

In case of an error on one of the previous checks

- Delete the host account from the Active Directory.
- Delete the `/etc/nginx/http-krb5.keytab` file.
- Reboot the host you are testing with and re-run the keytab creation process again.

Note:

- It is important to restart the host to purge the kerberos tickets previously obtained by the host.
- To avoid restarting you can also execute the command “klist purge” as SYSTEM.

17.3 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 WAPT HTTPS Server certificate according to the `verify_cert` value in section `[global]` in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Table 1: 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.

17.3.1 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
wapt-get restart-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*.

Attention: If *wapt-get enable-check-certificate* returns an error, remove the `.crt` with same name on `C:\Program Files (x86)\wapt\sslserver`

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 WAPT 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 with the WAPT Server.

17.3.2 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 of the *Certificate Authority*.

You can manually add the certificate chain of 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, if you are with Redhat based OS with selinux, please fix rights with **restorecon** :

Debian / Ubuntu

```
chown root:www-data /opt/wapt/waptserver/ssl/*.pem
```

CentOS / RedHat

```
chown root:nginx /opt/wapt/waptserver/ssl/*.pem
restorecon -v -R /opt/wapt/waptserver/ssl/
```

- 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 with `verify_cert` in the `wapt-get.ini` file of the WAPT Agent.

To apply the new configuration to your entire fleet:

- Regenerate a WAPT Agent with the appropriate settings.
- Use a WAPT package to modify `wapt-get.ini` and push the certificate.

17.3.3 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.

17.4 Configuring user authentication against Active Directory

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 networks, the SuperAdmin account should not be used since it cannot provide the necessary traceability for administrative actions that are done on the network assets.

It is thus necessary to configure authentication against the Active Directory for the WAPT Console users; this will allow to use named accounts for 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 that the Administrator manages 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).

17.4.1 Enabling Active Directory authentication

- To enable authentication of the WAPT Server with Active Directory, configure the file `waptserver.ini` as follows.

Note: The WAPT Server configuration file on GNU/ Linux and macOS systems is found in `/opt/wapt/conf/waptserver.ini` or in `/opt/wapt/waptserver/waptserver.ini`.

The WAPT Server configuration file on Windows systems is found in `C:\wapt\conf\waptserver.ini`.

```
#waptserver.ini

wapt_admin_group=waptadmins
ldap_auth_server=srvads.mydomain.lan
ldap_auth_base_dn=DC=mydomain,DC=lan
ldap_auth_ssl_enabled=False
```

Table 2: Available authentication options

Options (Default Value)	Description	Example
`wapt_admin_group_dn` = (default [])	LDAP DN of Active Directory User Group allowed to connect to WAPT Console.	wapt_admin_group_dn = CN=waptadmins,OU=groups,DC=ad,DC
`wapt_admin_group` (default [])	Defines the sAMAccountName of the Active Directory User Group allowed to connect to WAPT Console, it is a list that can contain several groups. You can use this option over `wapt_admin_group_dn`, but **DO NOT** use both attributes at the same time.	wapt_admin_group = waptadmins, wapttech
`ldap_auth_server` (default `None`)	Defines the LDAP authentication server.	ldap_auth_server = srvads.mydomain.lan
`ldap_auth_base_dn` (default `None`)	Defines the LDAP authentication base DN.	ldap_auth_base_dn = dc=domain,dc=lan
`ldap_auth_ssl_enabled` (default `True`)	Use SSL authentication on LDAP connections.	`ldap_auth_ssl_enabled` = False
`verify_cert_ldap` (default `True`)	Checks the SSL certificate for LDAP connections, unless `ldap_auth_ssl_enabled` is set to **False** (or it will do nothing).	verify_cert_ldap = False

- Restart **`waptserver`** service.

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 do not have a certificate installed, you will 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 will need to modify the `ldap server require strong auth` parameter in `/etc/samba/smb.conf`. For more information you may refer to Tranquil IT's documentation on Samba-AD.

17.4.2 Enabling Single Sign On (SSO) for the WAPT Console and the self-service

Warning: This configuration is only available for WAPT Servers running on WAPT supported Linux distributions.

You can use Kerberos to authenticate yourself on the **waptconsole** and the **selfservice**. This way, users do not need to enter their password.

It is not necessary to register the WAPT Agent using kerberos in order to use the kerberos SSO (Single Sign-On) on the WAPT Console and in the Self-Service.

Preparing the WAPT Server for Kerberos Single Sign On

Attention: To enable Kerberos on the WAPT Server with `use_kerberos = True` option, launch the WAPT Server postconf script.

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

Please, refer to *the documentation on configuring kerberos for authentication* beforehand.

If you do not want to use Kerberos for Client registration, set the option `allow_unauthenticated_registration` to `True`.

Finally, restart the waptserver and wapttasks services.

```
systemctl restart waptserver wapttasks
```

There are 3 ways to configure the WAPT Server for Kerberos and LDAP authentication.

For each of these methods, you will need to modify the *waptserver.ini*.

1. **The first way** is the least secure.

This method does not verify the LDAP certificate nor does it use a secured port to contact the WAPT Server.

```
ldap_auth_ssl_enabled = False
verify_cert_ldap = False
```

Indeed, `ldap_auth_ssl_enabled = False` will not try to query the Active Directory using the LDAPS protocol.

The `verify_cert_ldap = False` option is set if you do not use *SSL/TLS support*.

Hint: If your Active Directory Server is a Samba-AD and you have this option in the `waptserver.ini`, then the Samba-AD server will refuse the connection.

```
ldap_auth_ssl_enabled = False
```

By default **Samba-AD** does not allow *SimpleBind* authentication without SSL / TLS.

If you do not have a valid certificate you will 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 Samba-AD.

---

2. **The second method** is more secure but still not perfect.

The method enables SSL authentication without verifying the certificate.

```
ldap_auth_ssl_enabled = True
verify_cert_ldap = False
```

The WAPT Server will try to use by default the LDAPS protocol without verifying the certificate when contacting the Active Directory.

3. **The third and recommanded way** is the most secure.

```
ldap_auth_ssl_enabled = True
verify_cert_ldap = True
```

- To make the method work, you will have to *enable the SSL/TLS support*.
- Then, you will need to add these options in the `waptserver.ini`:

  ```
  ldap_account_service_login = wapt-ldap@ad.tranquil.it
  ldap_account_service_password = PASSWORD
  ldap_auth_server = srvads.mydomain.lan
  ldap_auth_base_dn = dc=mydomain,dc=lan
  use_kerberos = True
  ```

  The `ldap_account_service_login` and `ldap_account_service_password` require a service user account on your Active Directory.

  It is not necessary that the service account had elevated right, just enough rights to read groups and group members. In other words, the WAPT Server **MUST** have read rights on the `memberof` attribute in the Active Directory.

- Finally, restart services on the WAPT Server:

  ```
  systemctl restart waptserver wapttasks
  ```

Configuring the WAPT Agent

On the client side, you will have to make sure that theses 2 options are set in the *wapt-get.ini* of the WAPT Agent:

```
service_auth_type = waptserver-ldap
use_kerberos = True
```

It is possible to make changes in `wapt-get.ini` manually or by deploying a WAPT package with the new configuration settings.

An example package is available from the Tranquil IT repository.

With this configuration, you can launch your WAPT Console or your selfservice without being prompted for a password.

To make this feature work, the Active Directory has to be available.

Note: The WAPT Console will continue to ask for a login / password. It is perfectly normal, this way you can use another user than the current user logged in the underlying desktop session.

Otherwise, you just have to put your login and click on *OK*.

17.4.3 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 636).

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 Certificate Authority *bundles* from the operating system for validating the SSL/ TLS connection to Active Directory.

If the Active Directory certificate is self-signed or the certificate has been signed by an internal CA, you will need to add the certificates to the operating system certificate store.

To do so, just add a *Certificate Authority* in the `/etc/pki/ca-trust/source/anchors/` and update the certificate store.

Debian / Ubuntu

```
cp cainterne.crt /usr/local/share/ca-certificates/cainterne.crt
update-ca-certificates
```

CentOS / RedHat

```
cp cainterne.crt /etc/pki/ca-trust/source/anchors/cainterne.crt
update-ca-trust
```

Windows

```
certutil -addstore -f "ROOT" cainterne.crt
```

- 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 `waptserver.ini`.

```
ldap_auth_ssl_enabled = True
```

- Restart **`waptserver`** service.

17.5 Configuring Client-Side Certificate Authentication

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 only to registered WAPT clients. It is done by relying on the WAPT Agent 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 strongly recommend enabling Kerberos or login / password registration in the WAPT Server post-configuration.

Warning: All actions are to be carried out on the WAPT Server

17.5.1 Enabling Client-Side Certificate Authentication on WAPT Server

Warning: For **Linux** check if the symbolic link in `sites-enabled` exists:

```
cd /etc/nginx/sites-enabled/
find . -maxdepth 1 -type l -ls
```

The expected result should be:

```
269091      0 lrwxrwxrwx   1 root     root           36 juil. 22 15:51 ./wapt.conf -> /etc/nginx/
↪sites-available/wapt.conf
```

Otherwise use the following command:

```
ln -s /etc/nginx/sites-available/wapt.conf ./wapt.conf
```

To enable the authentication, you need to add those parameters on WAPT *server configuration file* in the option section:

```
use_ssl_client_auth = True
```

Relaunch the post-configuring script.

Attention: Please note that as of 2024-09-20, WAPT does not support CRL, which means that when you delete a host in the WAPT Console, the host will still have access to the WAPT repository.

The WAPT Deployment utility cannot use **https** to retrieve the WAPT Agent, you will have to add this section in the file:

```
server {
  listen                    80;
  listen                    [::]:80;
  server_name               _;

  location ~ ^/(wapt/waptsetup-tis.exe|wapt/waptagent.exe|wapt/waptdeploy.exe)$ {
      add_header Cache-Control "store, no-cache, must-revalidate, post-check=0, pre-check=0";
      add_header Pragma "no-cache";
      root "/var/www";
  }

  return 301                https://$host$request_uri;
}
```

The WAPT Server having been successfully installed, now we will install the WAPT Console.

CHAPTER

EIGHTEEN

HOW TO INSTALL THE MANAGEMENT WAPT CONSOLE

18.1 On Windows

If you have already *generated the WAPT Agent* and *deployed the Agent* on your *Administrator*'s workstation, then *launch the WAPT Console*.

- Managing WAPT is done mainly via the WAPT Console installed on the *Administrator*'s workstation.
- It is recommended that the Administrator's computer 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.
- As of 2024-09-20, the WAPT Console is only supported on Windows. The Linux and macOS version are techpreview.

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.

18.1.1 The WAPT management Console

To download the `waptsetup.exe` file, point your web browser to your waptserver url https://srvwapt.mydomain.lan, then click on the *WAPTSetup* link on the right-hand side of the WAPT Server web page. The WAPT Server home page only provides basic server status information and the download link for the WAPT Console.

Installing the WAPT Agent on the Administrator's computer

Attention: If **the WAPT Agent** is not compiled and installed on your computer, you need to run de WAPT Agent installer to open and *configure the WAPT Console*.

- Start the executable installer as *Local Administrator* on the *Administrator*'s workstation.
- Choose the language for the WAPT installer.

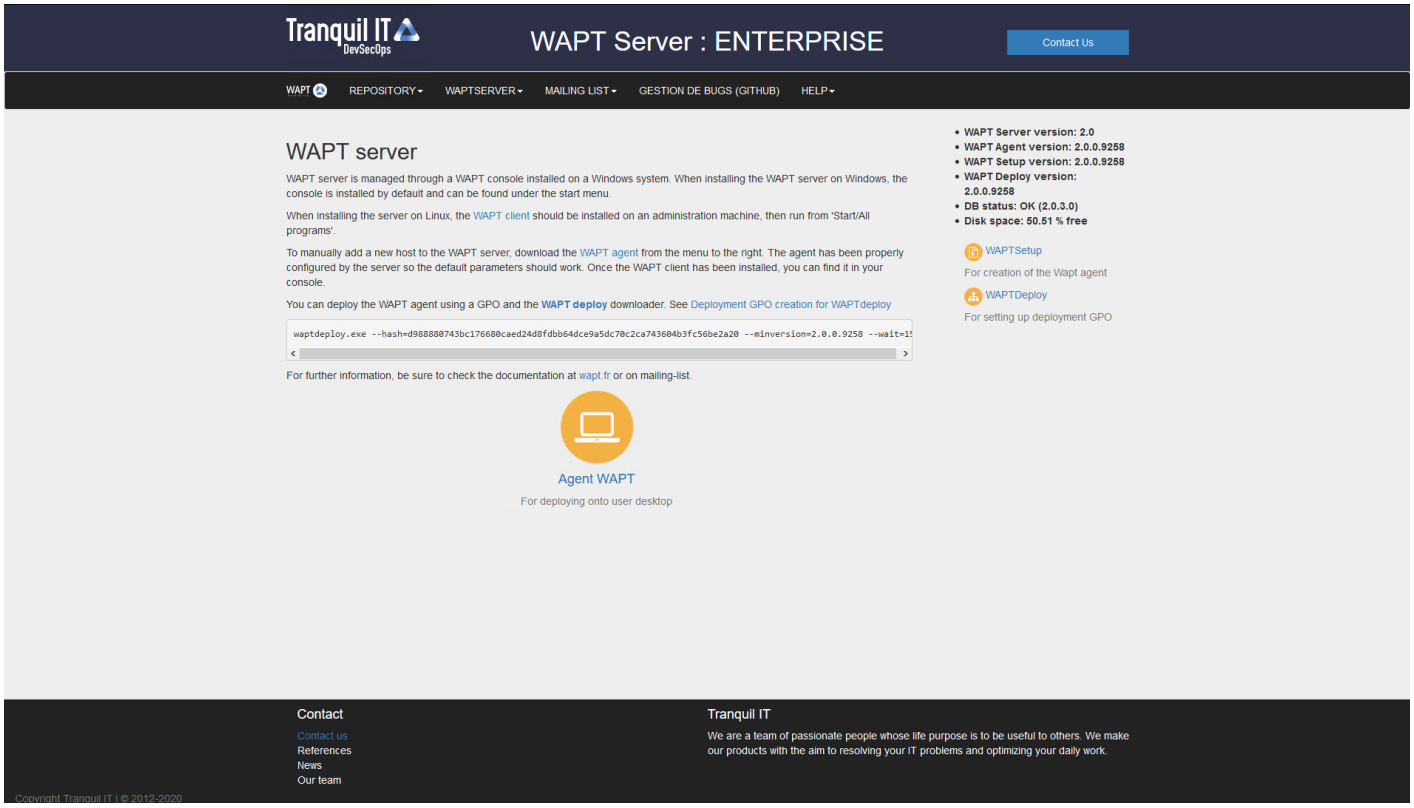
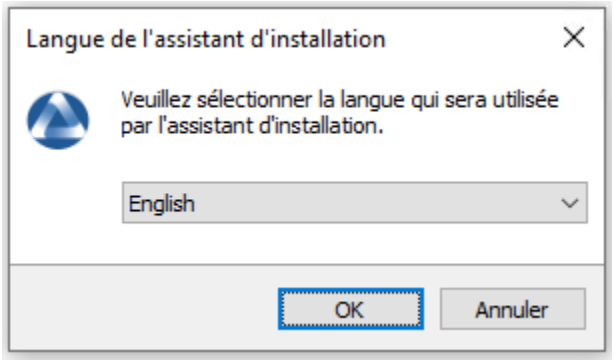
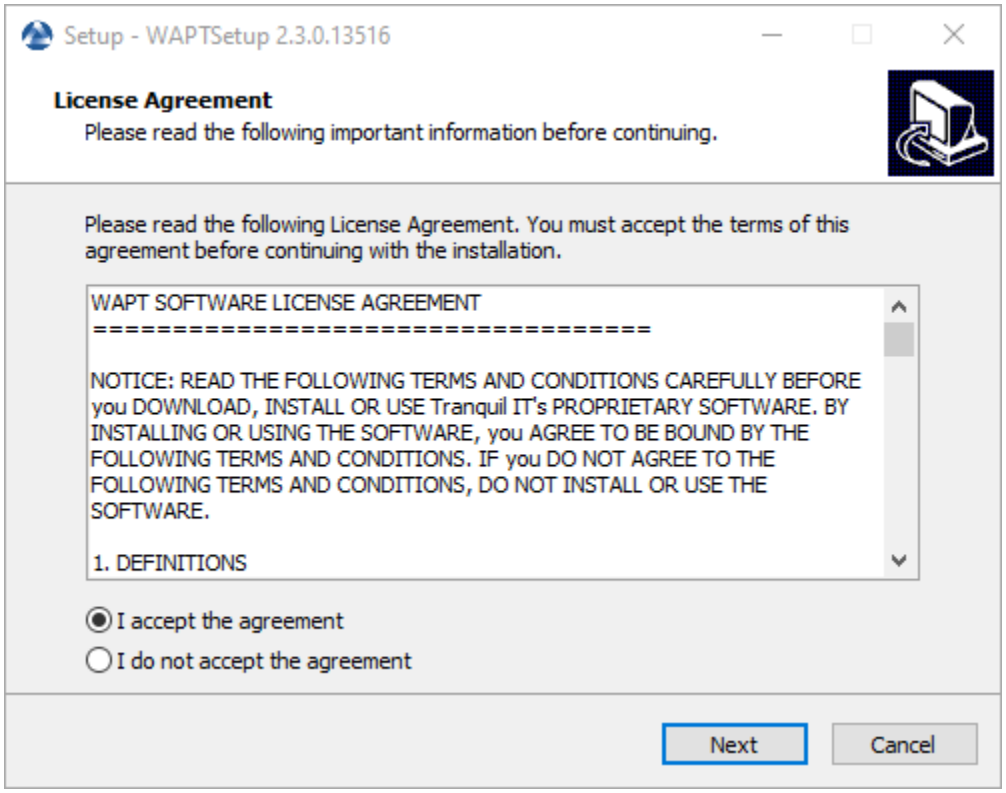


Fig. 1: The WAPT Server interface in a web browser



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).

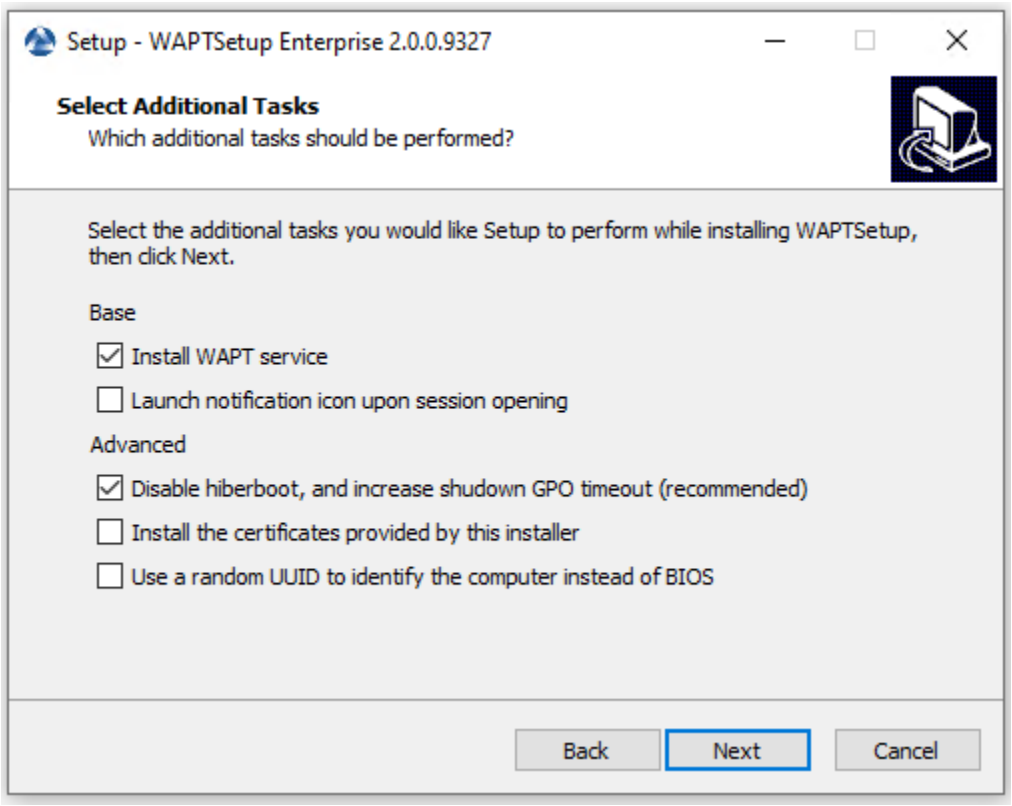


Fig. 2: Choosing the WAPT Agent installer options

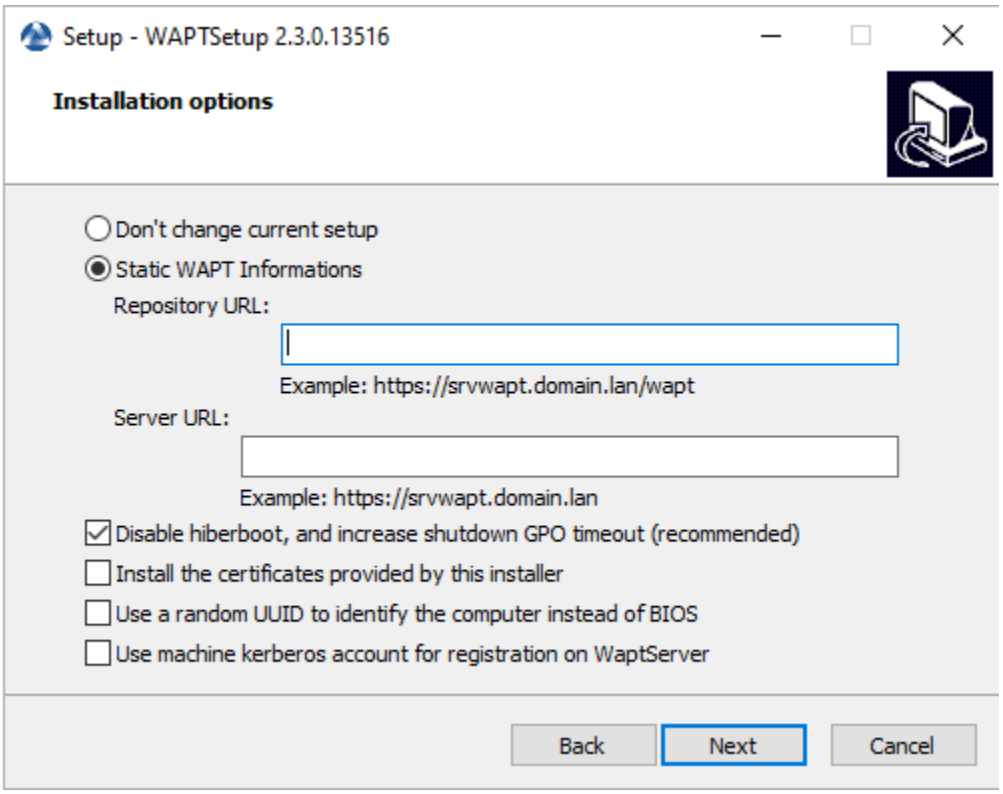
Table 1: Available options of the WAPT Agent installer

Settings	Description	Default value
Install WAPT service checkbox	Enables the WAPT service on this computer.	Checked
Launch notification icon upon session opening checkbox	Launches the WAPT Agent in systray on host startup.	Not checked
Disable hiberboot, and increase shutdown GPO timeout (recommended) checkbox	Disables Windows fast startup for stability, it increases the timeout for the WAPT Exit utility.	Checked
Install the certificates provided by this installer checkbox	Installs Tranquil IT certificate on this computer.	Not checked
Use a random UUID to identify the computer instead of BIOS checkbox	For more information, check the documentation on *BIOS UUID bugs*	Not checked

- Set up the WAPT Server URL.

Fisrt installation

- Check *Static WAPT Informations* and set:
  - WAPT repository URL: http://srvwapt.mydomain.lan/wapt.
  - WAPT Server URL: https://srvwapt.mydomain.lan.

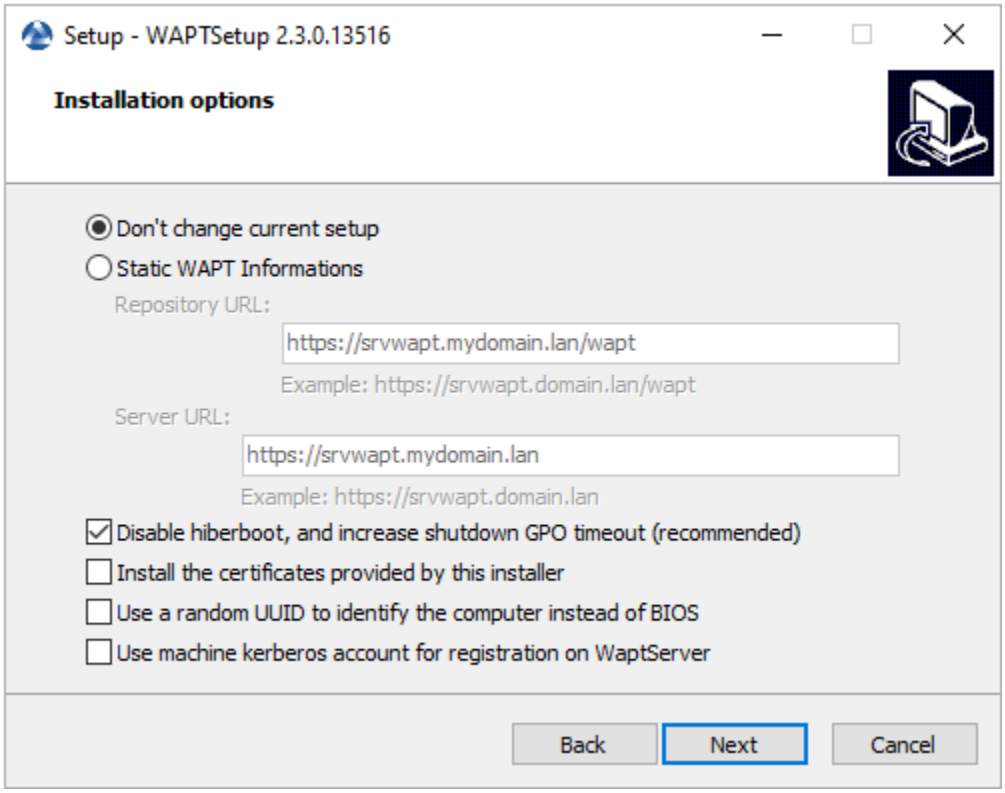


Choosing the WAPT repository and the WAPT Server

- Choose the WAPT repository and the WAPT Server; click *Next*.

Upgrade

- Check *Don't change current setup*, then click *Next*.



The WAPT repository and the WAPT Server are already set

- Get a summary of the WAPT Console installation.
- Click *Install* to launch the installation, wait for the installation to complete, then click on *Finish* (leave default options).

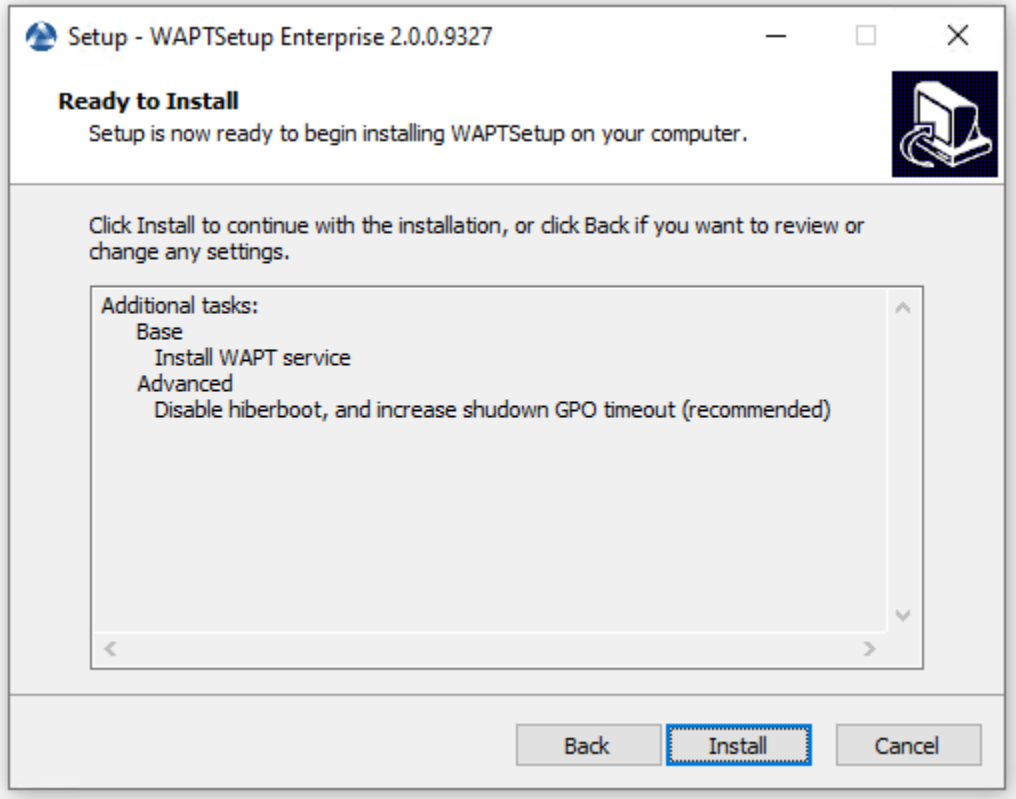
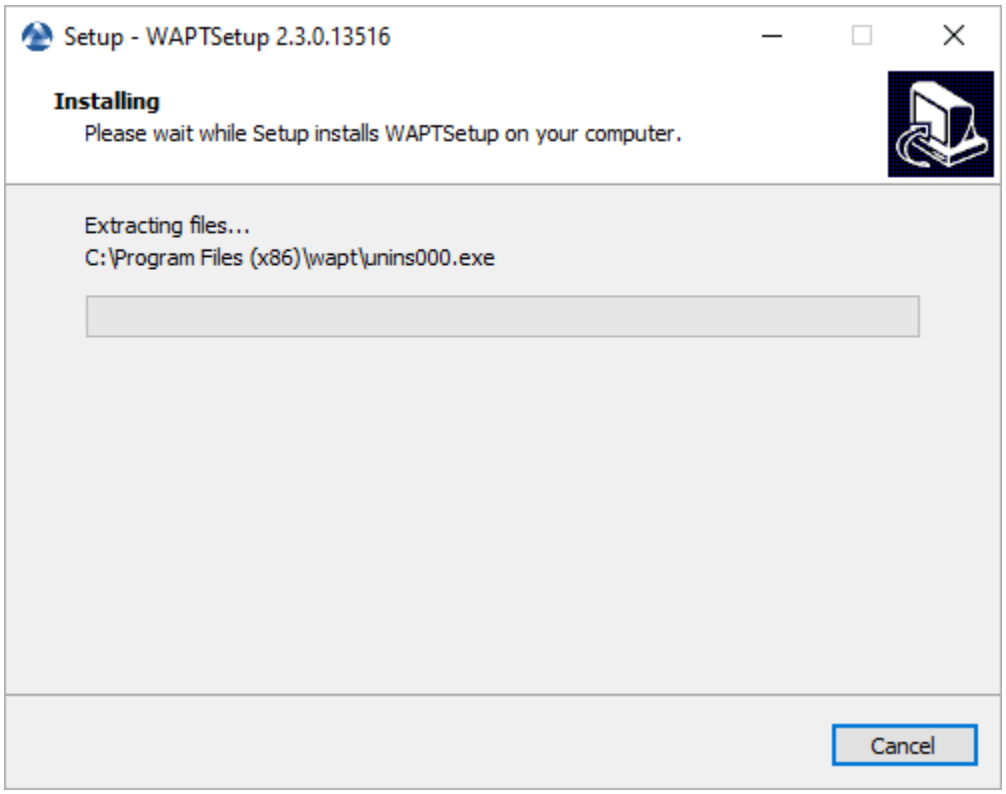
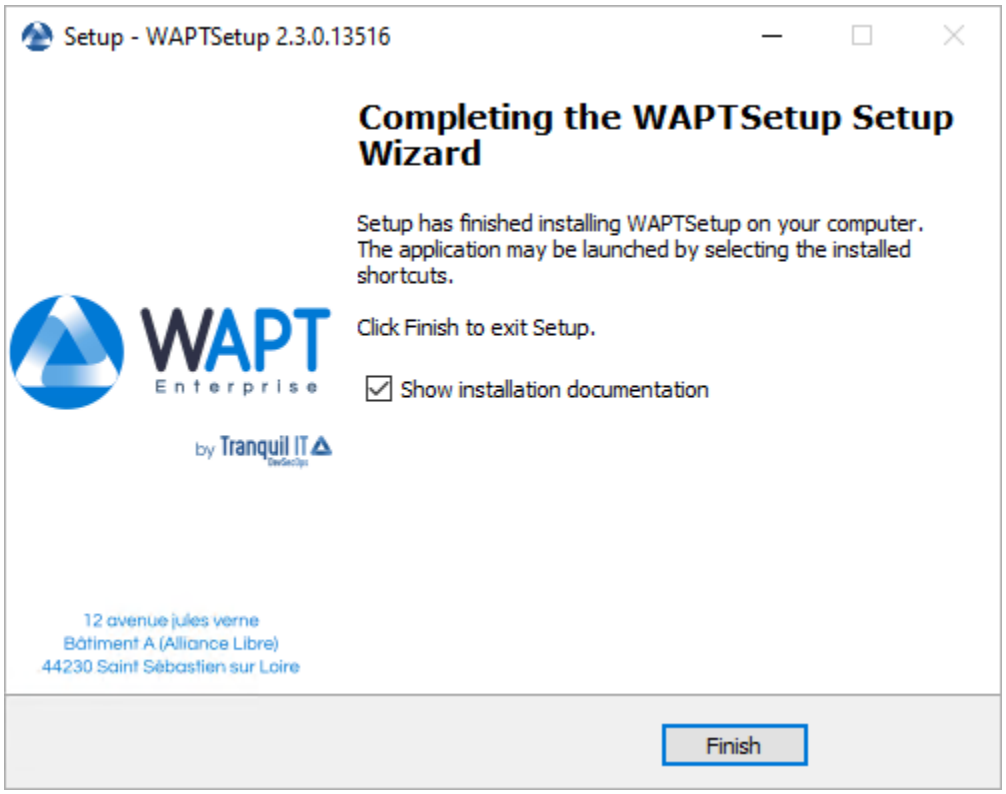


Fig. 3: Summary of the WAPT installation abstract





- Uncheck *Show installation documentation.*

Starting the WAPT Console

- Launch the WAPT Console:
  - By looking for the binary.

    `C:\Program Files (x86)\wapt\waptconsole.exe`
  - Or using the *Start* Menu.

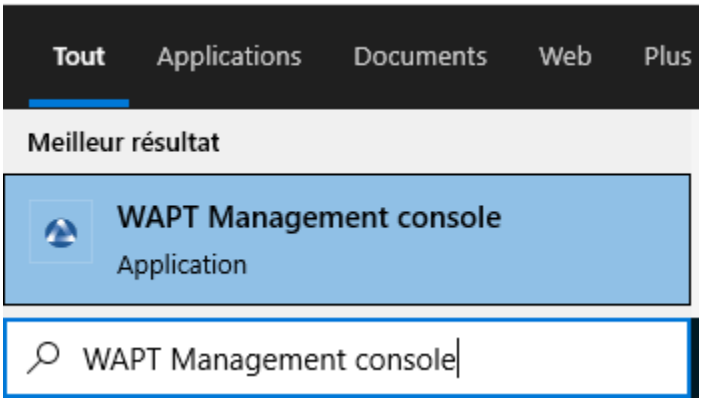


Fig. 4: Launching the WAPT Console from the Windows Start Menu

- Log into the WAPT Console with the *SuperAdmin* login and password.

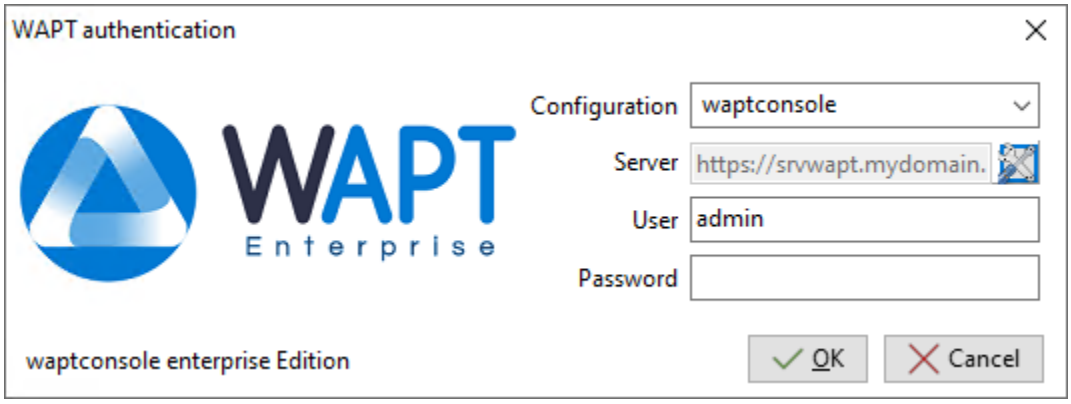


Fig. 5: The WAPT Console authentication window

If you have any issue logging into the WAPT Console, please refer to the FAQ: *Error message when opening the WAPT Console*.

It is recommended to launch the WAPT Console with a Local Administrator account to enable local debugging of WAPT packages.

For Enterprise version, it is possible to authenticate with *Active Directory*.

18.2 First start after the WAPT Server installation

Hint: On first start, you **MUST** start the WAPT Console with elevated privileges. *Right-click on the WAPT Console binary → Start as Local Administrator*.

18.2.1 Certificate affectation

Note: A message may appear indicating that no personal certificate has been defined.

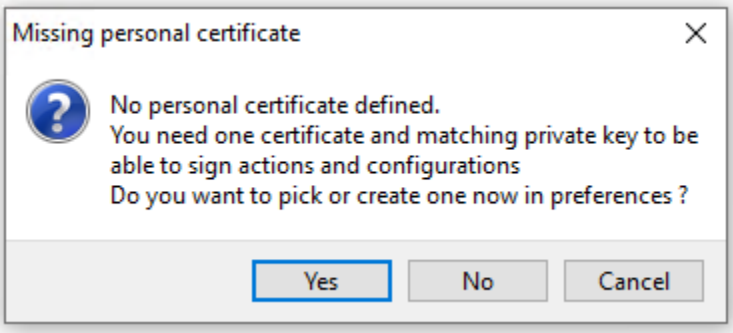


Fig. 6: WAPT personal certificate not found in the WAPT Console

- Select *Yes*
- Click on *New private key and certicate* and see *create your certificate*.

18.2.2 Packet prefix definition

Note: A message may appear indicating that no package prefix has been defined.

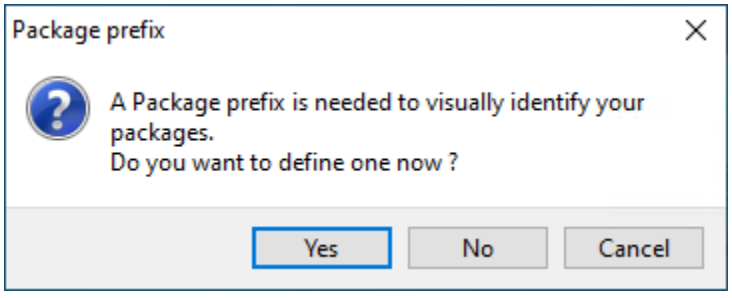


Fig. 8: Dialog box informing that no prefix has been set in the WAPT configuration

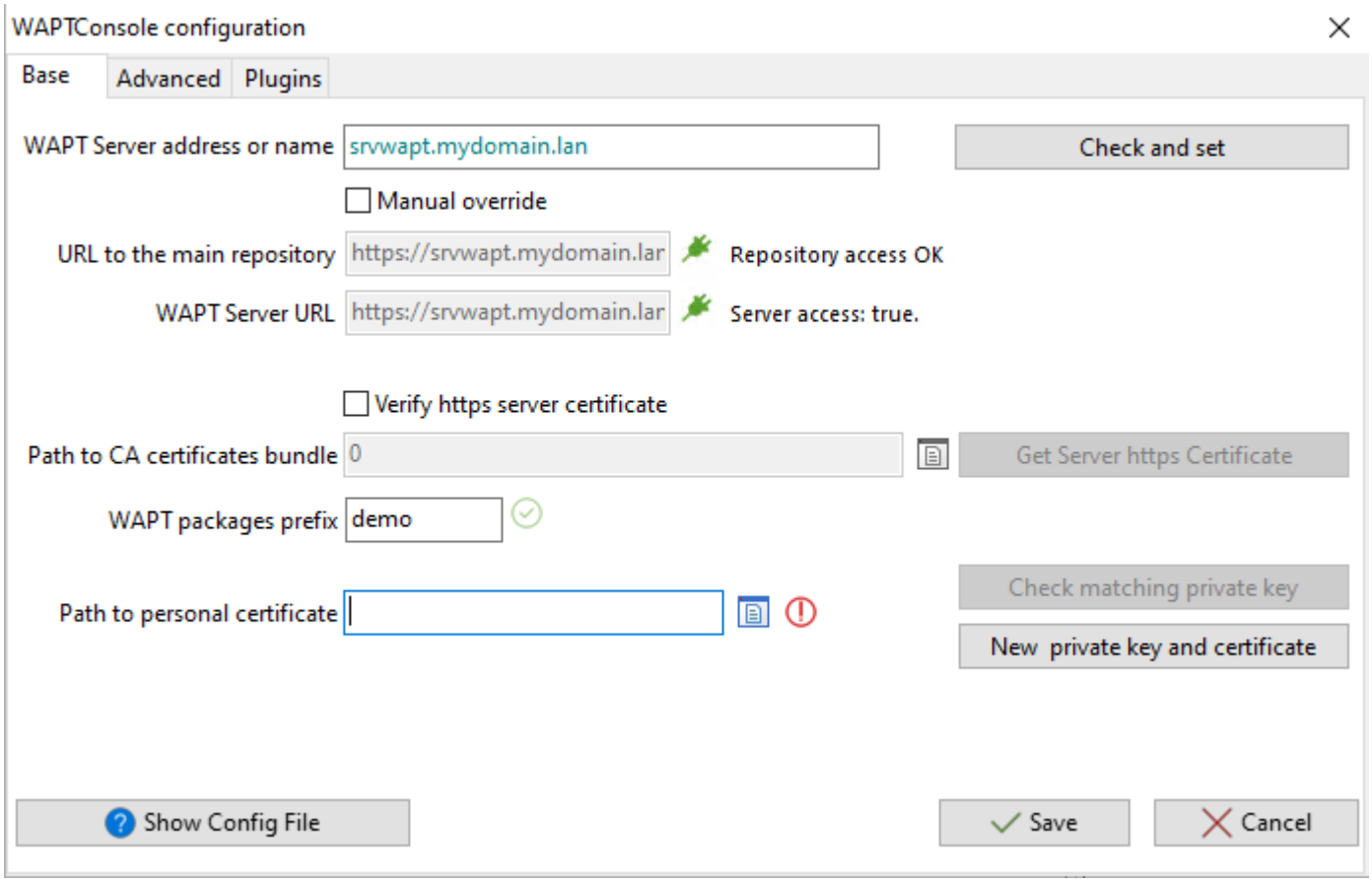


Fig. 7: Window for the basic configuration of the WAPT Console

- Select *Yes*
- Set your packages prefix on *WAPT packages prefix*

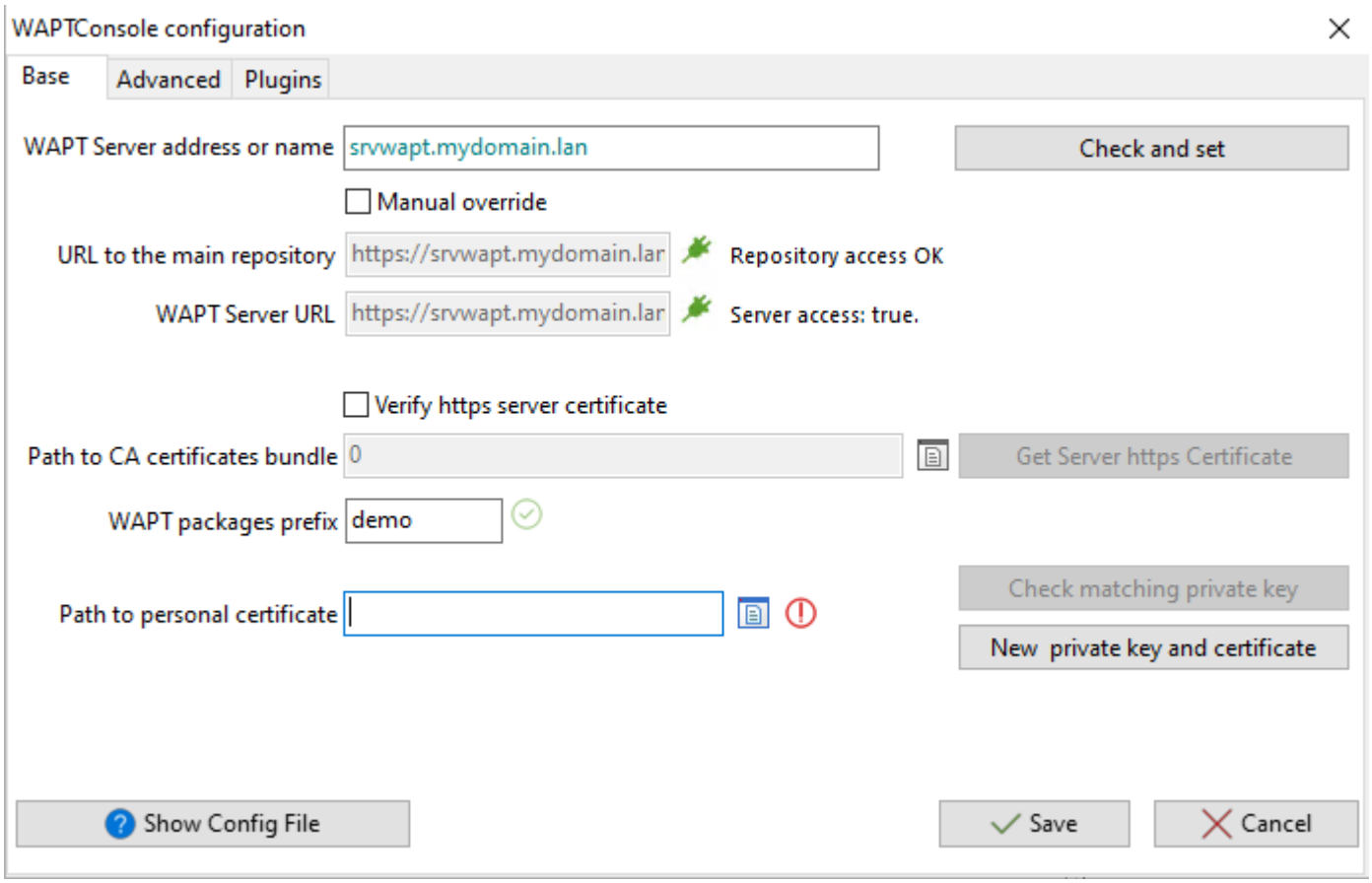


Fig. 9: Window for the basic configuration of the WAPT Console

Warning: The prefix is case sensitive, we recommand to use lower case.

18.2.3 waptagent.exe errors

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

If the *administrator's certicate* existing, it's possible to *generating new WAPT Agent* by clicking on *Yes*.

Also click on *No* and generate the *administrator's certicate*.

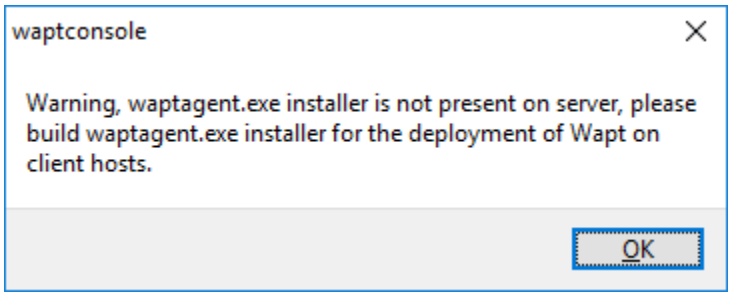


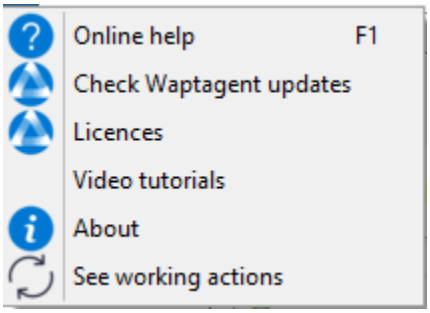
Fig. 10: Dialog box informing that the WAPT Agent is not present on the WAPT Server

18.3 Activating a WAPT licence

With WAPT, **Discovery** and **Enterprise** versions have different licences.

To activate the licence, use the `licence.lic` file provided by our sales department.

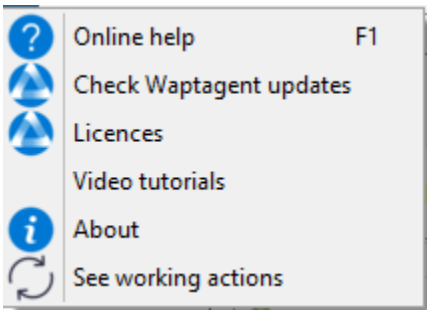
- In the WAPT Console, click on the *?* tab:



- Then choose *Licences*:
- Finally, select your `licence.lic` and click *Open*:

18.3.1 Removing a WAPT licence

- In the WAPT Console, click on the *?* tab:



- Then choose *Licences*:
- Finally, select the row and click *Remove License*:
- When confirmed, the selected licences are removed:

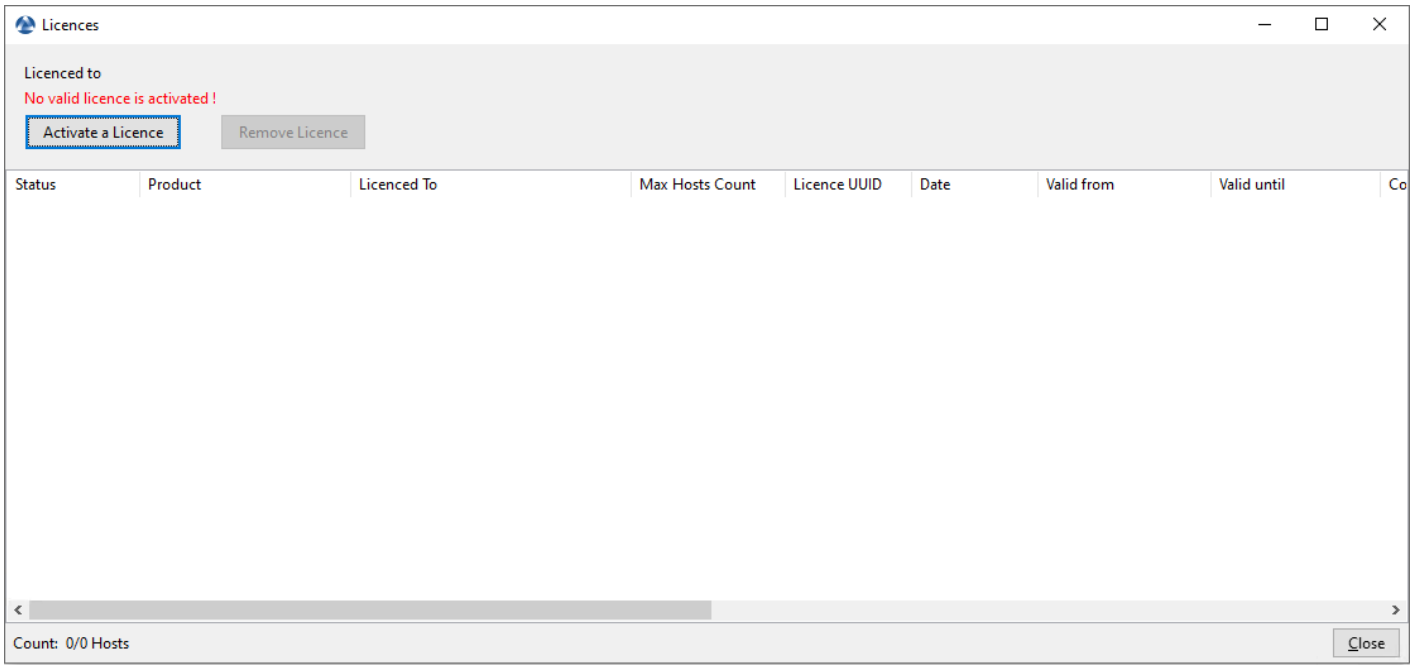


Fig. 11: Window listing no subscribed WAPT licences in the WAPT Console

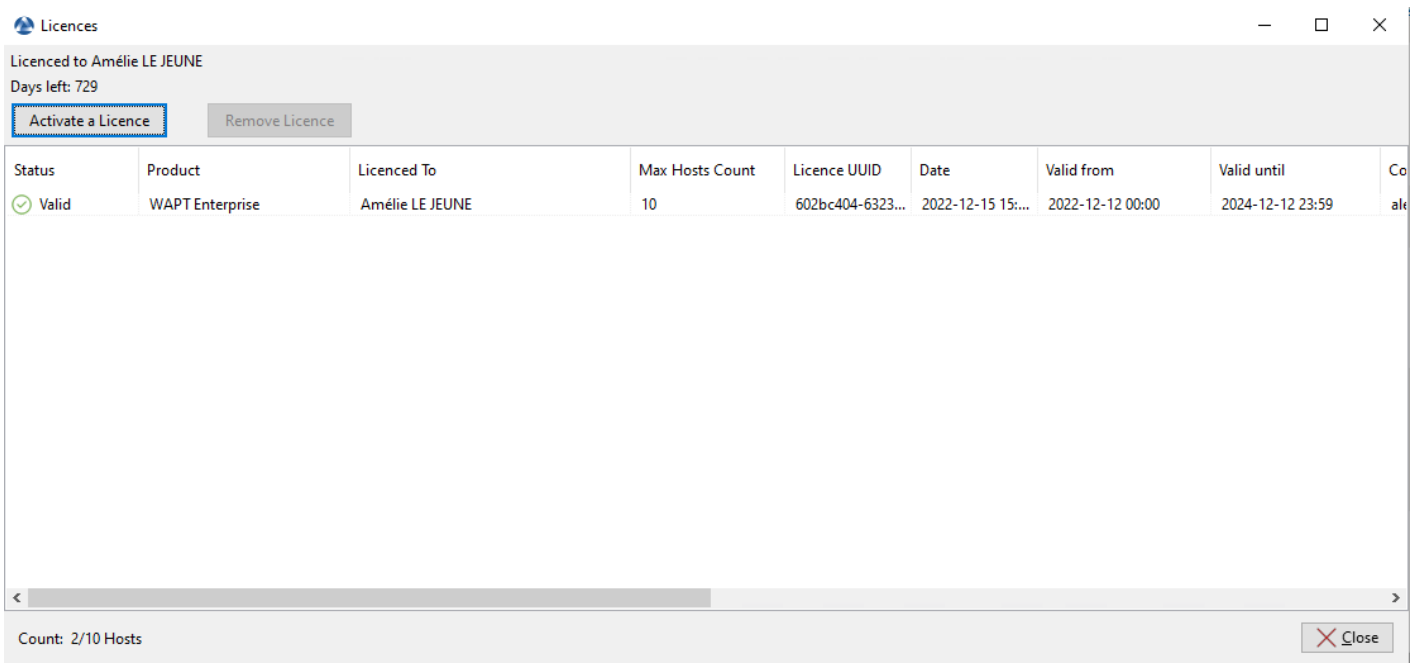


Fig. 12: Window showing an activated licence in the WAPT Console

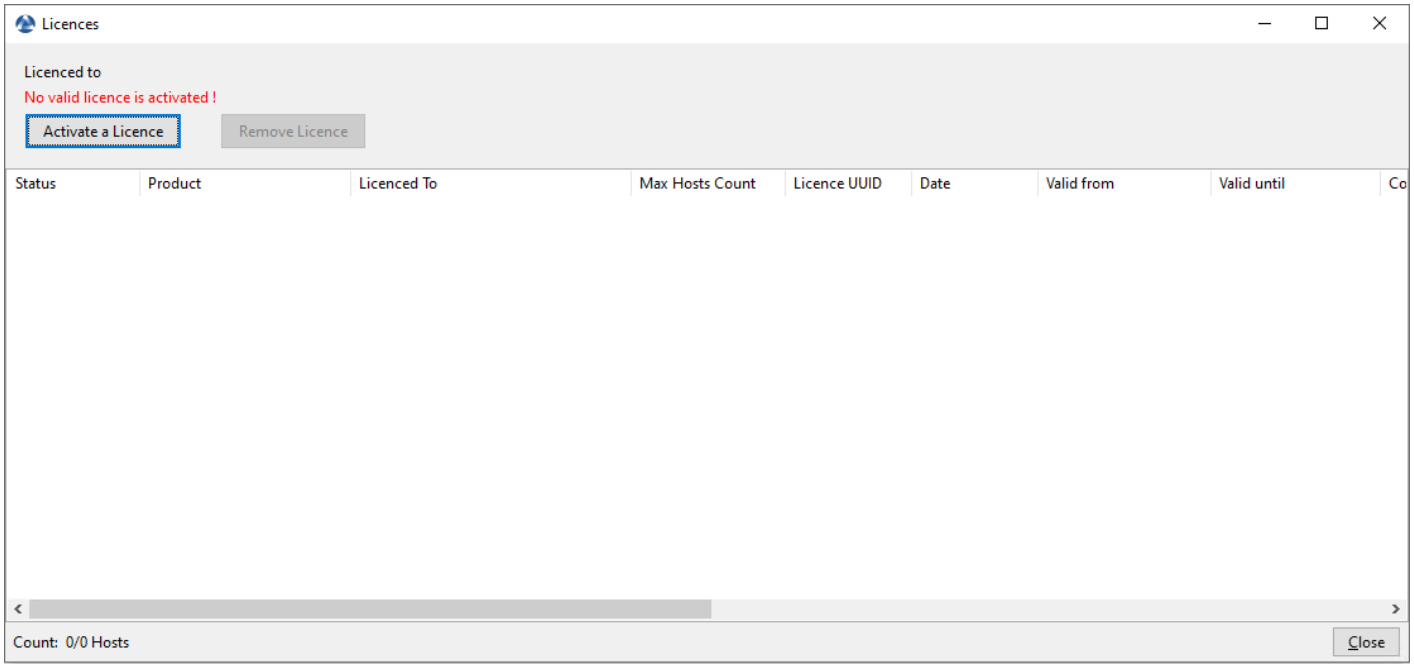


Fig. 13: Window listing no subscribed WAPT licences in the WAPT Console

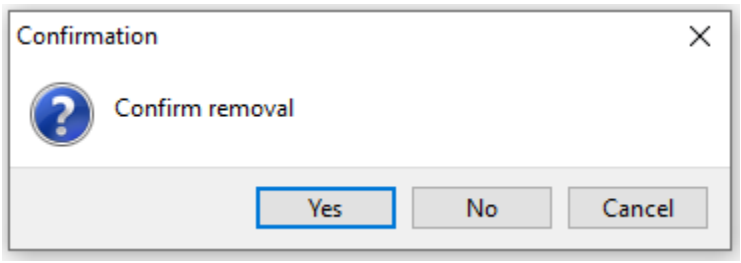


Fig. 14: Confirmation window to remove a licence from the WAPT Console

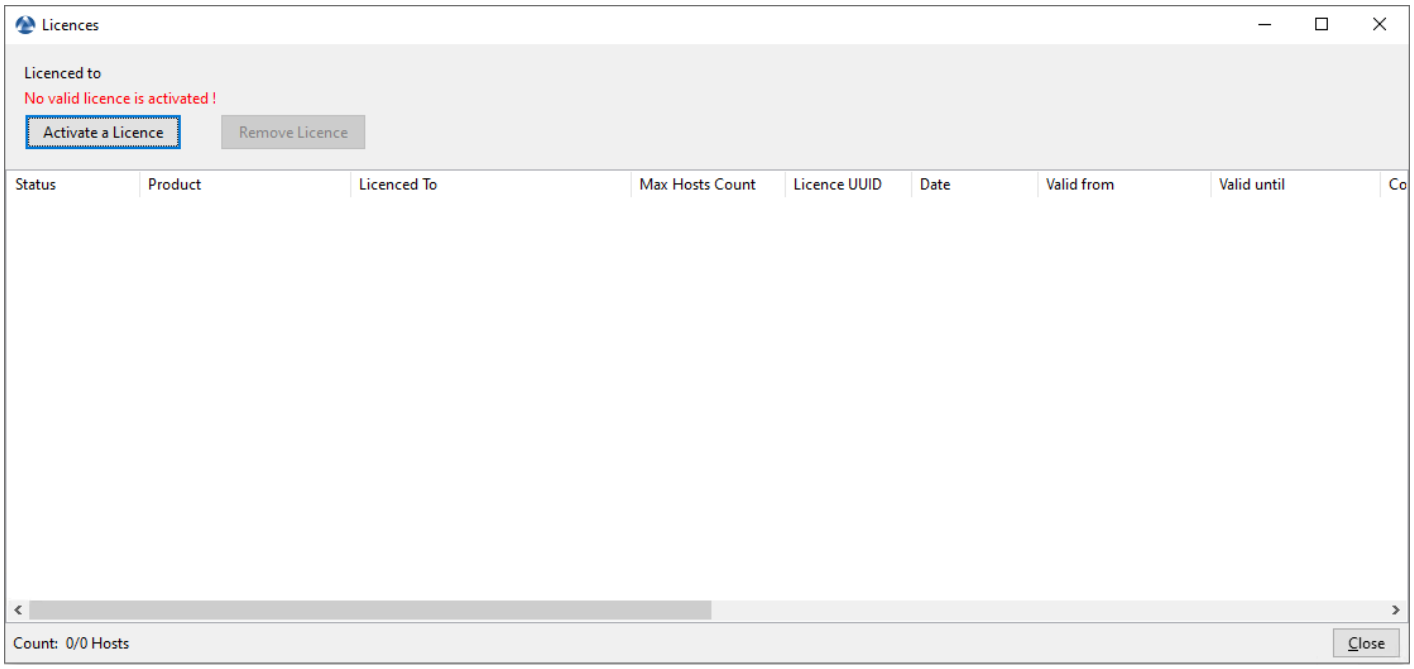


Fig. 15: Window listing no subscribed WAPT licences in the WAPT Console

18.3.2 License location

`licence.json` are stocked on the WAPT Server in the following location:

Debian / Ubuntu

```
/var/www/licences.json
```

RedHat and derivatives

```
/var/www/html/licences.json
```

Windows

```
C:\wapt\waptserver\repository\licences.json
```

18.3.3 License error

Expired licence

If a licence has expired, then its status displays *Expired*.

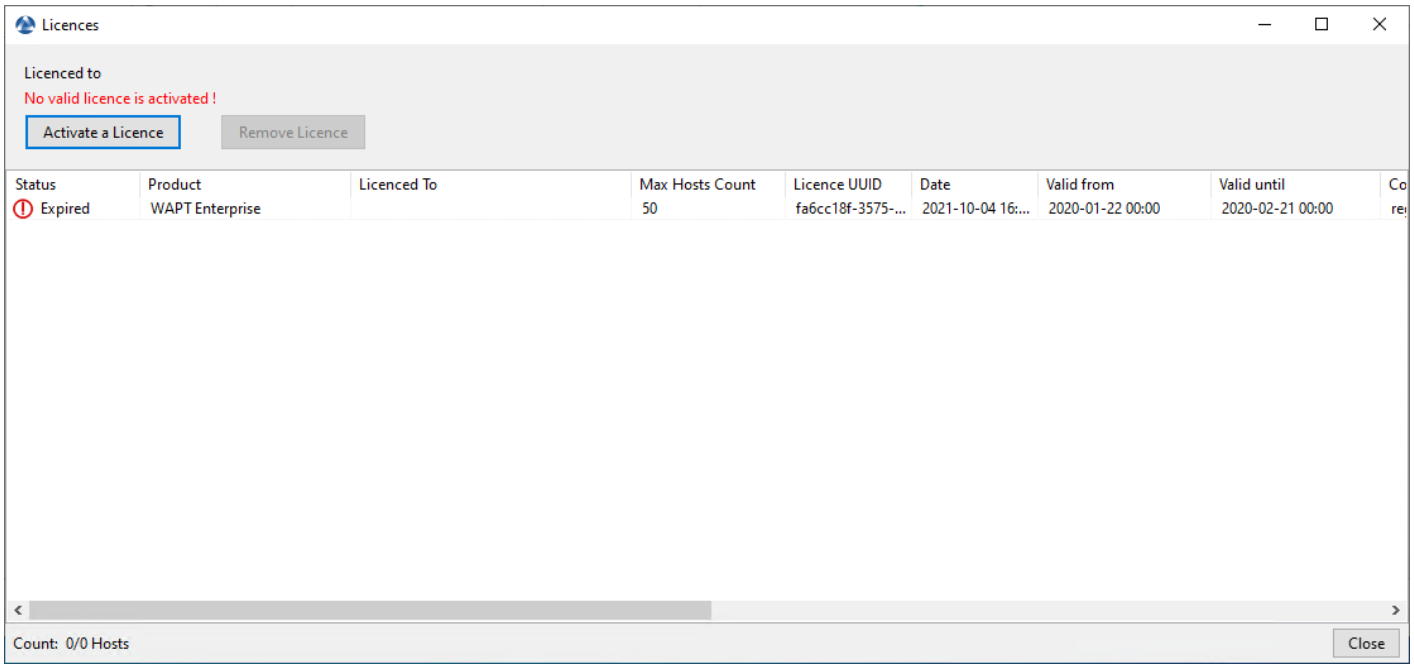


Fig. 16: Window showing an expired licence in the WAPT Console

Old licence location

When installaing the WAPT Console, if licence is located in an old location, this error appear will show:

Error activating a WAPT licence

This error is due to a problem with the post-configuration script and a special configuration of NGINX.

3 points are to be checked:

1. Check whether `/etc/nginx/sites-enabled/wapt.conf` is a symbolic link of `/etc/nginx/sites-available/wapt.conf`, using this command:

```
ls -l /etc/nginx/sites-enabled/wapt.conf
```

   - If the symbolic link exists, the output should be:

```
lrwxrwxrwx 1 root root 36 Jun 9 09:35 /etc/nginx/sites-enabled/wapt.conf --> /etc/nginx/
↪sites-available/wapt.conf
```

   - If the symbolic link does not exist, then remove `/etc/nginx/sites-enabled/wapt.conf` and create a new symbolic link:

```
rm /etc/nginx/sites-enabled/wapt.conf

ln -s /etc/nginx/sites-available/wapt.conf /etc/nginx/sites-enabled/wapt.conf
```

2. Check whether the file `licences.json` is present in *location* section of `/etc/nginx/sites-enabled/wapt.conf`:

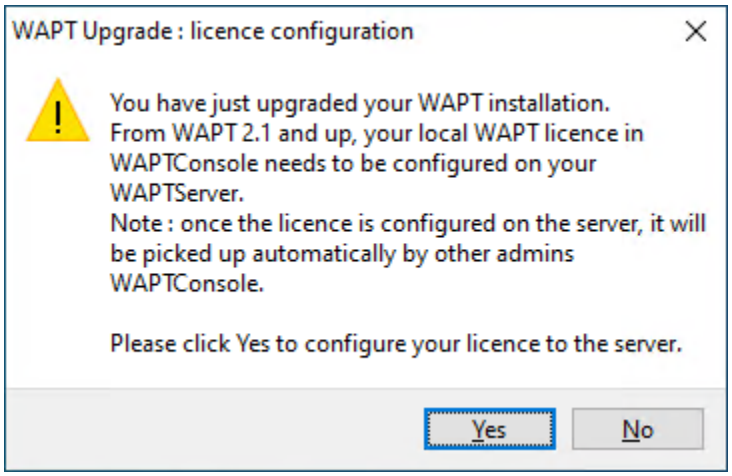


Fig. 17: WAPT licence error message when upgrading WAPT to 2.1

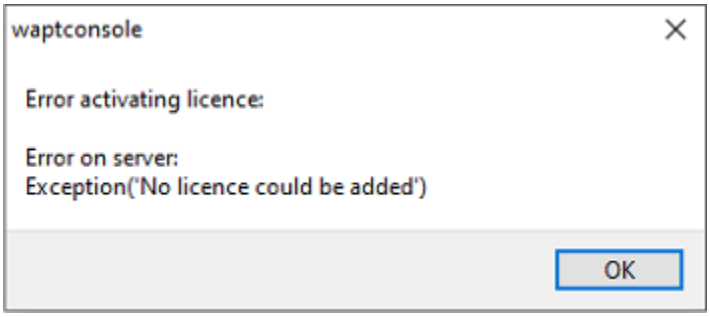


Fig. 18: Dialog box informing an error occured while activating a WAPT licence

```
location ~ ^/(wapt/waptsetup-tis.exe|wapt/waptagent.exe|wapt/waptdeploy.exe|sync.
↪json|rules.json|licences.json)$ {
        add_header Cache-Control "store, no-cache, must-revalidate, post-check=0, pre-
↪check=0";
        add_header Pragma "no-cache";
        root "/var/www";
    }
```

- If the `licences.json` file exist, then restart **`Nginx`**:

```
systemctl restart nginx
```

- Then, add the `licences.json` file in *location* section of `/etc/nginx/sites-enabled/wapt.conf` and restart NGINX.

```
systemctl restart nginx
```

3. If you get an error, empty `/var/www/licences.json`:

```
> /var/www/licences.json
```

- Then, retry *activating the WAPT licence*.

18.4 Generating the Administrator's certificate for signing WAPT packages

- In the example, the name of the private key is `wapt-private.pem`.
- In the example, the name of the public certificate signed with the private key is `wapt-private.crt`.

18.4.1 Private key *wapt-private.pem*

Danger: 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 **MUST NOT** be stored on the WAPT Server.

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

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

18.4.2 Public certificate : *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 of the Administrator, copied and deployed in `C:\Program Files (x86)\wapt\ssl` on the Windows desktops or in `/opt/wapt/ssl` on the Linux and MacOS devices managed by the Administrator via a WAPT package, a GPO or an Ansible role.

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

Attention:

- If the public certificate used on the WAPT Console is not derived from the private key used for generating the WAPT Agents, the WAPT Console will not see the WAPT Agents and you will not be able to perform any action on any WAPT Agent.
- The child certificates of private keys are functional for interactions.

18.4.3 Generating a certificate to use with WAPT

In the WAPT Console go to *Tools* → *Build certificate*.

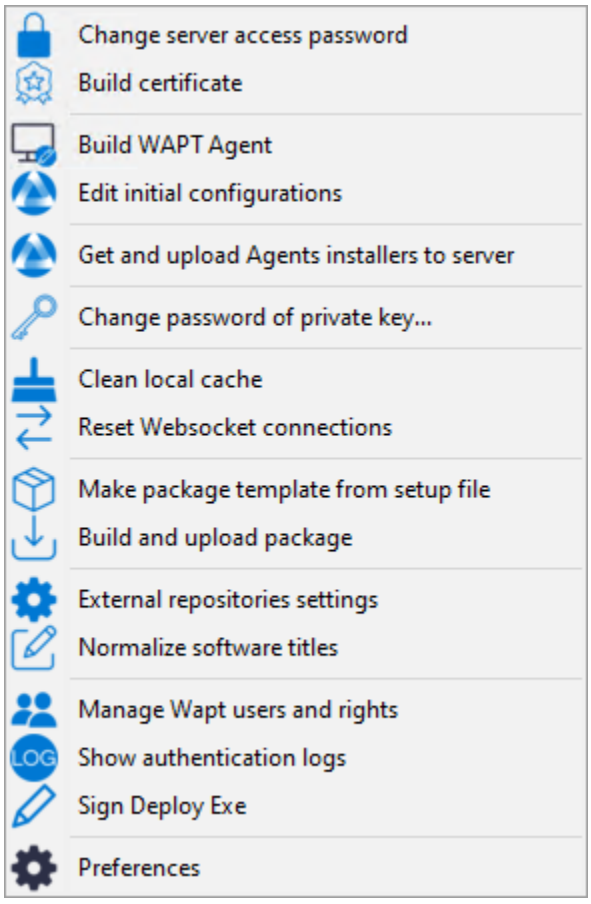


Fig. 19: Building a self-signed certificate

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

In order to create new signed certificates for delegated users, please refer to *creating a new certificate*.

Table 2: Certificate informations

Value	Description	Required	Enterprise
Target key directory	Defines the folder where the private key and the public certificate will be stored.	✔	
Key filename	Defines the name of the `.pem` private key.	✔	
Private key password	Defines the password for unlocking the key.	✔	
Confirm password	Confirms the password for unlocking the key.	✔	
Certificate name	Defines the name of the `.crt` certificate.	✔	
Tag as code signing	Defines whether the certificate/ key pair will be allowed to sign software packages.	✔	+
Tag as CA certificate	Defines whether the certificate can be used to sign other certificates (main or intermediate Certificate Authority).	✔	+
Common Name (CN)	Defines the Common Name to register in the certificate.	✔	
City	Defines the name of the certificate holder's city to register in the certificate.	✘	
Country (2 chars. E.g : FR)	Defines the name of the certificate holder's country (FR, EN, ES, DE ...) to register in the certificate.	✘	
Service	Defines the name of certificate holder's service or organizational department to register in the certificate.	✘	
Organization	Defines the name of the certificate holder's Organization to register in the certificate.	✘	
E-mail address	Defines the email address of the certificate holder to register in the certificate.	✘	
Authority Signing Key	Defines the key (`.pem`) of the CA.	✘	+
Authority Signing Certificate	Defines the certicate (`.crt`) of the CA.	✘	+
Export PKCS12	Forces the creation of the `*.p12` certicate in the *Targets keys directory*	✘ (recommended)	

Additional details are stored in the private key. This information will help with identifying the origin of the certificate and the origin of the WAPT package.

The password complexity **MUST** comply with your *Organization*'s security requirements (visit the ANSSI website for recommendations on passwords).

Danger:

- The `wapt-private.pem` file **MUST NOT** be stored on the WAPT Server.

- Click on *OK* to go on to the next step.

  If everything has gone well the following message will appear:

- Click on *OK*.
- Click on *Yes* to copy the newly generated certificate in the folder `C:\Program Files (x86)\wapt\ssl` on Windows or `/opt/wapt/ssl` on Linux or macOS. This certificate will be picked up during the compilation of the WAPT Agent and deployed

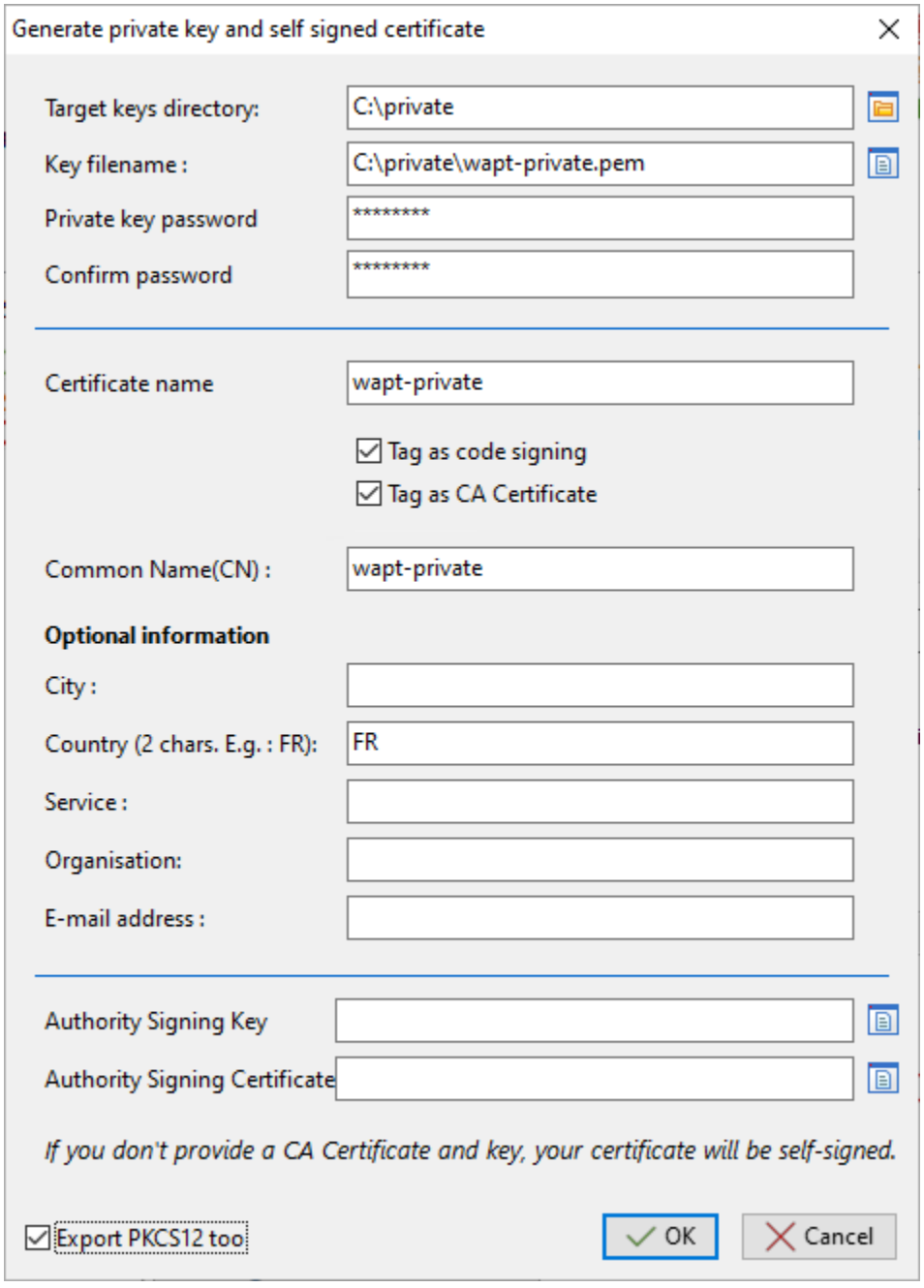


Fig. 20: Creating a self-signed certificate for the WAPT Enterprise version

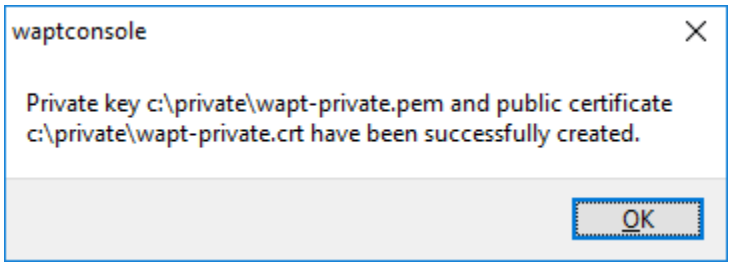


Fig. 21: Dialog box informing the certificate has been generated successfully

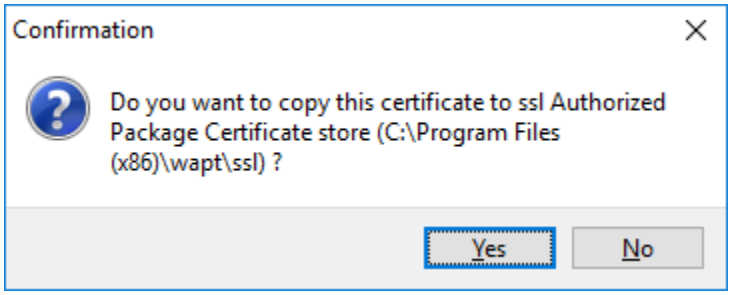


Fig. 22: Dialog box requesting confirmation of the copy of the certificate in the ssl folder in the WAPT Console

on the client computers.

You may go on to the next step and *Building the WAPT Agent installer*.

18.5 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:

Before building the WAPT Agent, please verify that your certificates are ready. 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 a common folder then select it when generating waptagent.

With the former waptagent build method, it was quite dangerous because one could **COPY the private key** of any *Administrator* in `C:\Program Files (x86)\wapt`. It means that, by error, a private key could be deployed on every computers, so it could be a serious **security breach**.

Before 2.3.0 version, this folder was used when building the WAPT Agent and the private keys would then be deployed on all the computers.

Now, the new method is far more secure:

It uses a waptsetup that is signed by Tranquil IT, we copy it and we push configuration into a *`json`* file. Alternatively, we can also *create a WAPT configuration package* that will be called when deploying the WAPT Agent. We call this method certificate stuffing.

On top of avoiding errors, like deploying a private certificate by error, the method has the advantage of no longer requiring to custom build a WAPT Agent build time. This method also avoids many Antivirus issues with false positives.

When the WAPT Agent will be silently installed, it will take the **default** configuration: it will build the WAPT Agent's `wapt-get.ini` configuration file and extract certificates into `wapt/ssl`.

To secure this installation (for example with GPOs), **`waptsetup.exe`** and its integrated *`json`* configuration have the name and hash of the configuration name on the WAPT Server. When the installer will apply the *`json`* configuration, it checks beforehand with this hash that the *`json`* data has not been altered.

- In the WAPT Console, go to *Tools* → *Build WAPT Agent*.

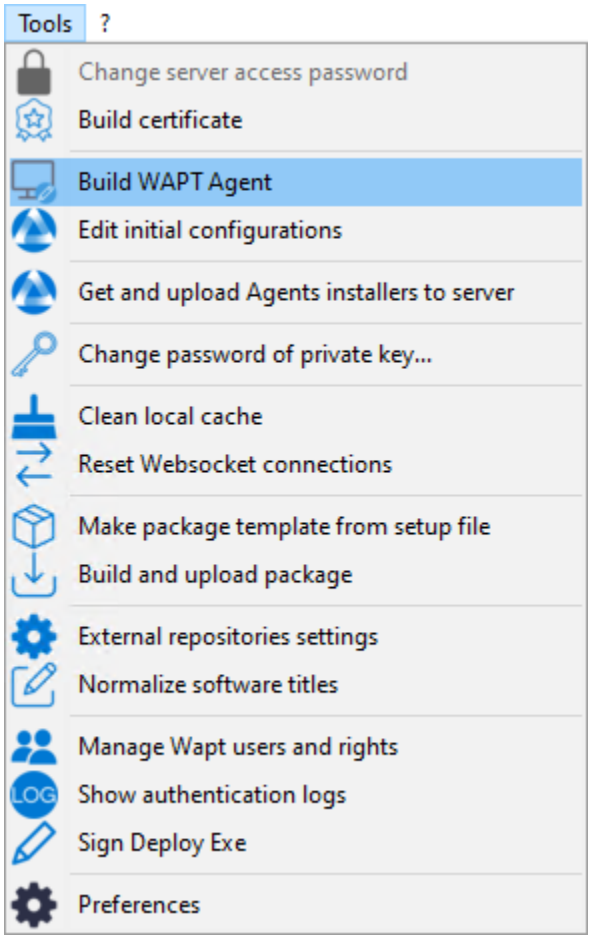


Fig. 23: Generating the WAPT Agent from the WAPT Console

Before building the WAPT Agent, you need to choose how it will identify itself with the WAPT Server.

18.5.1 Choosing the mode to uniquely identify the WAPT Agents

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

When a WAPT Agent registers the WAPT Server **MUST** know if it is a new host or if it is a host that has already been registered.

For this, the WAPT Server looks at the UUID (Universal Unique IDentifier) in the inventory.

WAPT offers 3 modes 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!

BIOS UUID (serial number)

This mode of operation makes it possible to identify the hosts in the WAPT 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!!!

host name

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

Note: This mode does not work if several hosts in your fleet share the same name.

We all know this should never happen.

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 device appear in your WAPT Console.

Note: In this mode, the UUIDs have the prefix RMD

18.5.2 Build

- In the WAPT Console, go to *Tools* → *Build WAPT Agent*
- Fill in the informations that are necessary for the installer.

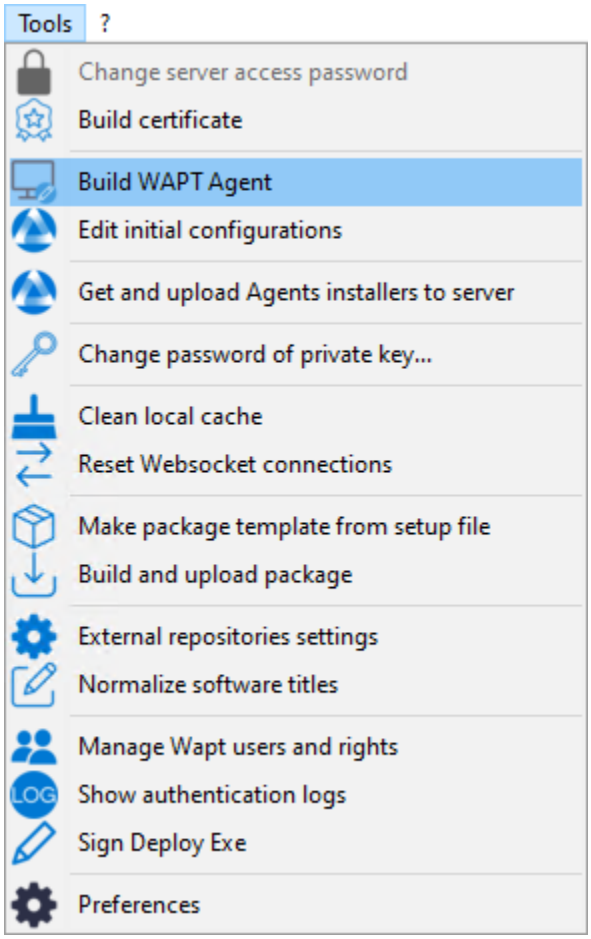


Fig. 24: Generating the WAPT Agent from the WAPT Console

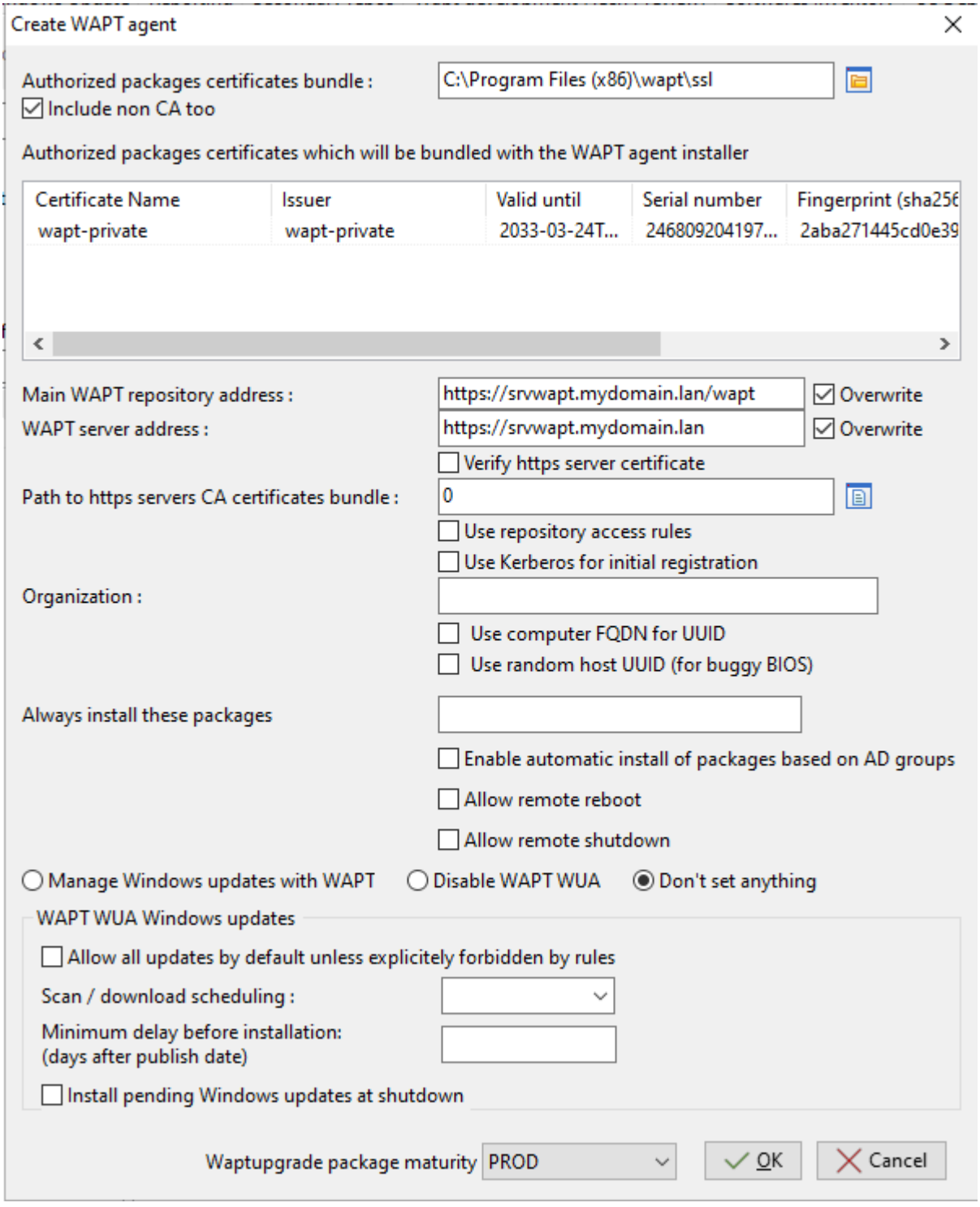


Fig. 25: Filling in the informations on your Organization

Table 3: WAPT Agent informations

Value	Description	Required	Enterprise
Authorized packages certificates bundle	Defines the folder of trusted certificates.	✓	
Include non CA too	Defines whether to include local WAPT certificate.	✗	
Main WAPT repository address	Defines the URL of the main WAPT repository.	✓	
WAPT Server address	Defines the URL of the WAPT Server.	✓	
Verify https server certificate	Defines whether the *HTTPS certificate client authentication* is activated on the WAPT Server.	✗	
Use repository access rules	Defines whether repository access rules are to be used for *replicating remote repositories*.	✗	+
Path to the WAPT https Servers CA certificates bundle	Defines the path to the certificates used for HTTPS verification.	✗	
Use Kerberos for initial registration	Defines whether *Kerberos authentication* of the WAPT Agents is to be used with the WAPT Server.	✗	
Organization	Defines the name of the Organization to identify the origin of WAPT packages.	✗	
Use computer FQDN for UUID	Defines whether FQDNs (Fully Qualified Domain Names) are to be used for *identifying WAPT Agents*.	✗	
Use random host UUID (for buggy BIOS)	Defines whether random UUIDs (Universally Unique IDentifiers) are to be used for *identifying WAPT Agents*.	✗	
Always install these packages	Defines whether to automatically install *group packages* upon WAPT Agent installation.	✗	+
Enable automatic install of packages based on AD Groups	Enables the installation of *profile packages*. **This feature can degrade the performance of WAPT**.	✗	+
Allow remote reboot	Defines whether to allow remote reboots from the WAPT Console.	✗	+
Allow remote shutdown	Defines whether to allow remote shutdowns from the WAPT Console.	✗	+
Manage Windows updates with WAPT | Disable WAPT WUA | Don't set anything	Enables or disables *WAPT WUA*.	✓	+
Allow all updates by default unless explicitely forbidden by rules	Defines whether to allow all Windows Updates if not forbidden by WUA rule packages.	✗	+
Scan / download scheduling	Sets the Windows Update scan periodicity.	✗	+
Minimum delay before installation (days after publish date)	Sets a deferred installation delay before publication.	✗	+
Install pending Windows updates at shutdown	Forces updates to install when the host shuts down.	✗	+
Waptupgrade package maturity	Allows to choose the maturity of the waptupgrade package.	✗	+

For more information to Windows update section, refer to *this article on configuring WAPTWUA on the WAPT Agent*

Danger:

- The checkbox *Use kerberos for the initial registration* may be checked **ONLY IF** you have followed the documentation on *Configuring the kerberos authentication*.
- The checkbox *Verify the WAPT Server HTTPS certificate* may 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.

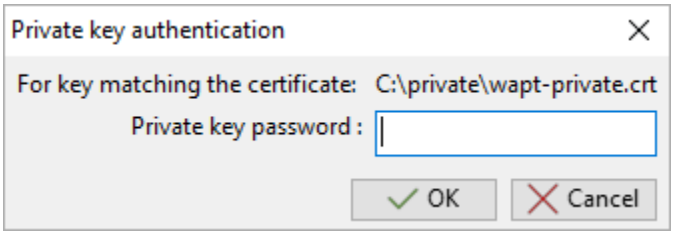


Fig. 26: Providing the password for unlocking the private key

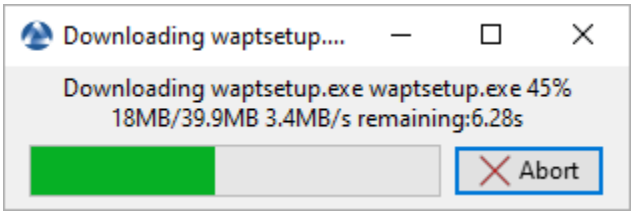


Fig. 27: Progression of WAPT Agent installer building

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/.

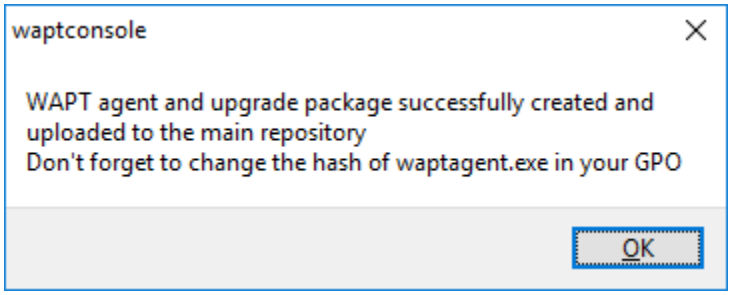


Fig. 28: Confirmation of the WAPT Agent loading onto WAPT repository

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.

Attention: After building the Agent on your management PC, quit the WAPT Console and *install* the **new WAPT Agent** that has been generated on your WAPT management computer.

18.6 Initial Configuration

It is possible to configure the WAPT Agent for standard and advanced options via a GUI. Very similar to *creating a configuration package*, we **strongly** recommand you to see the section beforehand. The initial configuration aims to configure important parameters in the WAPT Agent, whether it be Windows, Linux or macOS. The method is very useful for installing a *WAPT Agent on Linux or macOS*.

- In the WAPT Console, go to *Tools* → *Edit initial configurations*

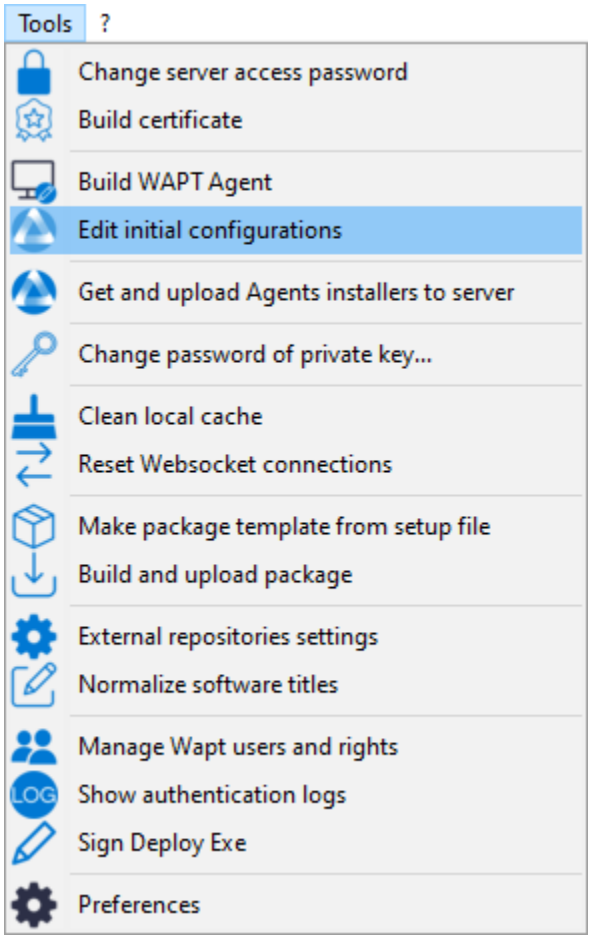


Fig. 29: Creating the initial configuration

- Fill in the informations that are necessary for the configuration

Table 4: Header

Value	Description
Advanced Editing	Displays *the WAPT Agent configuration options* as in `wapt-get.ini`.
Add certificate	Adds *certificate* with the configuration.
Load Json	Loads a previously created configuration.
Refresh Server Configuration	Refreshes the list of available configurations.
+	Creates a new configuration.
-	Deletes a configuration.

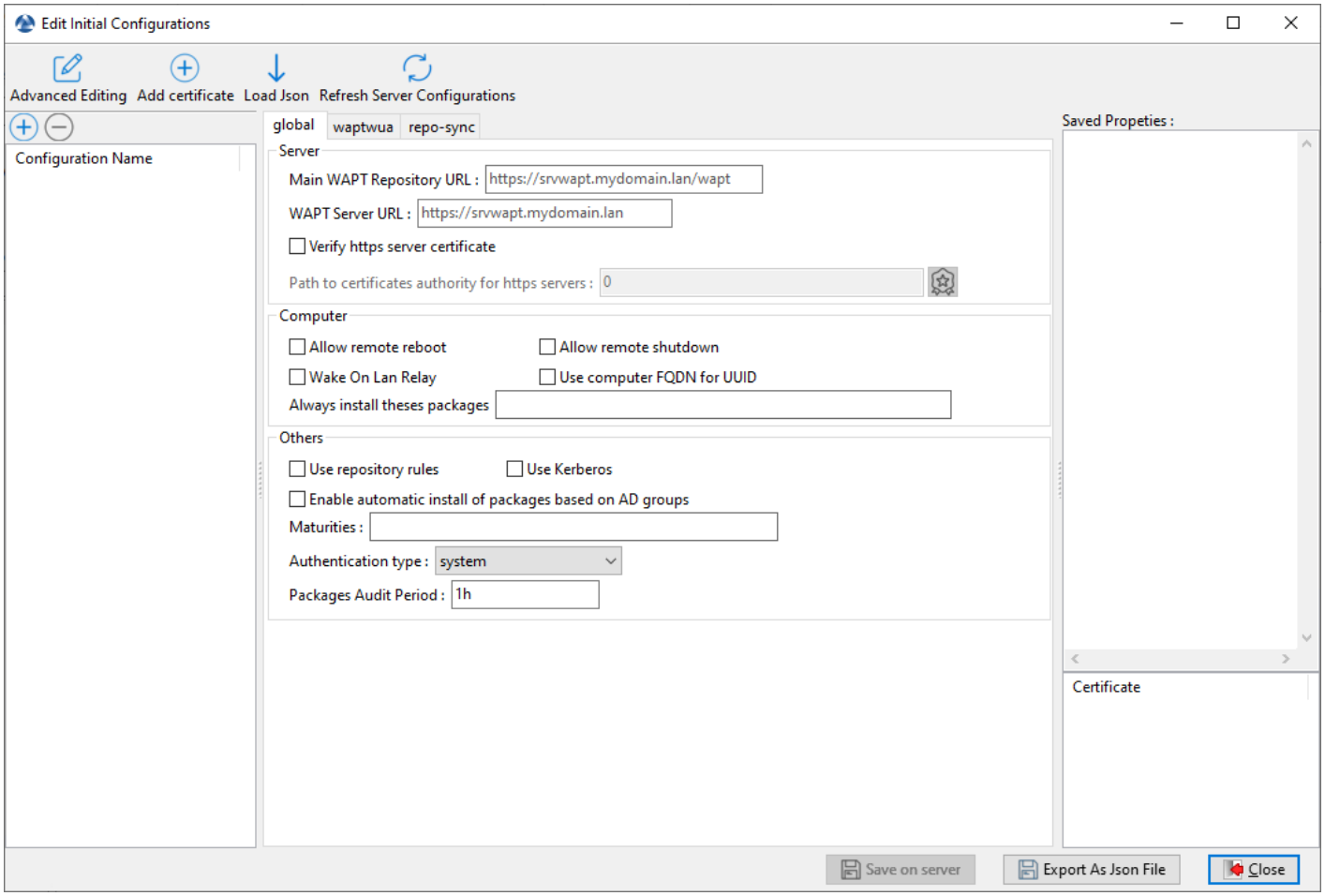


Fig. 30: Editing the initial configuration

global

Value	Description	Required	Enterprise
Main WAPT Repository URL	Defines the URL of the main WAPT repository.	✓	
WAPT Server URL	Defines the URL of the WAPT Server.	✓	
Verify https server certificate	Defines whether the *HTTPS certificate client authentication* is activated on the WAPT Server.	✗	
Path to certificate authority for https servers	Defines the path to the certificates used for HTTPS verification.	✗	
Allow remote reboot	Defines whether to allow remote reboots from the WAPT Console.	✗	+
Allow remote shutdown	Defines whether to allow remote shutdowns from the WAPT Console.	✗	+
Wake On Lan Relay	Activates the WoL (Wake-on-Lan) functionality on secondary repositories.	✗	+
Use computer FQDN for UUID	Defines whether FQDNs are to be used for *identifying WAPT Agents*.	✗	
Always install these packages	Defines whether to automatically install *group packages* upon WAPT Agent installation.	✗	+
Use repository rules	Defines whether *repositories are replicated*.	✗	+
Use Kerberos	Defines whether *Kerberos authentication* of the WAPT Agents is to be used with the WAPT Server.	✗	
Enable automatic install of packages based on AD Groups	Enables the installation of *profile packages*. **This feature can degrade the performance of WAPT**.	✗	+
Maturities	List of package maturities than can be viewed and installed by WAPT Agent. Default value is `PROD`. Only `DEV`, `PREPROD` and `PROD` values are used by Tranquil IT, however any value can be used to suit your internal processes.	✗	
Authentification type	Sets how the self service authentication works. Possible values are: `system`, `waptserver-ldap` or `waptagent-ldap`.	✓	
Packages Audit Period	Defines the frequency at which audits are triggered.	✓	

waptwua

Value	Description	Required
Manage Windows updates with WAPT	Enables or disables *WAPT WUA*.	✓
Allow all updates by default unless explicitely forbidden by rules	Defines whether to allow all Windows Updates if not forbidden by WUA rule packages.	✗
Allowed Severities	Defines a severity list that will be automatically accepted during a WAPT windows update scan. ex: *Important*, *Critical*, *Moderate*.	✗
Download updates from Microsoft Servers	Defines whether updates are downloaded directly from Microsoft servers.	✗
Scan / download scheduling	Defines the Windows Update scan recurrence (Will not do anything if *waptwua* package rule or `wsusscn2.cab` file have not changed).	✗
Install pending Windows updates at shutdown	Forces updates to install when the host shuts down.	✗
Installation scheduling	Defines the Windows Update install recurrence (Will do nothing if no update is pending).	✗
Minimum delay before installation (days after publish date)	Sets a deferred installation delay before publication.	✗

repo-sync

Attention: These options should only be used on a secondary repository.

Value	Description	Required
Use remote repo	Enables the WAPT Server to serve as a repository.	✓
Remote repository directories	Defines folders to synchronize	✓
Synchronize only when asked	Enable or disable automatic synchronization	✗
Synchronize task period	Sets synchronization periodicity	✓
Local repository time for synchronization start	Sets synchronization start time (HH:MM / 24h format)	✗
Local repository time for synchronization end	Sets synchronization start stop (HH:MM / 24h format)	✗

Table 5: Column

Value	Description
Saved Properties	List of *options* with the configuration.
Certificate	List of *certificate* with the configuration.

Table 6: Footer

Value	Description
Save on server	Save the configuration on the server
Export As Json File	Export the configuration in JSON
Close	Close the window

- After configuration it is possible to copy commands by right clicking on the configuration

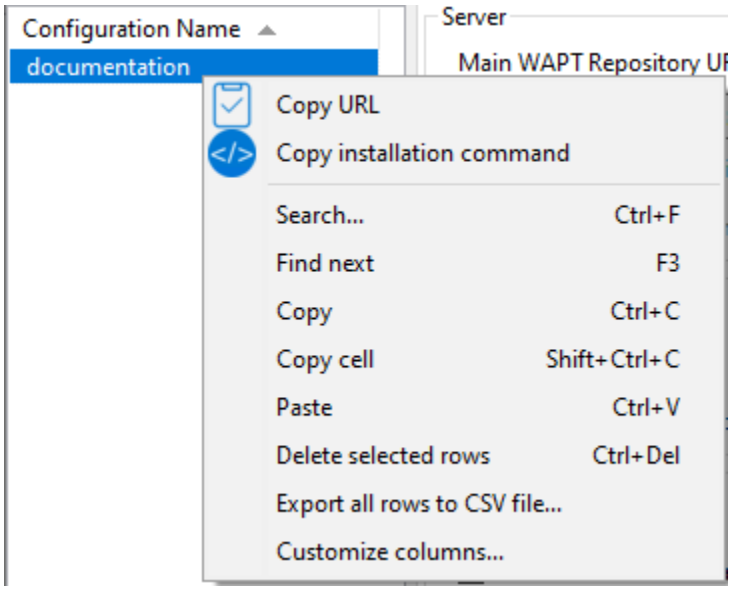


Fig. 31: Copy command

Table 7: Copy options

Value	Description
Copy URL	Gives a download URL to retrieve the `.json` from the server.
Copy installation command	Gives a command to install the configuration for a WAPT agent.

Note: It is possible to install a blank agent and give it the copied installation command to provide the configuration.

CHAPTER

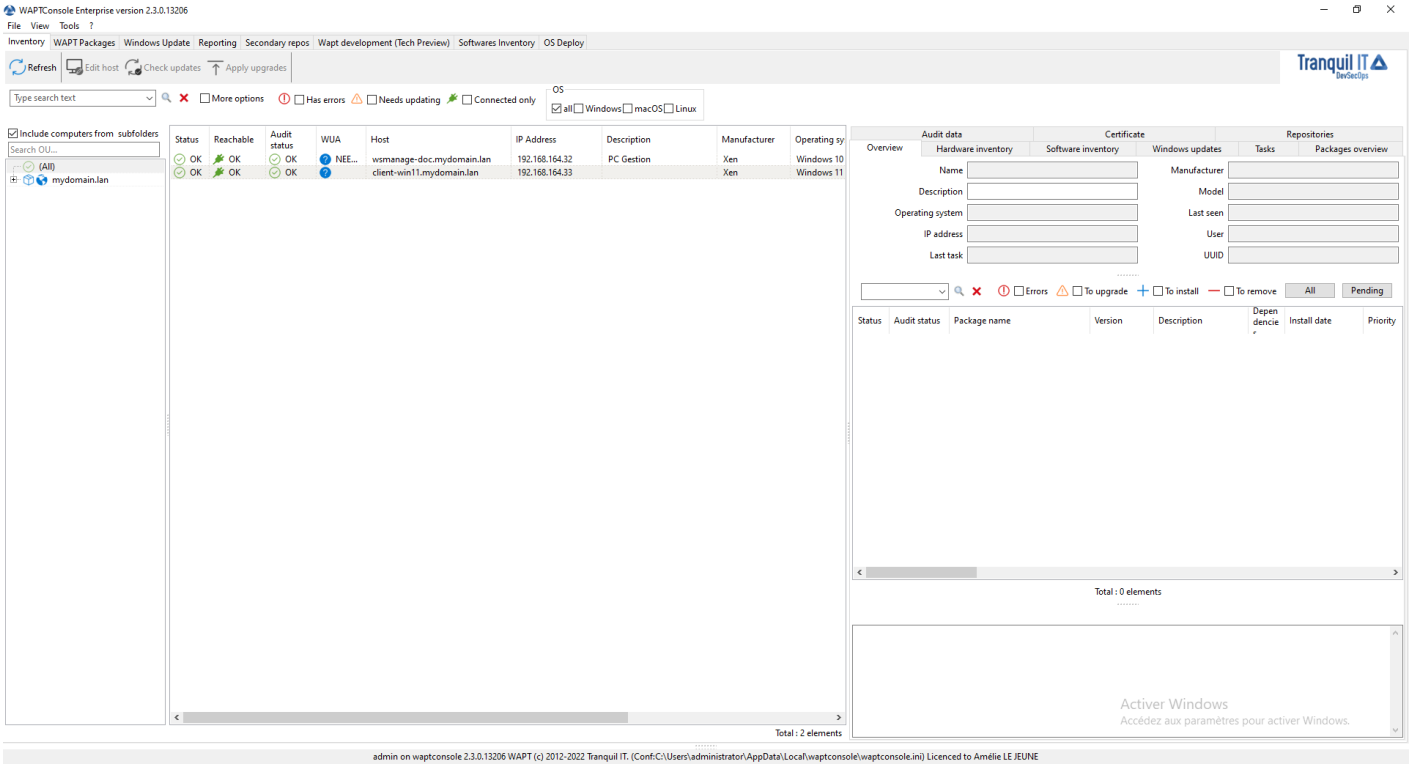
NINETEEN

USING THE WAPT CONSOLE

To install and start the WAPT Console visit the documentation for *installing the WAPT Console*.

If you have skipped the step for creating the WAPT Agent, return to the documentation on *building the WAPT Agent installer*.

On your **management computer**, hosts are displayed in the WAPT Console.



Note: The recommended size for using the WAPT console is 1920x1080 and the minimum size is 1280x1024.

If a host does not appear in the WAPT Console after having installed the WAPT Agent, open the Windows command line utility **`cmd.exe`** on the host and type **`wapt-get register`**.

19.1 Showing the inventory

When the WAPT Agents *register*, they send some information to the WAPT Server.

Information displayed in the WAPT 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.

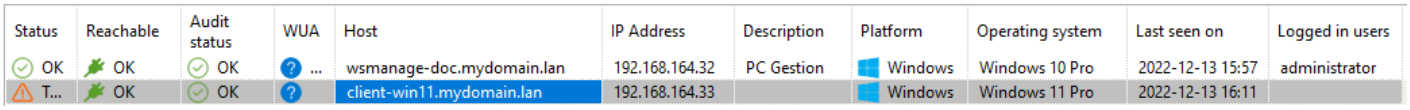
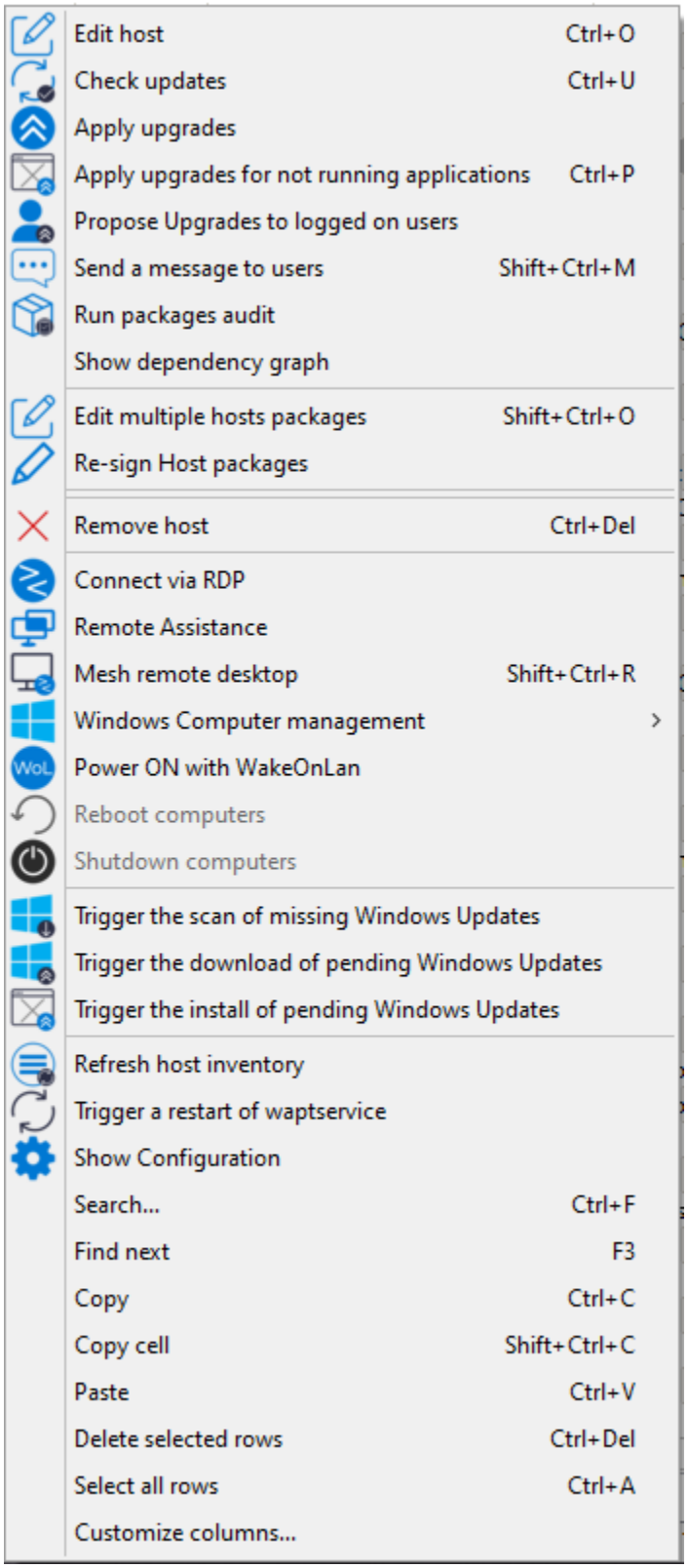


Fig. 1: The WAPT Console displaying the WAPT inventory

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*).

19.2 How to perform actions on the hosts?



Some actions are not available when selecting multiple hosts.

Table 1: List of actions available on a selection of hosts in the WAPT Console

Name	Multi-selection
Edit host	✗
Check updates	✓
Apply upgrades	✓
Apply upgrades for applications not currently running	✓
Propose upgrades for applications not currently running	✓
Send a message to users	✓
Run package audits	✓
Add packages to host dependencies	✓
Remove packages from host dependencies	✓
Re-sign *host* packages	✓
Add package to host conflicts	✓
Remove package from host conflicts	✓
Remove the host	✓
Connect via RDP	✗
Remote Assistance	✗
Mesh remote desktop	✓
Windows Computer management	✗
Update AD Group Policies on hosts	✓
Run CleanMgr on host	✗
Computer management	✗
Local users and groups management	✗
Service management	✗
Power ON with WakeOnLan	✓
Reboot computers	✗
Shutdown computers	✗
Trigger the scan of missing Windows updates	✓
Trigger the download pending Windows updates	✓
Trigger the install of pending Windows updates	✓
Refresh host inventory	✓
Trigger a restart of waptservice	✓

19.3 Send message to selected hosts

With option *Send message to users*, when you do a right-click on select hosts, you can send a html formatted message to connected users.

You can send html format only, no css nor javascript. It is possible to send images, gifs… anything as long you can encode it with html.

Here is an example of code to send a text and an image. Images and gif have to be in base64 encoded or else you may use the *base64image* id.

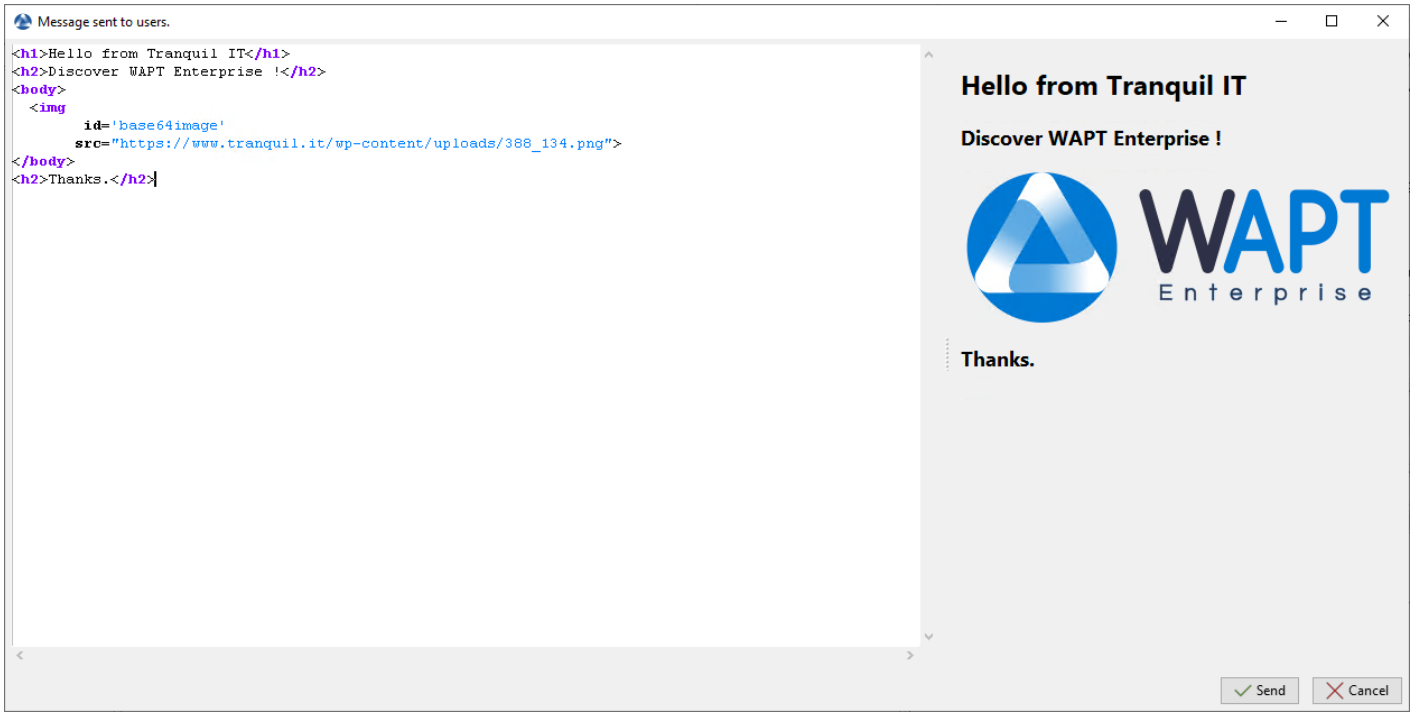
```
<h1>Hello from Tranquil IT</h1>
  <h2>Discover WAPT Enterprise !</h2>
```

(continues on next page)

(continued from previous page)

```
  <body>
  <img
     id='base64image'
    src="https://www.tranquil.it/wp-content/uploads/388_134.png">
  </body>
 <h2>Thanks.</h2>
```

When you create your message, you will have a preview on the right.



When you have finished, click on *Send*, here is how your user will see it.

The first icon above can be customized. Please see how to *use a custom icon in your waptmessage*.

19.4 Importing WAPT packages from external repositories

Importing a WAPT package consists of:

- Importing an existing WAPT package from an external repository.
- Changing its prefix (for example from *tis* to *my-prefix*).
- Re-signing the WAPT package with the *Administrator*'s or the *Code signing* private key to allow the deployment of the imported 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 its repositories.

Without a support contract, Tranquil IT does not guarantee the suitability of the package for your own particular use case, nor does she guarantee the ability of the package to comply with your *Organization*'s internal security policies.

Tranquil IT uses a package build farm to maintain its repository up to date, which is nicknamed LUTI (from the French word "l'outil"). LUTI status is now available publicly at https://luti.tranquil.it.

LUTI monitors when possible the software vendor web site to trigger a package update. It will check the software installer file status on virustotal, and then test the install, uninstall and upgrade of the package. The build results are available in the https://wapt.tranquil.it/wapt-testing repository.

After 5 days, if the virustotal status of the package has not changed, the new package will be uploaded to WAPT main store repository. There is an exception to this rule for web browsers, which are uploaded from wapt-testing to wapt repository after 1/2 an hour.

- Go to the *Private repository* tab.

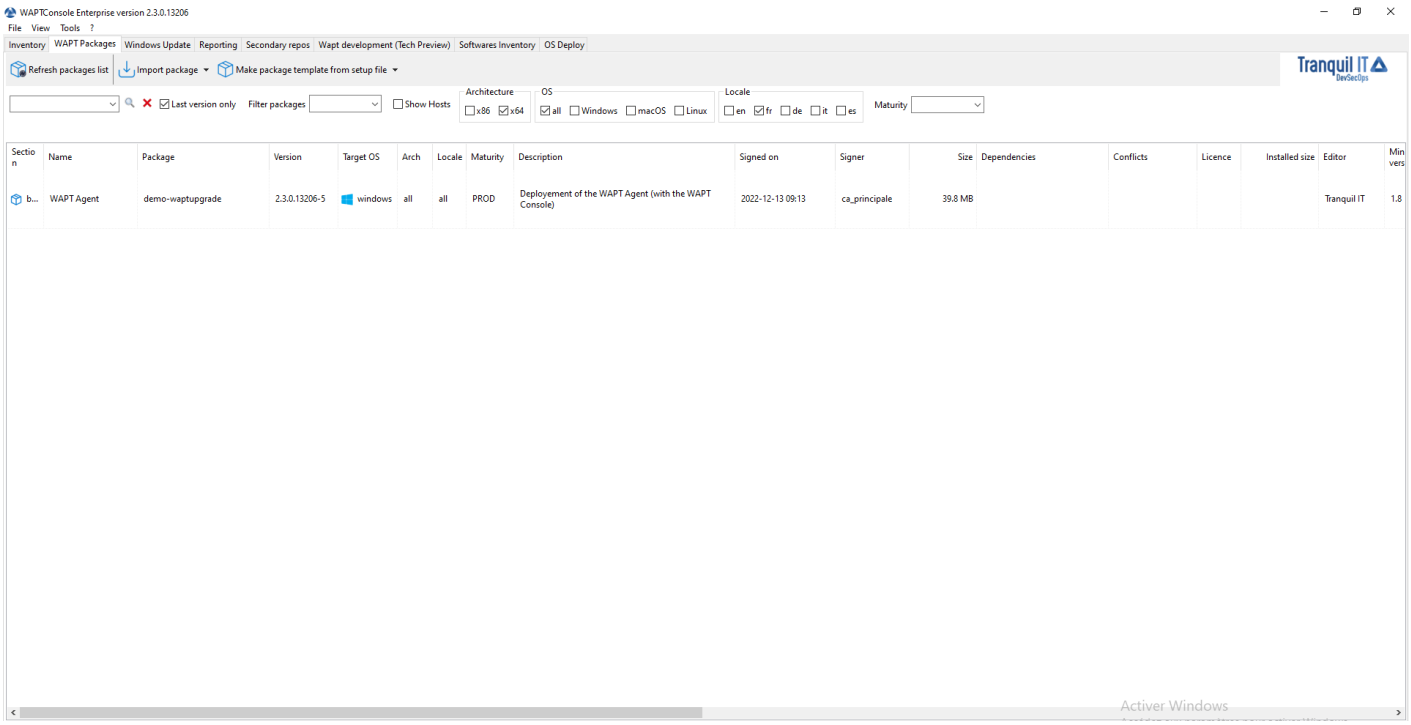


Fig. 2: Available WAPT packages displayed in the WAPT Console

Every WAPT package version loaded 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*.

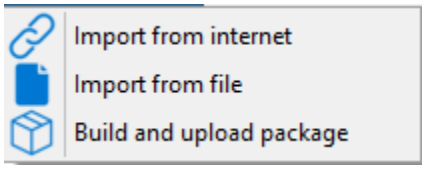
19.4.1 Importing a WAPT package from an external repository on the Internet

This method allows you to download WAPT packages directly from a WAPT repository that is outside of your *Organization*.

By default the Tranquil IT repository is configured. To add another repository, check the documentation for *configuring external repositories*.

By default, the SSL/ TLS certificates to external repositories are verified.

- Click on *Import package* and *Import from Internet*.



Note: The grid view displays the list of available packages on the remote repository. It is possible to choose the architecture, the OS and the locale.

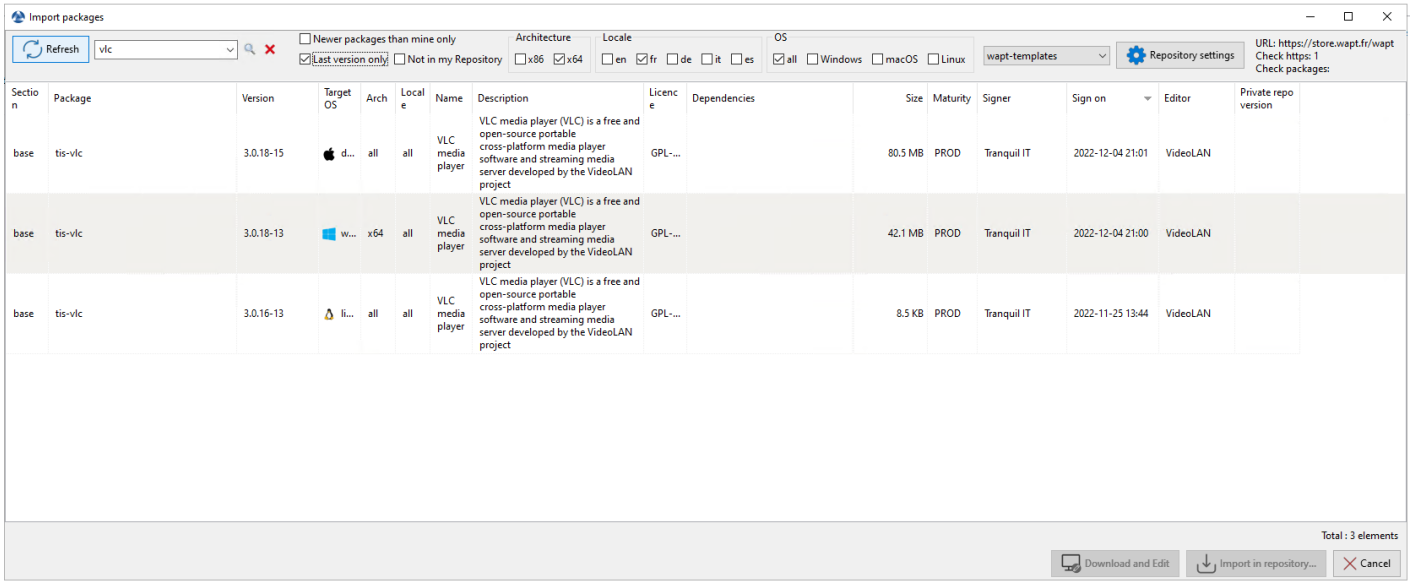
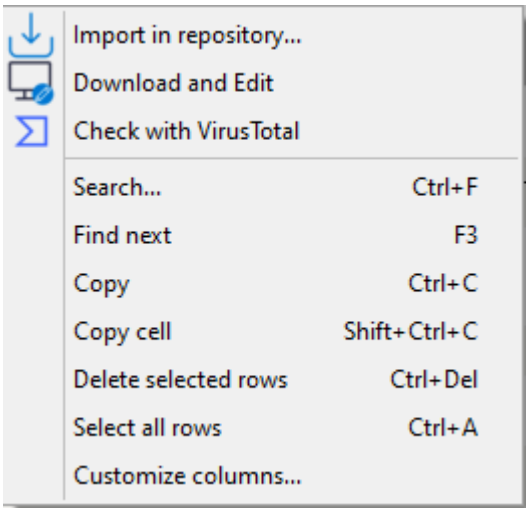


Fig. 3: Imported WAPT package in the local WAPT repository

- There are 2 methods for importing a WAPT package:
  - *Right-click → Import in repository*;



  - Or in the bottom right of the Window *Import in repository*.

Hint: You can check the package with Virus Total in order to check if the package is not flagged in the Antivirus list.

- Validate the importation in your local repository. It is possible to *change the maturity* of a WAPT package before importing the WAPT package into your private repository.

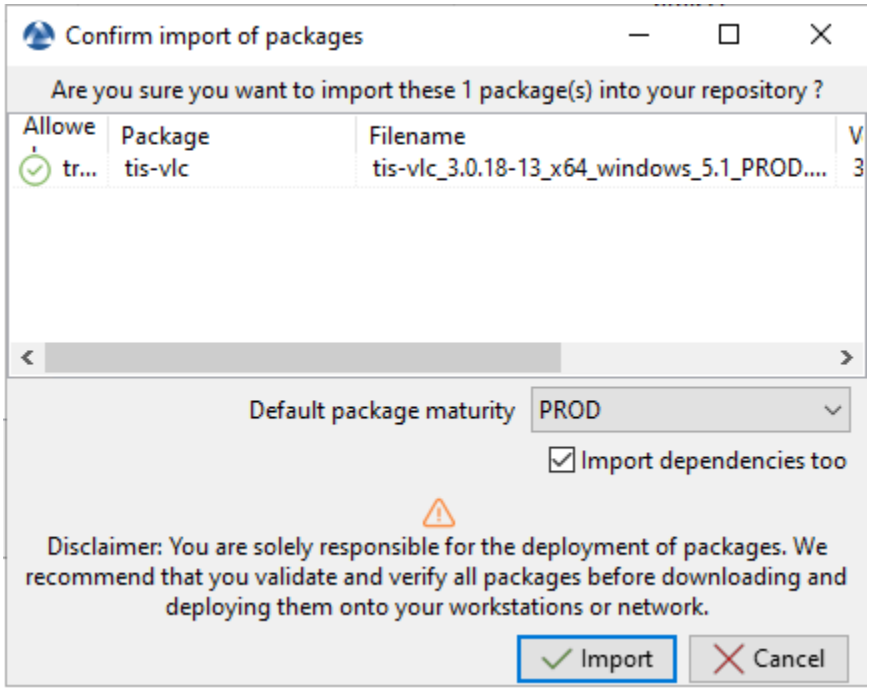


Fig. 4: Dialog box to prepare and confirm the import of a WAPT package into a WAPT repository

Hint: You can update all your packages from your repository by checking *Newer packages than mine only* and *Last version only*, then use `Ctrl+A` to select all newer packages that are available on the Tranquil IT store and that you have on your repository in a older

version, then click on *Import in repository*.

You can search WAPT package that are not in your local repository if you check *Not in my Repository*.

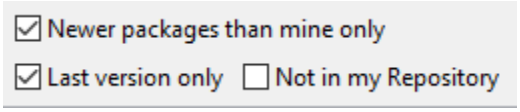


Fig. 5: Checkbox to select newer version and last version of a WAPT package from Internet

- The download of the WAPT package starts.

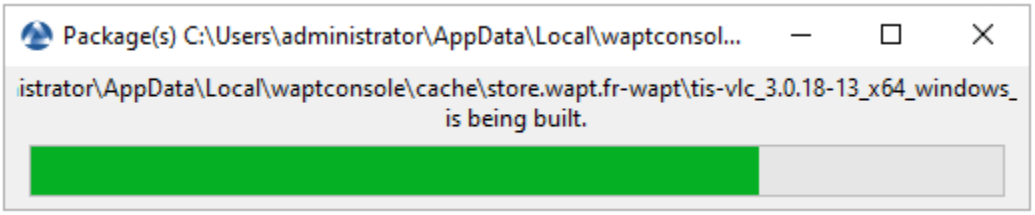


Fig. 6: Progress of the package import process

- Then, enter your private key password.

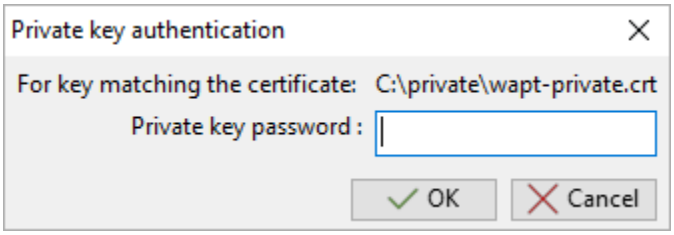


Fig. 7: Entering the password for unlocking the private key in the WAPT Console

The WAPT Console confirms that the WAPT package has been imported into the local WAPT repository.

The package then appears in the local WAPT repository with the Organization's prefix.

Downloading a restricted package version from your repository

Note:

- The WAPT Enterprise thematic store service is currently in "beta" version.
- The WAPT Enterprise theme store is reserved for customers with a valid WAPT Enterprise license.
- Packages downloaded from the WAPT Enterprise store can be deployed on all workstations with a WAPT Enterprise license.
- It is the customer's responsibility to validate the license of the software he is deploying and to respect the constraints related to this license (depending on the software: paying license, deployment in a given context e.g. educational environment only, etc.)

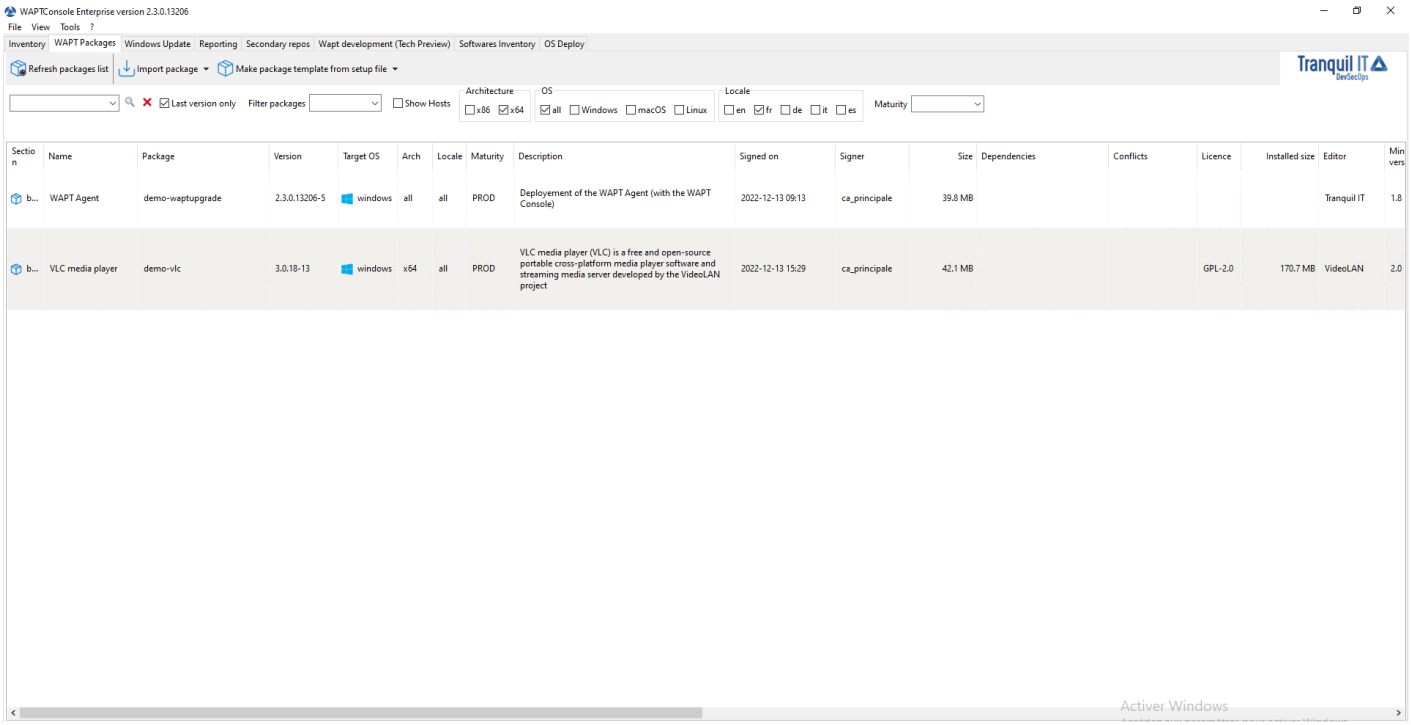


Fig. 8: WAPT Console displaying the imported WAPT package

To meet the needs of its WAPT Enterprise customers, Tranquil IT launched a thematic store program in partnership with its WAPT Enterprise customers in the education and research.

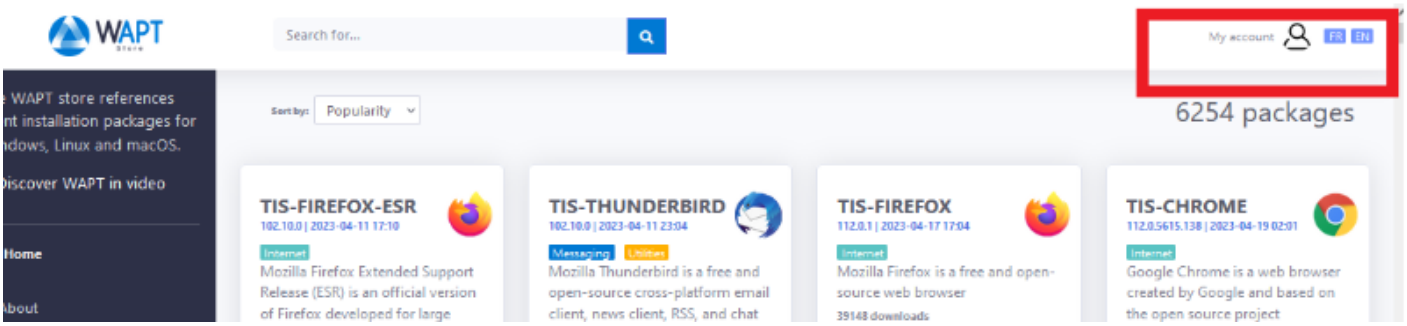
Currently 350 packages have been prepared by Tranquil IT teams and are progressively put online progressively on the wapt store.

To download a package from the thematic store, it is necessary to be identified.

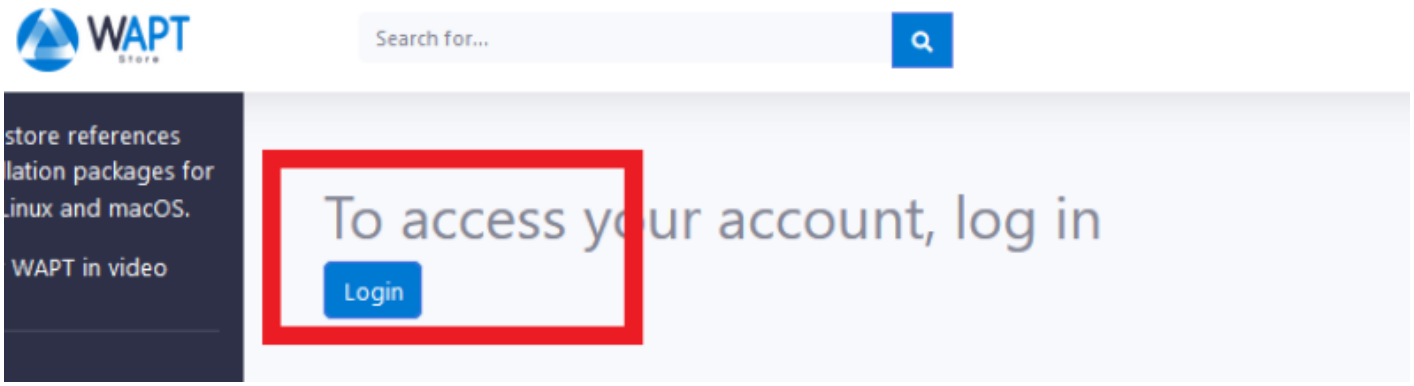
The support of the authentication of the store in the console is supported from the next WAPT 2.4 version. If you are in WAPT 2.3 version, you have to download the package from the website and import it into your WAPT console.

To authenticate yourself on the thematic store, follow the procedure below:

- Connect to the following url: https://wapt.tranquil.it/store and click on the **my account** link



- Click on login.



- The store will redirect you to the Tranquil IT SSO client authentication. If you are logging in for the first time, click on **forgot password**.
- An email will be sent to you sent to you to initialize your password.
- After changing your password, click on *Reset token* and a new token will appear, copy it.
- In your WAPT Console, go to *Tools* → *External Repositories Settings* then select the Tranquil IT store in the repository list, click on *Show advanced parameters* then add you user (your email address) and your token copied before.

Important: The email address used must be the one provided to Tranquil IT sales or technical service. If your email address is not known to our services, it will not be possible to authenticate you or to reset your password.

If you have a connection problem you can call your technical or commercial contact person at Tranquil IT sales representative.

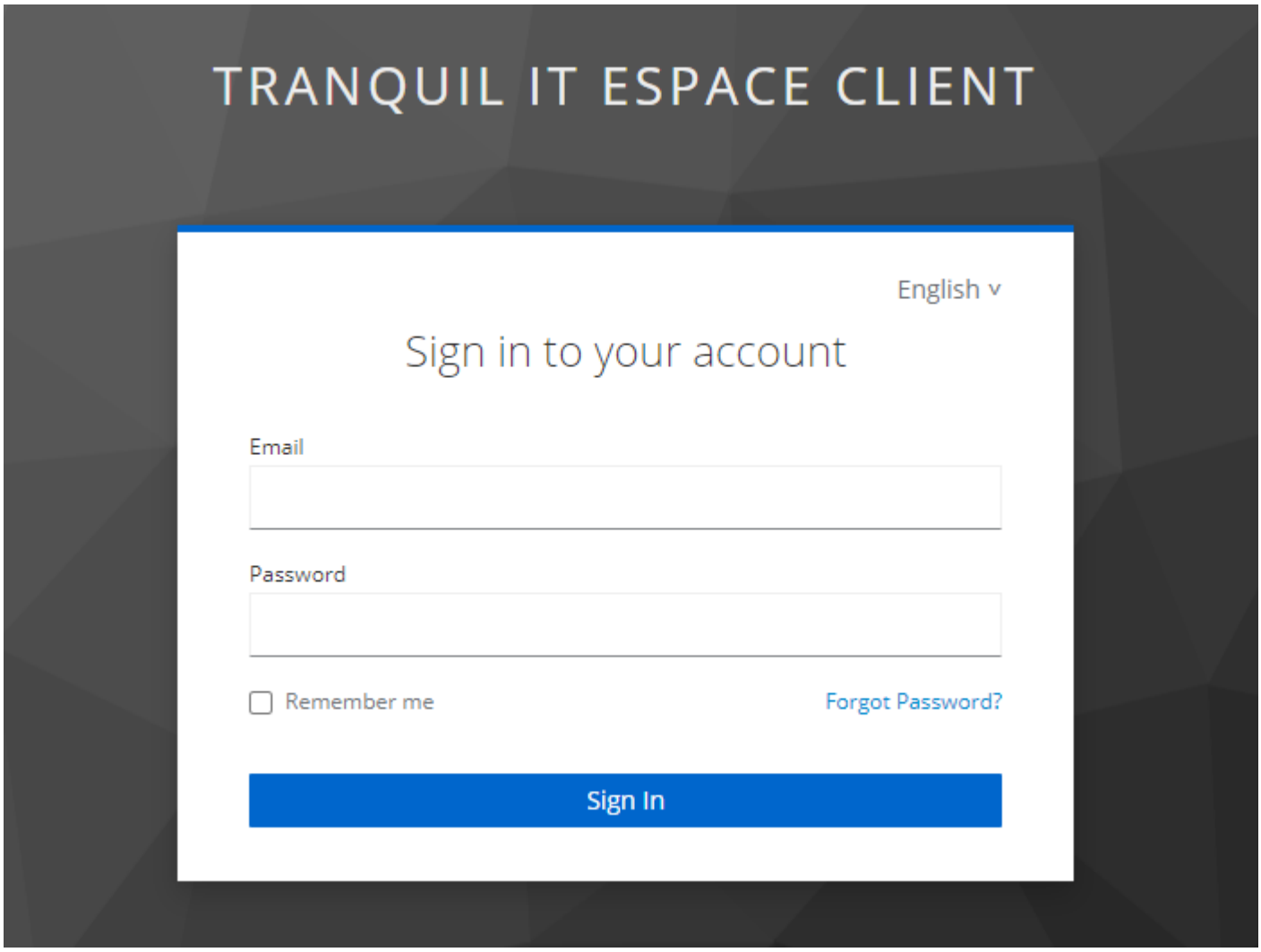
Once you are connected you can then download the WAPT Enterprise packages.

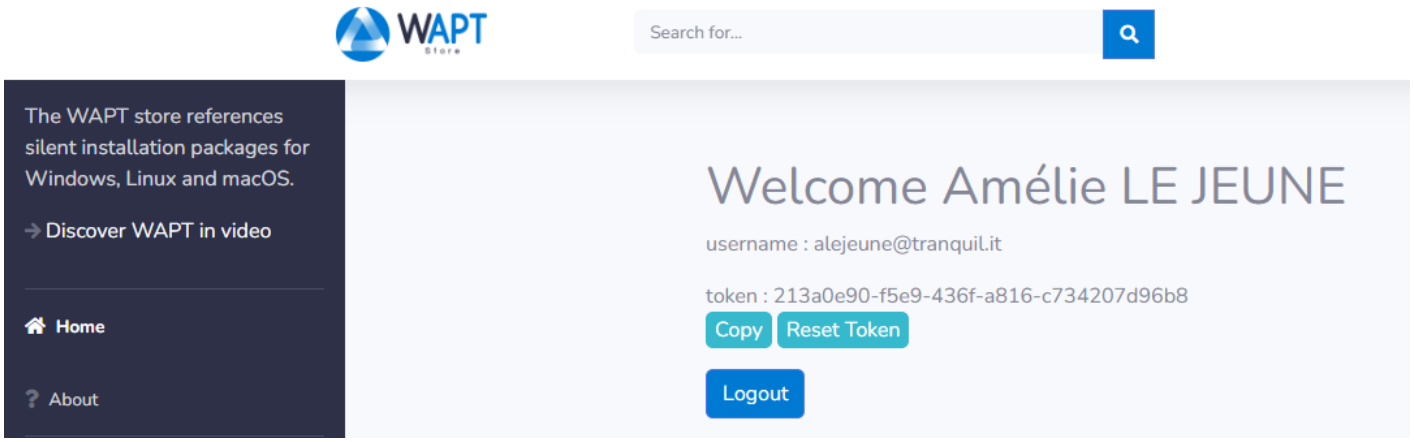
Downloading a newer package version from your repository

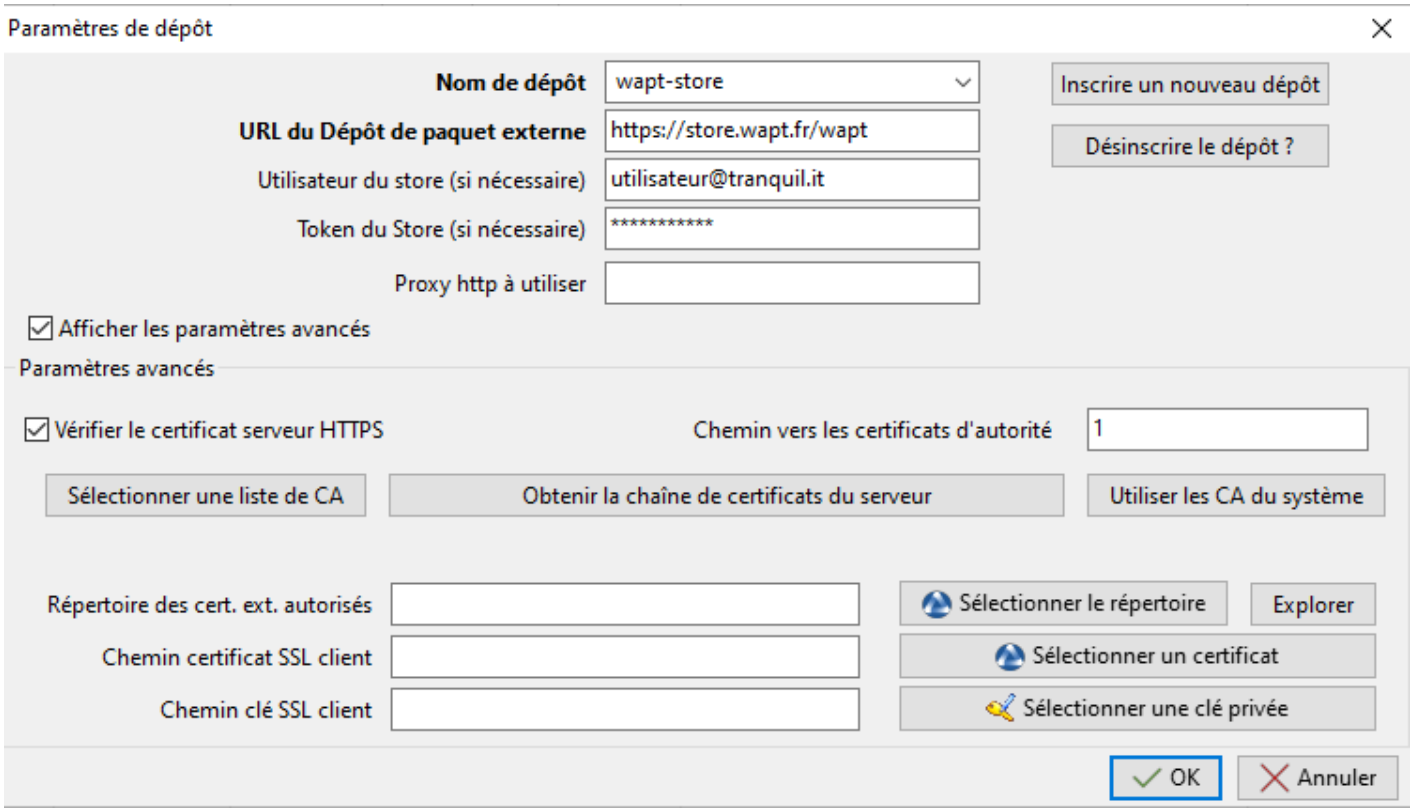
If you see in your repository a package version in red and **bold**, it means a newer version exist on the public store (by default the store is Tranquil IT's). In the *WAPT Packages* tab, *Version* is the version of the package in the local repository; *Store Version* is the public store version and *Software Version* is the Luti version when Luti knows the editor software version.

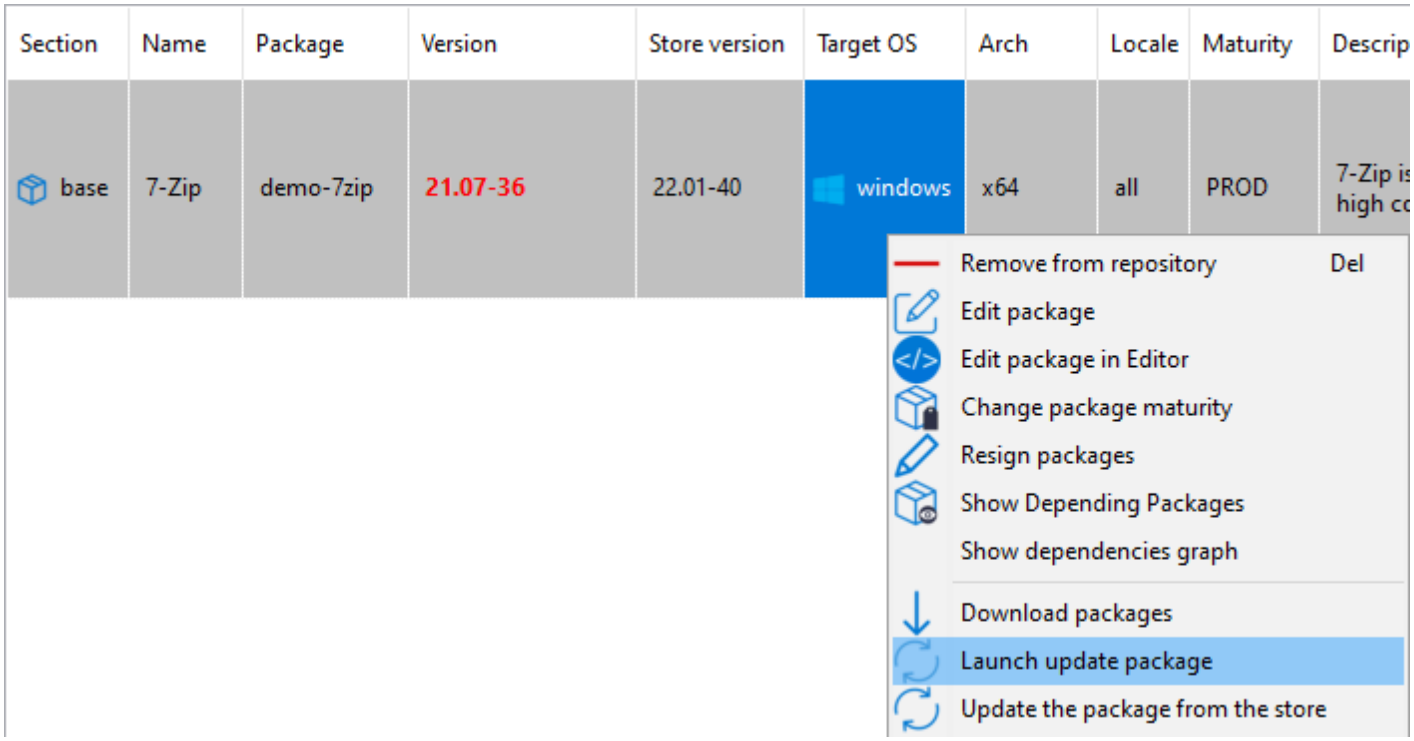
You have 2 options to upgrade the WAPT package in the local repository:

- Launch **`update_package`** with right-click on the package to be updated then *Launch update package*. It will execute the **`update_package`** defined in the package. If there is no code for updating the software, the command will do nothing. This method has one invonvenient: if the `setup.py` has been improved, you will not benefit from the improvement. For more, please see *the documentation on updating a WAPT package*.
- Click on *Update the package from the store*. The method will fetch the latest package version from the public repositories (by default Tranquil IT). The benefit of this method is that you will benefit from code improvement in the `setup.py`. Tjhis is therefore the recommended method. You may want to consult this documentation of *how to update a WAPT package from a public repository*.


TRANQUIL IT ESPACE CLIENT
English
Sign in to your account
Email
Password
Remember me
Forgot Password?
Sign In


WAPT
Search for...
The WAPT store references silent installation packages for Windows, Linux and macOS.
Discover WAPT in video
Home
About
Welcome Amélie LE JEUNE
username : alejeune@tranquil.it
token : 213a0e90-f5e9-436f-a816-c734207d96b8
Copy
Reset Token
Logout


Paramètres de dépôt
Nom de dépôt
wapt-store
Inscrire un nouveau dépôt
URL du Dépôt de paquet externe
https://store.wapt.fr/wapt
Désinscrire le dépôt ?
Utilisateur du store (si nécessaire)
utilisateur@tranquil.it
Token du Store (si nécessaire)
****
Proxy http à utiliser
Afficher les paramètres avancés
Paramètres avancés
Vérifier le certificat serveur HTTPS
Chemin vers les certificats d'autorité
1
Sélectionner une liste de CA
Obtenir la chaîne de certificats du serveur
Utiliser les CA du système
Répertoire des cert. ext. autorisés
Sélectionner le répertoire
Explorer
Chemin certificat SSL client
Sélectionner un certificat
Chemin clé SSL client
Sélectionner une clé privée
OK
Annuler


Section
Name
Package
Version
Store version
Target OS
Arch
Locale
Maturity
Descrip
base
7-Zip
demo-7zip
21.07-36
22.01-40
windows
x64
all
PROD
7-Zip is high co
Remove from repository
Del
Edit package
Edit package in Editor
Change package maturity
Resign packages
Show Depending Packages
Show dependencies graph
Download packages
Launch update package
Update the package from the store

Changing the maturity of a WAPT package before importing it into the repository

It is possible to change the maturity of a WAPT package before loading it into your private repository by choosing **DEV**, **PREPROD** or **PROD** in *Default package maturity*.

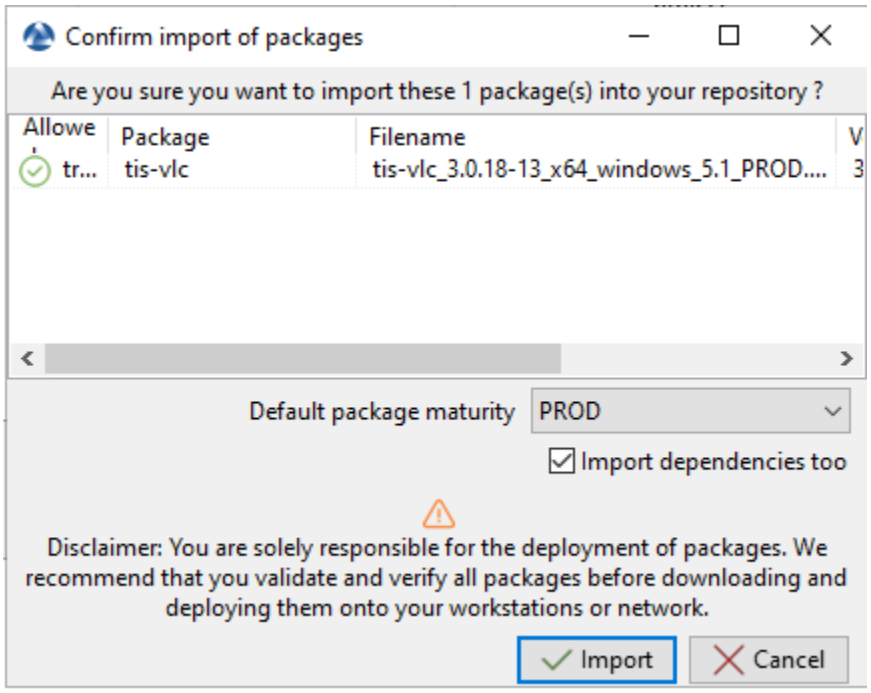
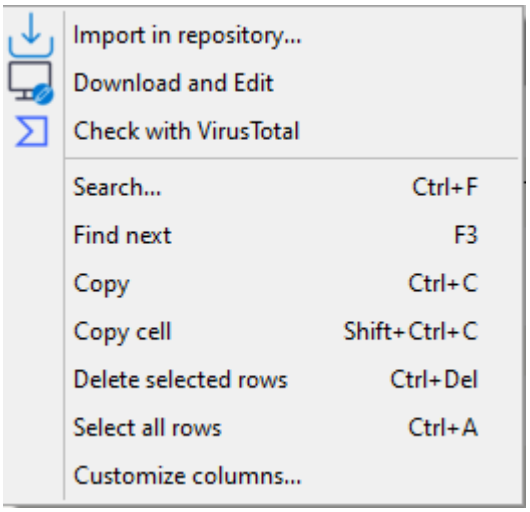


Fig. 9: Dialog box to prepare and confirm the import of a WAPT package into a WAPT repository

Editing a package before importing it

It is possible to edit a package downloaded from an external repository before importing it into your main WAPT repository.

- To do this, **2** choices are available:
  - *Right-click → Download and Edit*;



- or by clicking *Download and Edit* on the bottom right of the window;

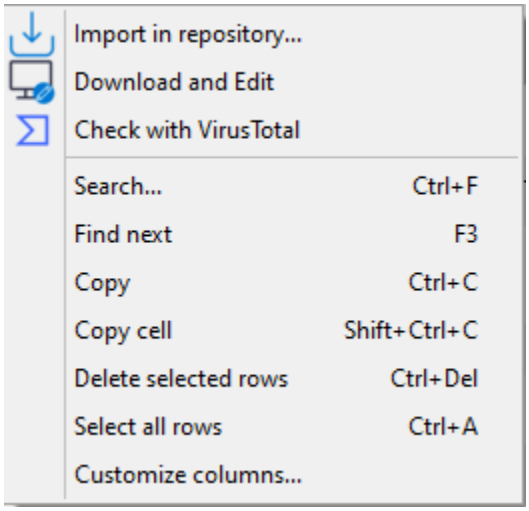
PyScripter, if installed, will open the `control` and `setup.py` files of the WAPT package.

For more information, visit the documentation on *creating WAPT packages from scratch*.

19.4.2 Importing WAPT packages from a local file

You can import a `.wapt` file from any storage.

- Click on *Import package* and then *Import from file*.



- Select the file to import.
- Click on the *Open* button to import the file.

The WAPT Console confirms that the package has been imported in the local WAPT repository.

The package then appears in the local WAPT repository with the Organization's prefix.

Note: It is not possible to change the maturity before importing here.

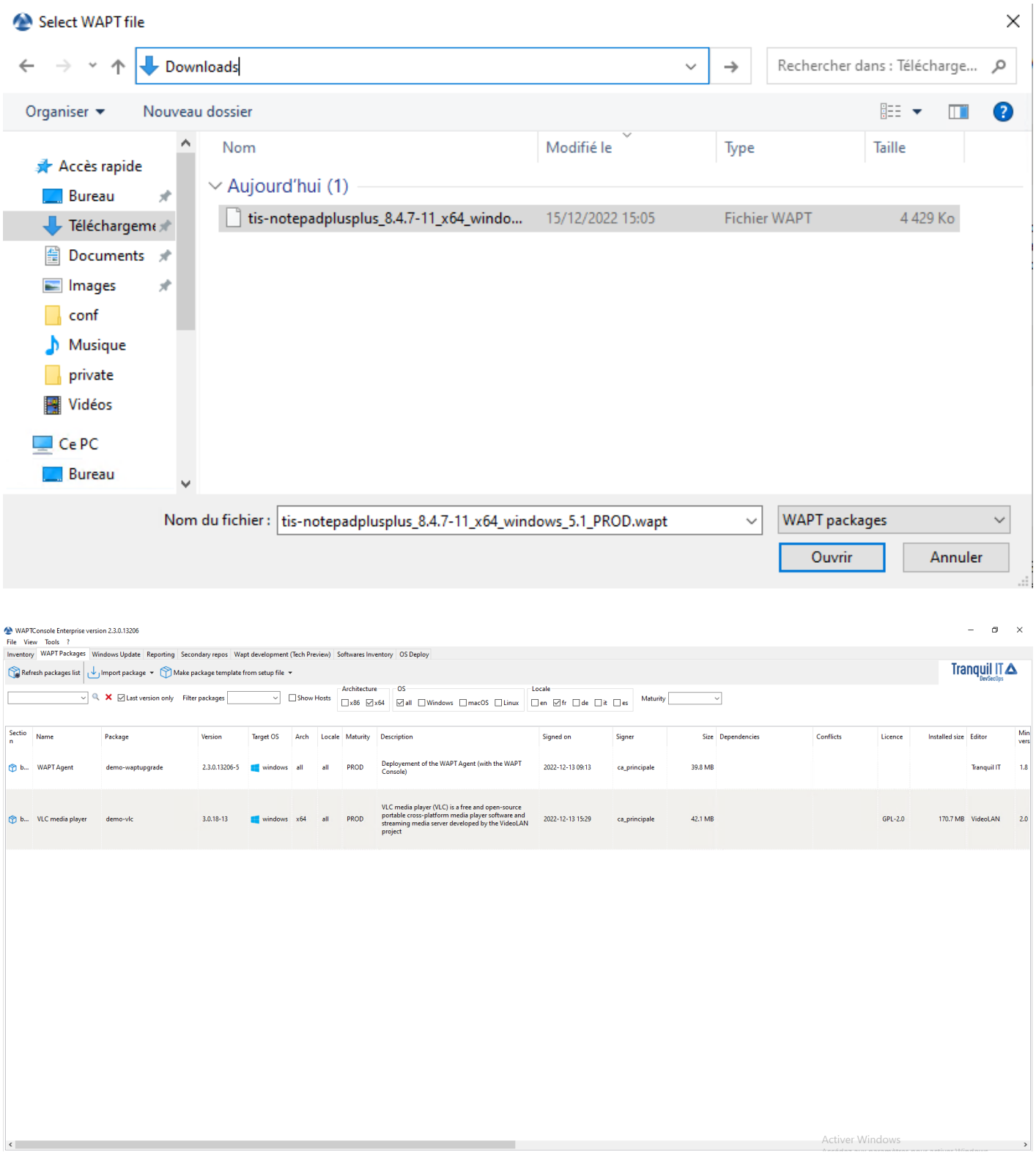


Fig. 10: WAPT Console displaying the imported WAPT package

When uploading a new WAPT package to the private repository, the changing of the prefix and the re-signing of the WAPT package are transparent and automatic.

19.5 Managing WAPT packages hosted in the repository

In the *WAPT Packages* tab, the list of packages currently available in the WAPT repository appears. By default, the WAPT Console will only show the latest version of WAPT packages.

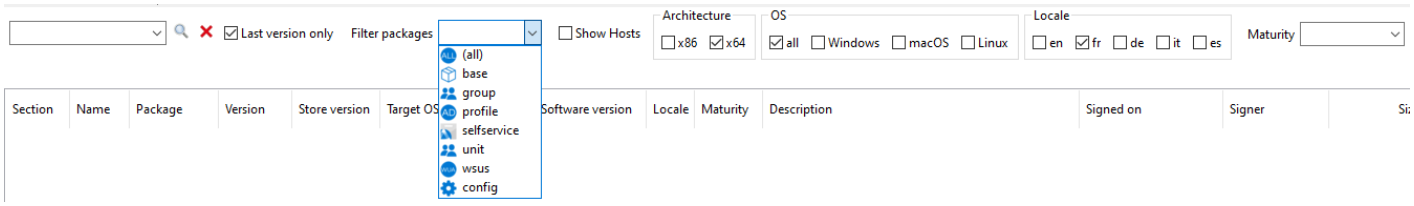


Table 2: Listing of items in the window

Label	Description
The *Type name or description* search bar	Allows to search by WAPT package name or description.
The *Last version only* checkbox	Allows to display all version of WAPT packages in the WAPT Console.
The *Filter packages* dropdown menu	Allows to filter WAPT packages by type (*all*, *base*, *group*, *profile*, *selfservice*, *unit*, *waptwua*).
The *Show hosts*	Displays the hosts on which the selected WAPT package is installed.
The *Architecture x86* checkbox	Allows to filter on hosts having a x86 based processor architecture.
The *Architecture x64* checkbox	Allows to filters on hosts having a x64 based processor architecture.
The *OS all* checkbox	Allows to filter hosts based on any OS (Operating System).
The *OS Windows* checkbox	Allows to filter hosts based on the Windows OS.
The *OS macOS* checkbox	Allows to filter hosts based on the macOS OS.
The *OS Linux* checkbox	Allows to filter hosts based on the Linux OS.
The *Locale en* checkbox	Allows to filter hosts localized in English.
The *Locale fr* checkbox	Allows to filter hosts localized in French.
The *Locale de* checkbox	Allows to filter hosts localized in German.
The *Locale it* checkbox	Allows to filter hosts localized in Italian.
The *Locale es* checkbox	Allows to filter hosts localized in Spanish.
The *Maturity* dropdown list	Allows to filter on the maturity level configured on the hosts.

19.5.1 Doing a search based on a WAPT package

In the 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 host (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.

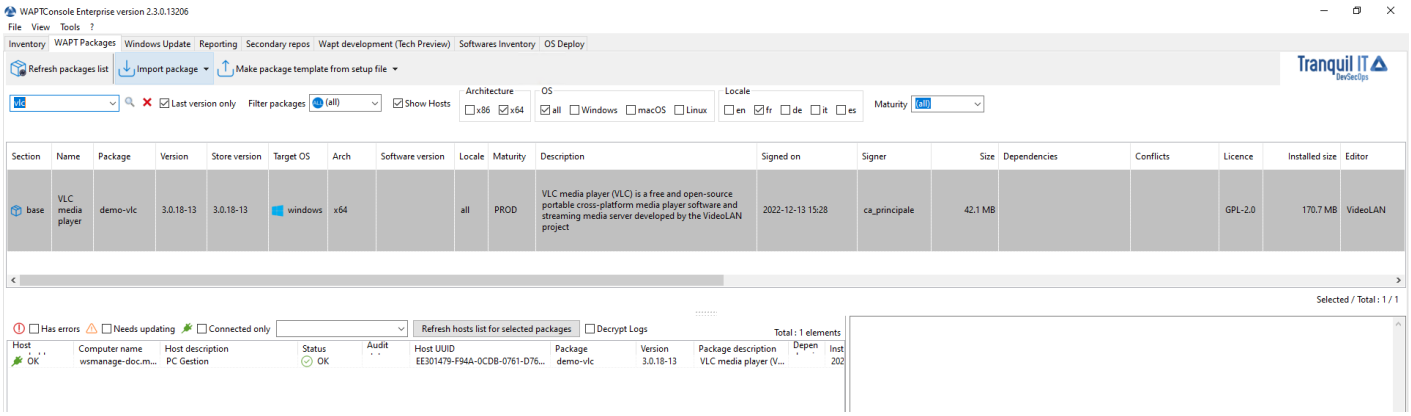


Fig. 11: Filtering hosts based on a WAPT package

19.5.2 Showing package dependencies

In the *Repository* tab, select the WAPT package and then click on *Show Depending Packages*. It will show whether the WAPT package has a dependency.

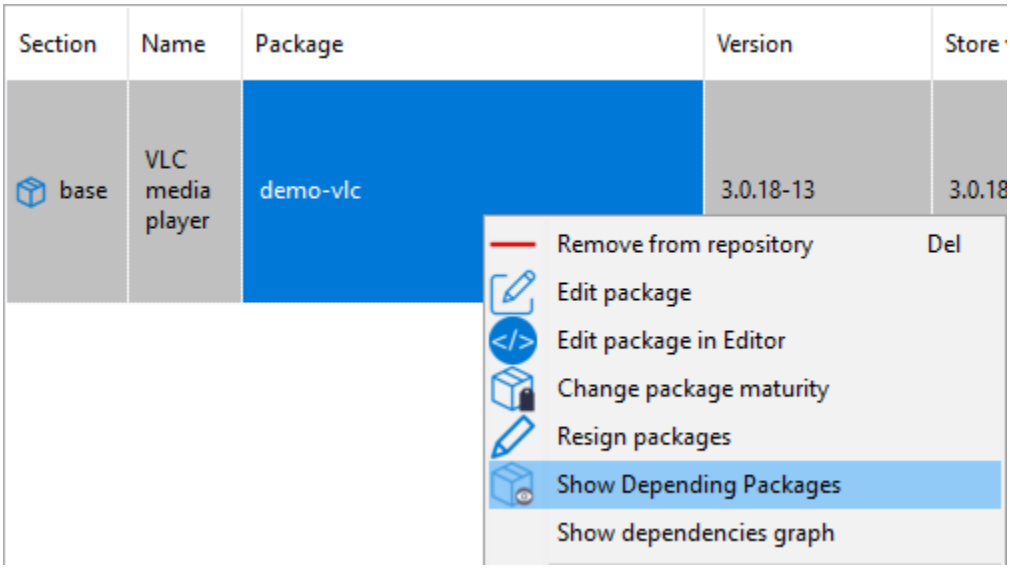


Fig. 12: Right-click on selected package to see if some other software is a dependency

In this example, the *unit* package *Computers* has **`demo-vlc`** in its dependencies.

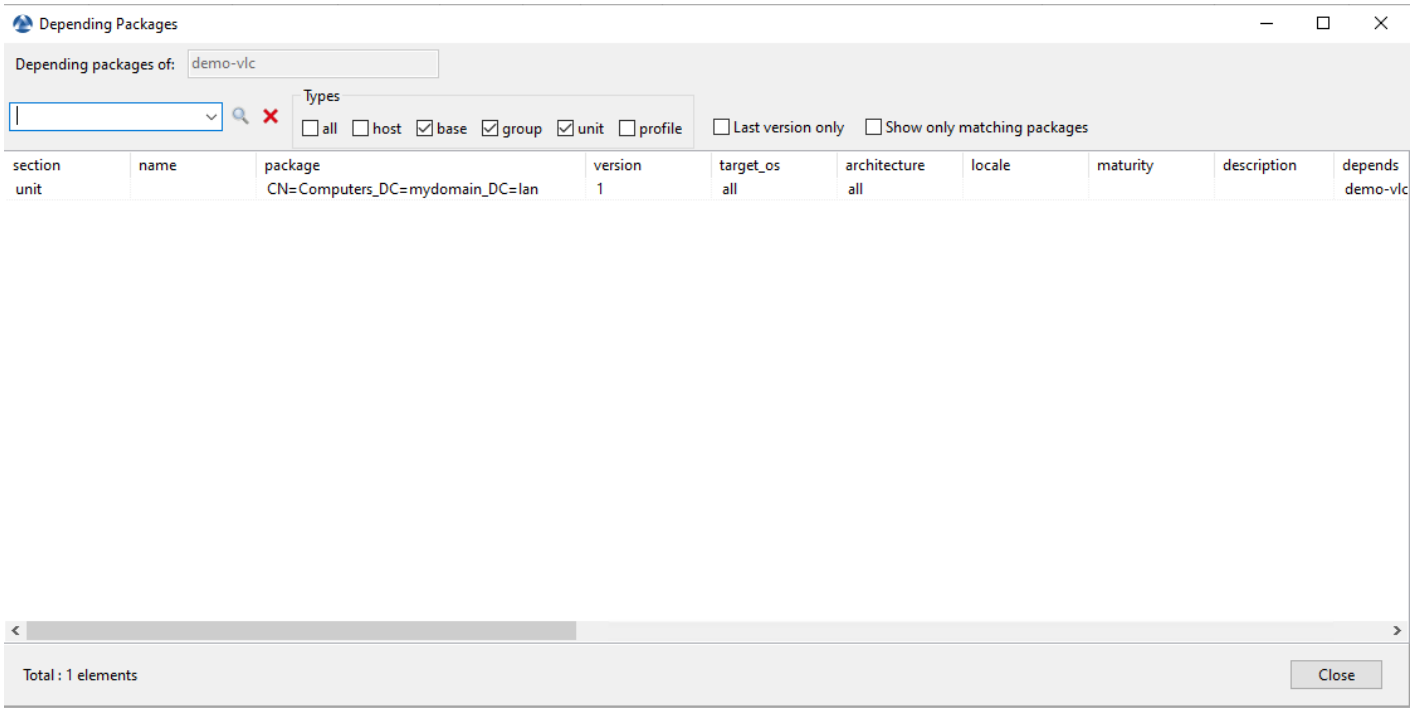


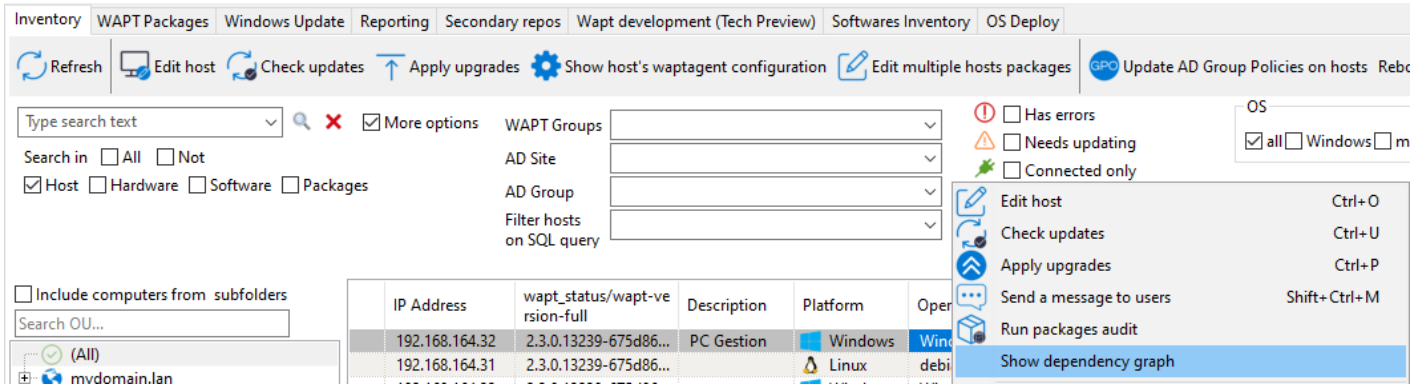
Fig. 13: Example of a package of which the selected package has a dependency

19.5.3 Showing dependencies graph

In the repository, select the WAPT package and then click on *Show Dependencies graph* to see all dependencies and sub-dependencies.

In this example, we can see all dependencies and sub-dependencies of `demo-waptdev` package.

Hint: You can do the same thing with a host in *Inventory* tab.



In this example, we can see all dependencies and sub-dependencies of the selected host.

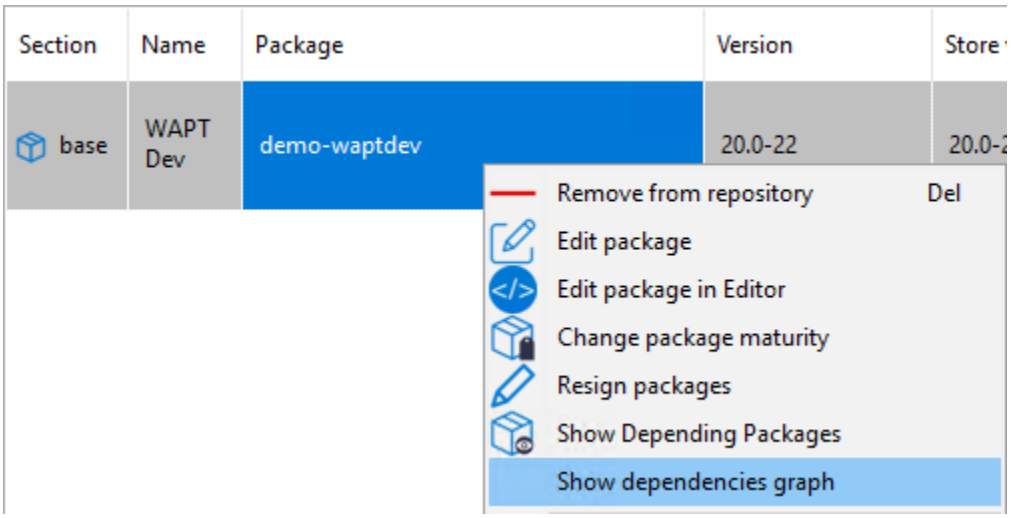


Fig. 14: Right-clicking on a package to see its dependency graph

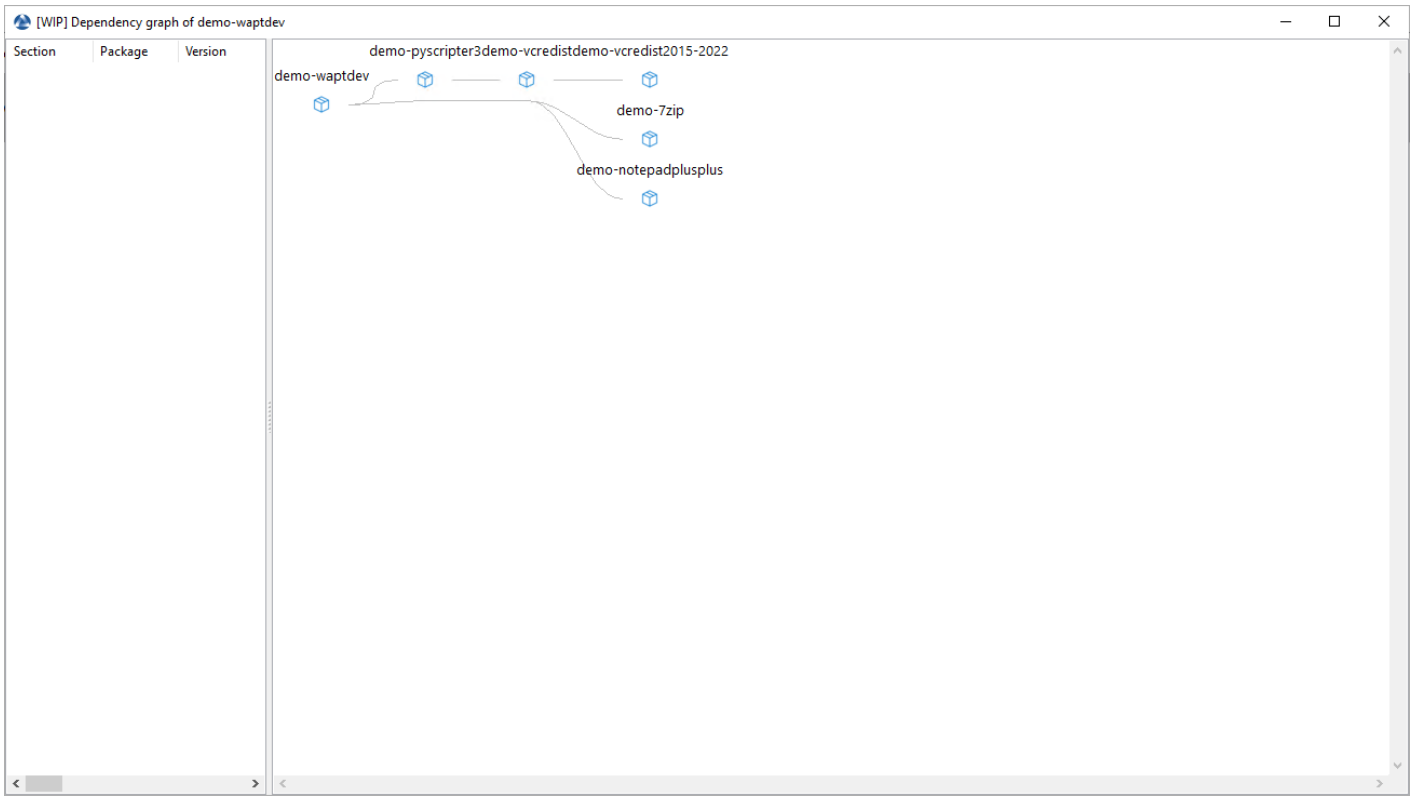


Fig. 15: Exemple of a package with its graph dependencies

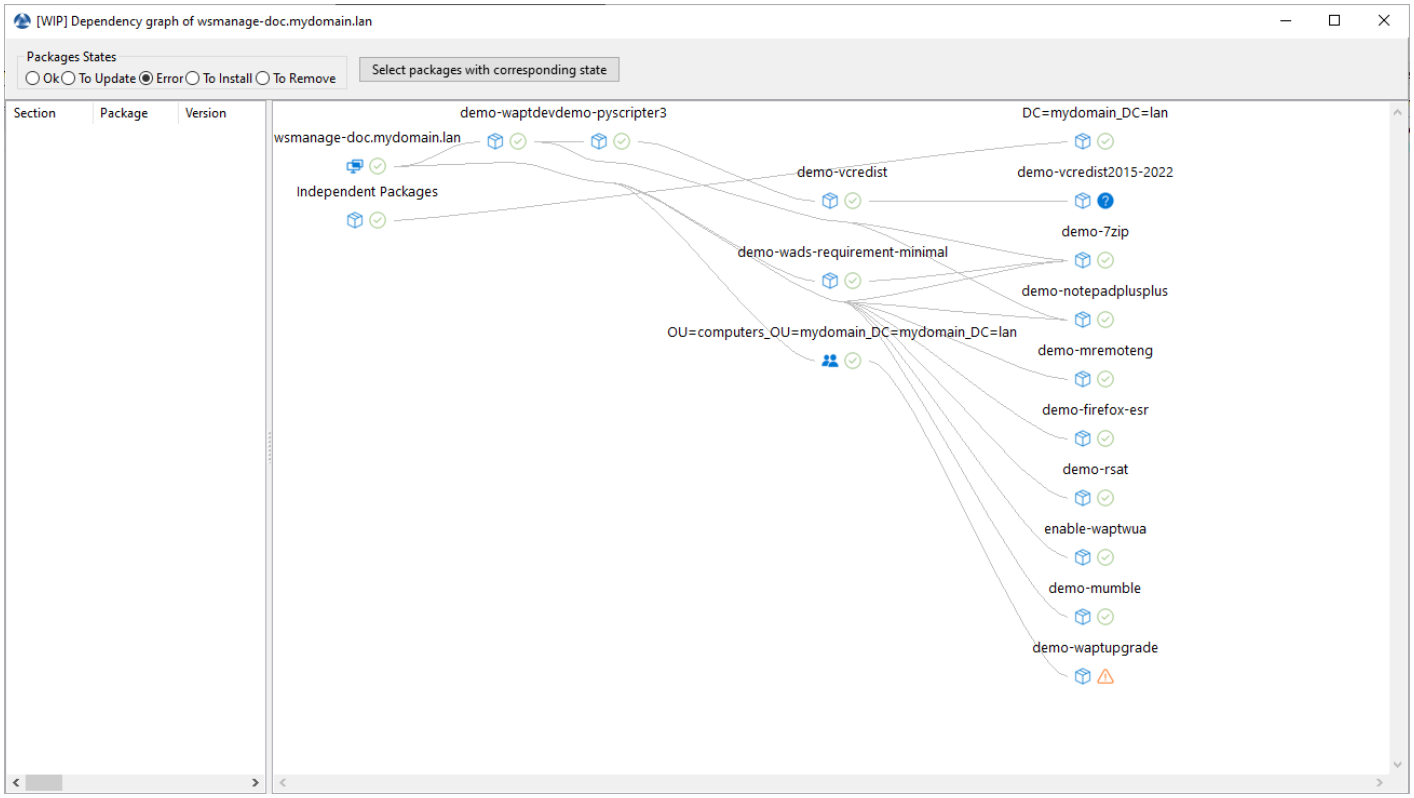


Fig. 16: Exemple of a host with its graph dependencies

19.5.4 Updating the package from the Tranquil IT repository

When you want to upgrade one or several WAPT Packages with the *Update the package from the store* option, here is how it works. First, select on or several package you want to upgrade.

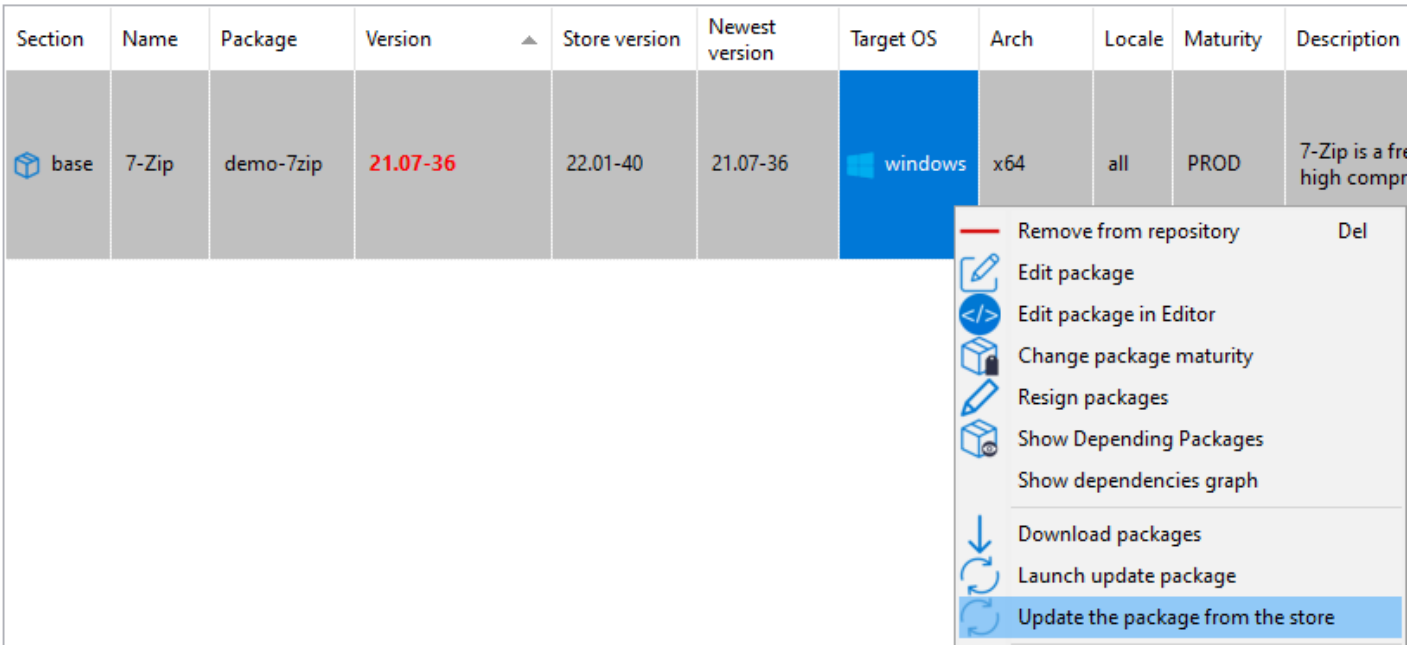


Fig. 17: WAPT Console displaying a newer package version

Then a window with the selected packages will open, you can click on *Update the package from the store*. After that, it will fetch the new version from the Tranquil IT repository with our code improvement. You will see the same dialog box when you *import from internet*.

19.5.5 Updating WAPT package in the local repository

Hint: Tranquil IT recommands you to upgrade your WAPT Packages initially downloaded from our store with this method : *Updating the package from the Tranquil IT repository*

When you want to upgrade one or several WAPT Packages with the *Launch update package* option, here is how it works. First, select one or several packages you want to upgrade.

Then a window with the selected packages will open, you can click on *Launch update_package on selected packages*. After that, if you did not check *Upload directly*, you will have to click on *Upload selected packages* in order to sign, build and upload the package to the WAPT Server.

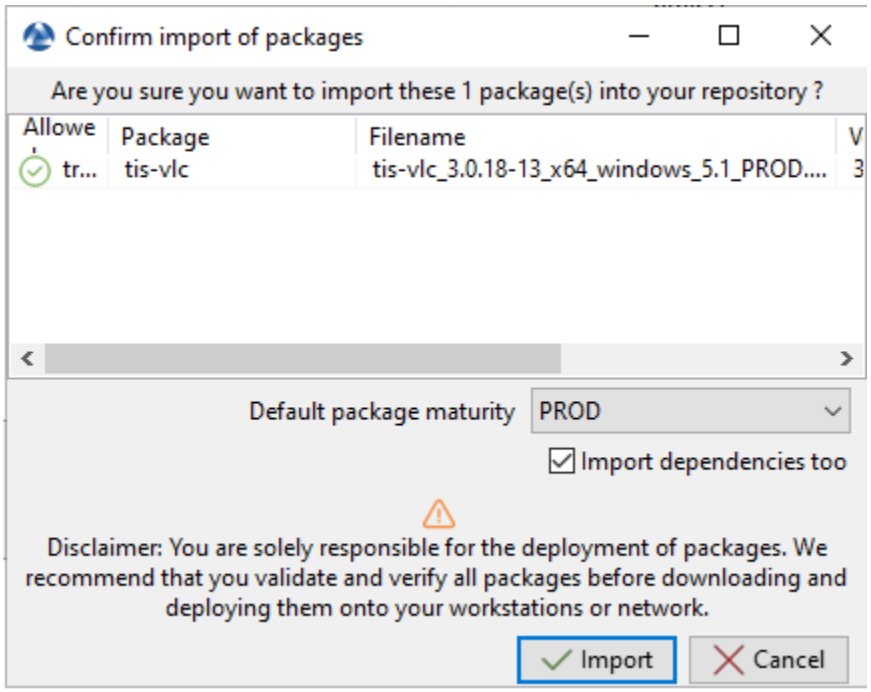
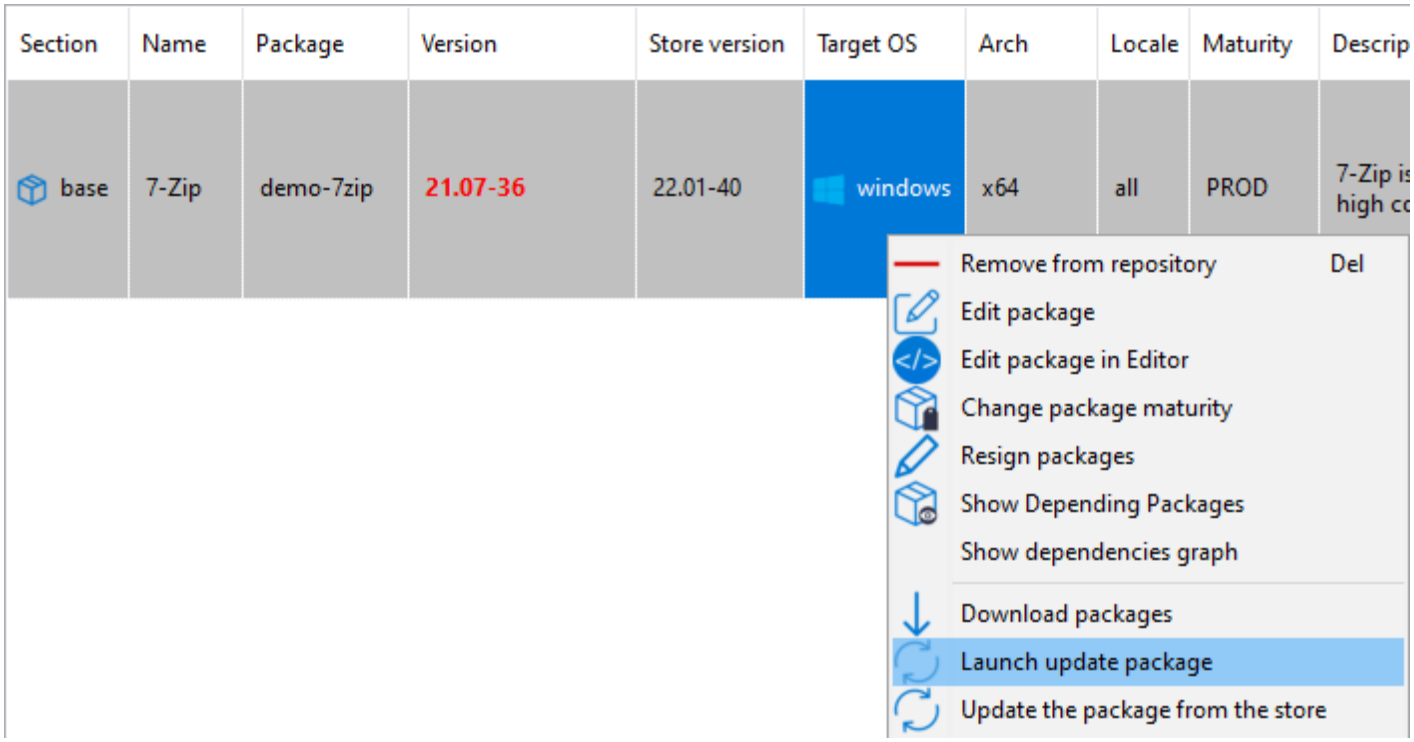


Fig. 18: WAPT Console update_package windows displaying a newer package version



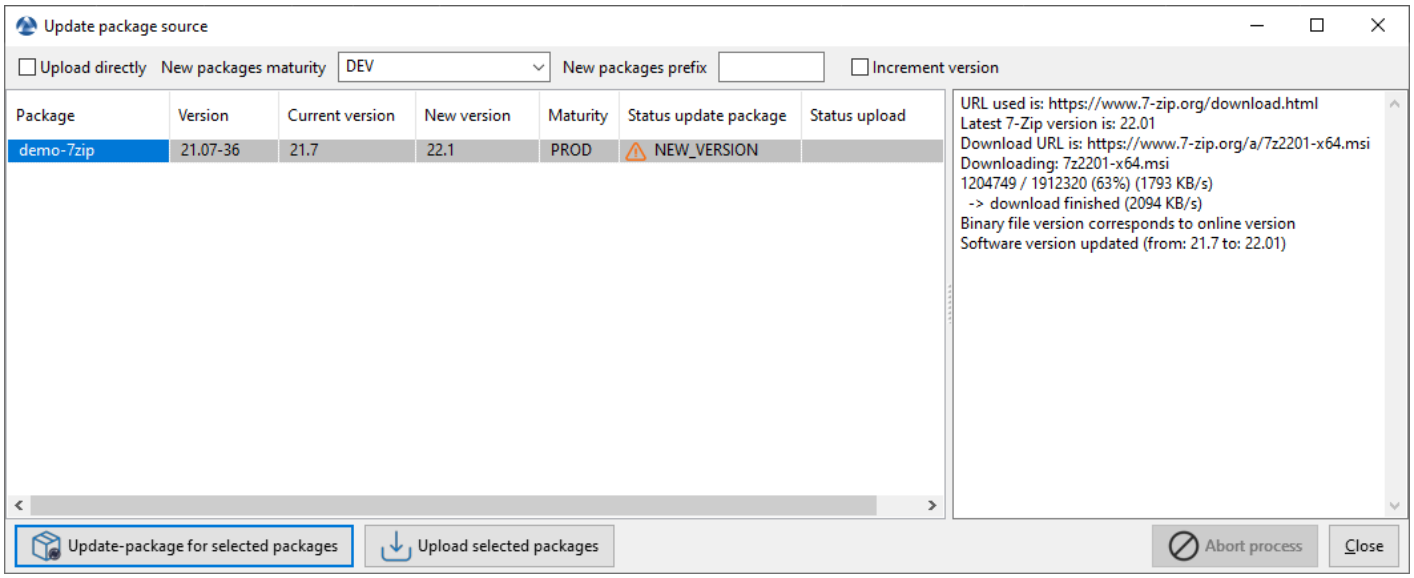
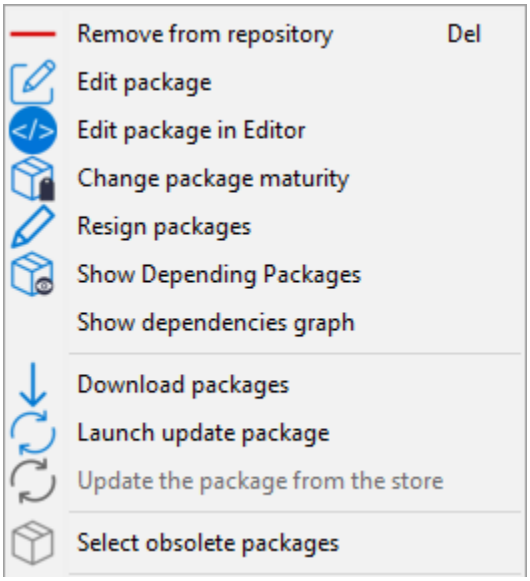


Fig. 19: WAPT Console update_package windows displaying a newer package version

19.5.6 Changing the maturity of a WAPT package after having imported it

When a package is imported on a WAPT repository it is possible to change the maturity by rigth-clicking on the WAPT package. Choose the maturity of the WAPT package to import using the *Change packages maturity* menu item.



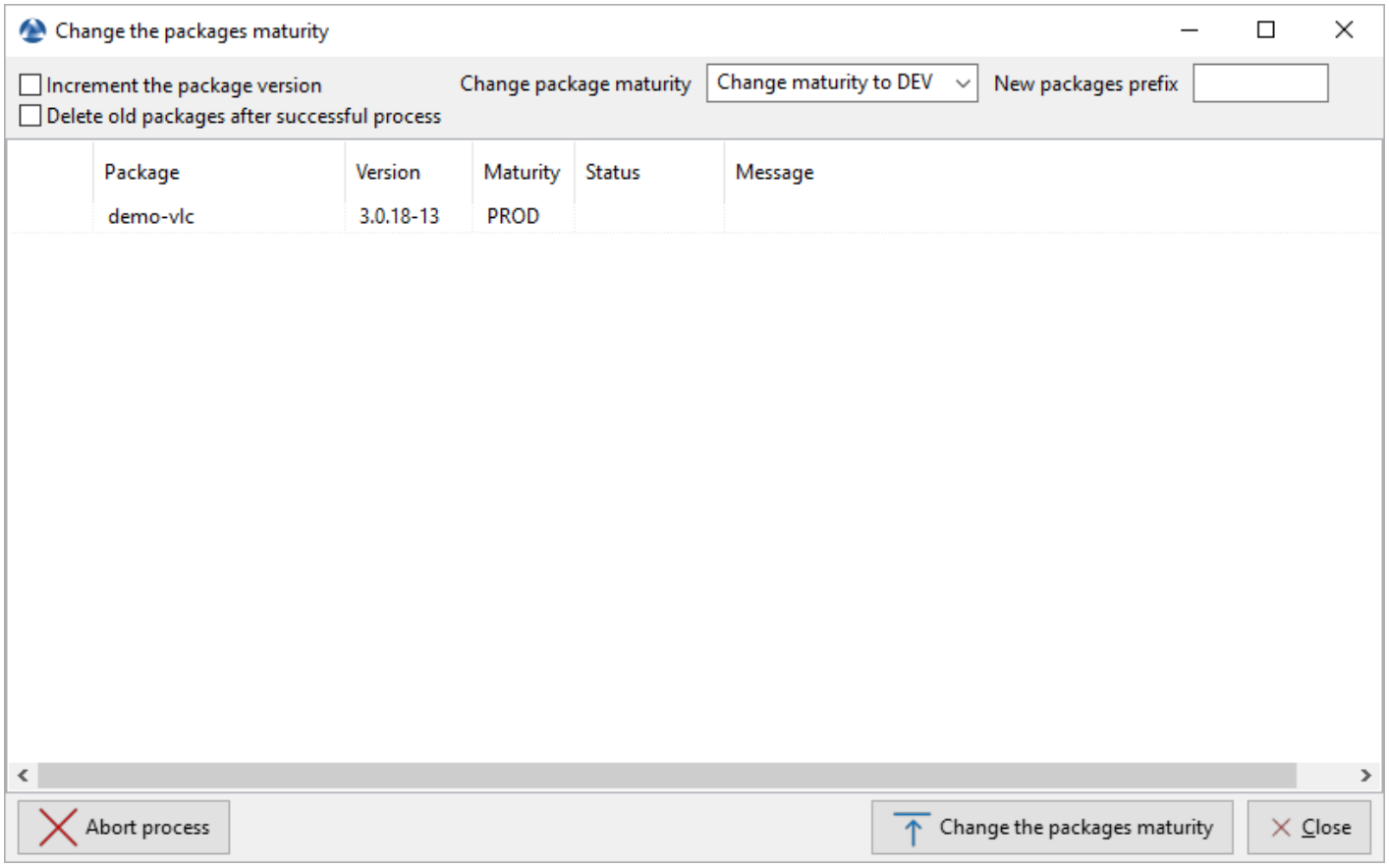


Fig. 20: Window for changing the maturity of a WAPT package

Table 3: Options for changing the maturity of WAPT packages

Label	Description
The *Increment the package version* checkbox	Increments the packaging version (version number after -).
The *Delete old packages after successful process* checkbox	Delete the old WAPT package after having changed the maturity.
The *Change package maturity* dropdown list	Configure the new maturity of the WAPT package.
The *New packages prefix* field	Configure a new prefix for the WAPT package. Prefix is case sensitive, we recommand to use lower case.

You can stop the process by pressing the *Abort process* button.

You can confirm the process by pressing the *Change the package maturity* button.

Once finished, the status switches to ✅.

Warning: You can change the maturity of a selection of WAPT packages at once

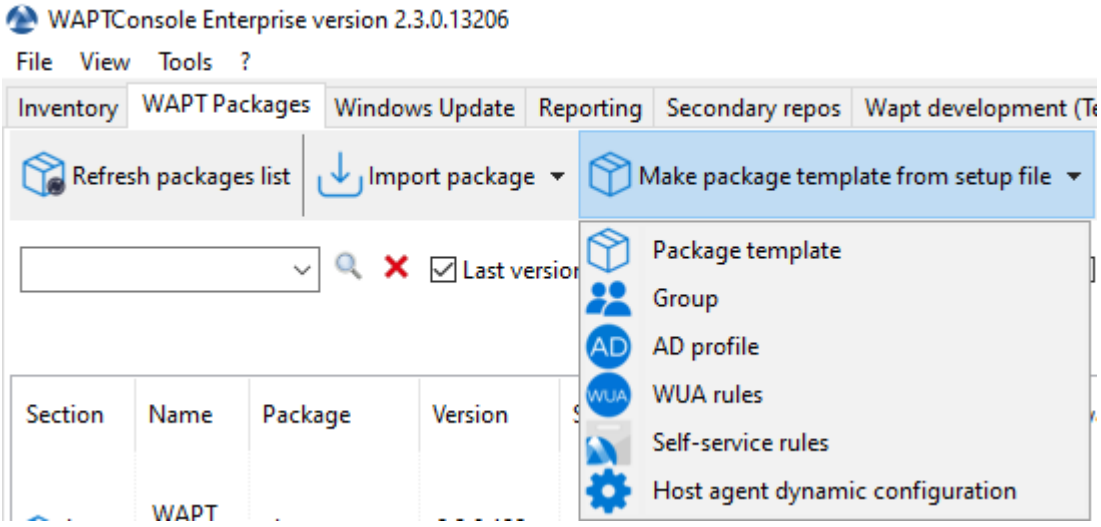
Changing the maturity of the WAPT package will change the hash of the file.

If the package is used in a GPO, like **`waptupgrade`**, the hash in the GPO will need to be changed.

19.5.7 Creating a group package

WAPT *group* packages allow to create a WAPT package containing other WAPT packages as dependencies.

To create a bundle of WAPT packages, go to the *WAPT Packages* tab in the WAPT console, then click on the *Make package template from setup file* button and finally choose the *Group* menu item.



- Change the name in the *Package name* field.
- Fill in the description.
- Add WAPT packages to the *group* package by dragging and dropping them or by Right-clicking on the WAPT package name and adding it to the bundle.

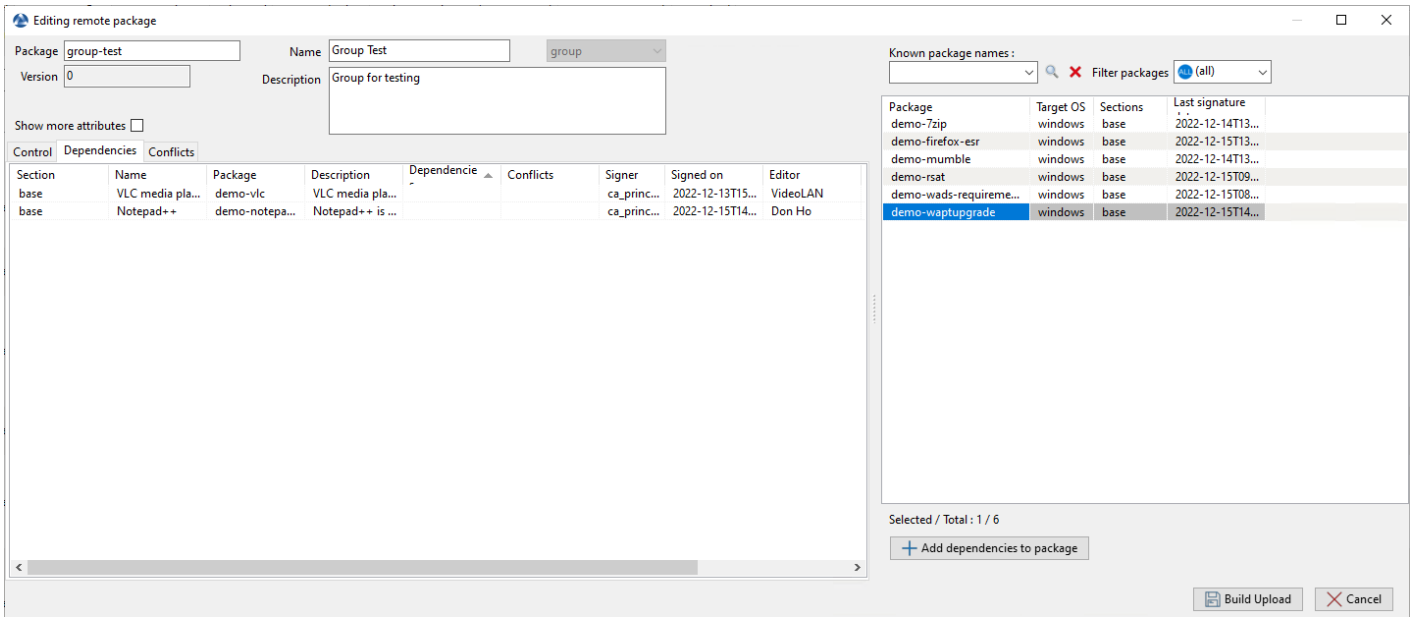


Fig. 21: Creating a group package

- Click on the *Save* button to save the bundle.

Hint: To uninstall a package, it is possible to add the package as a forbidden package to a *group* package. The forbidden WAPT package, if installed, will be removed before other WAPT packages are installed.

19.5.8 Removing a WAPT package

To delete a WAPT package from the repository, do a *Right-click → Remove from repository*.

You can select multiple WAPT packages to delete at once.

Removing the package this way will **only** delete this package and not anterior versions. To do so, you will have to uncheck *Last version only* and select all packages version to delete. You can use the option *Select obsolete packages* to help you then *Remove from repository* to go faster.

If you remove a package used by at least one host, you will have an alert window. If you check *Apply to the following* it will remove the dependencies on all other concerned hosts.

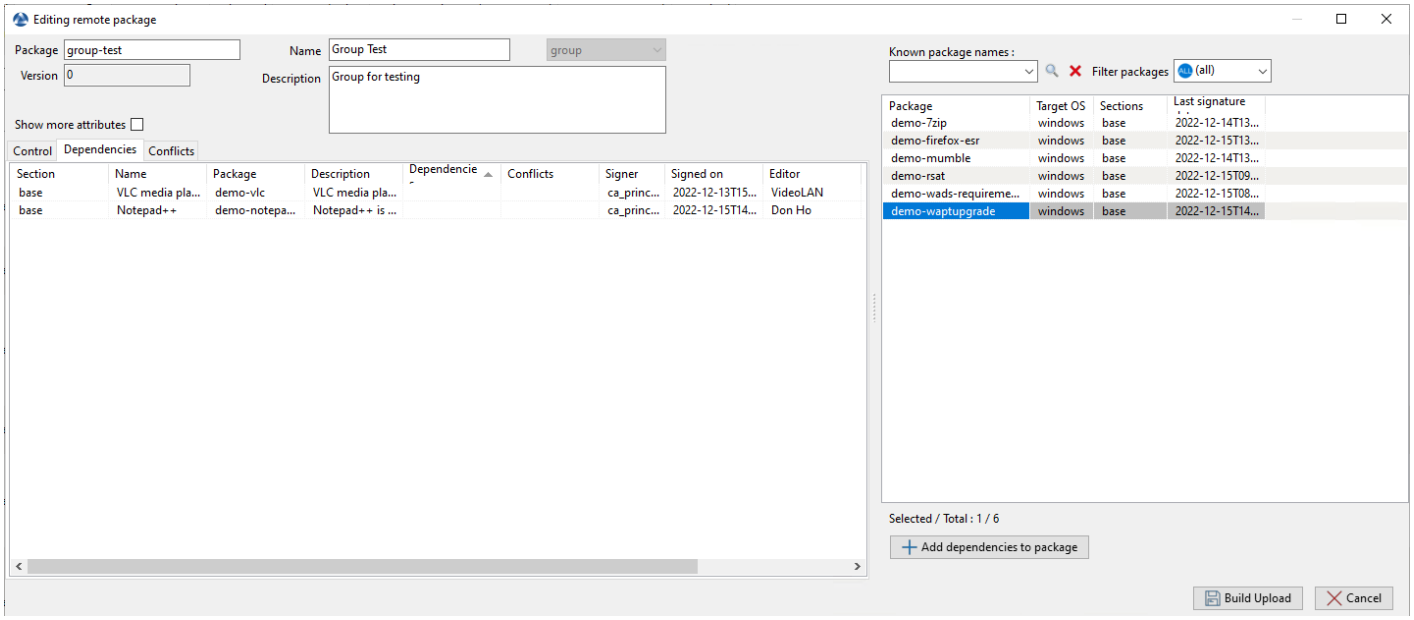
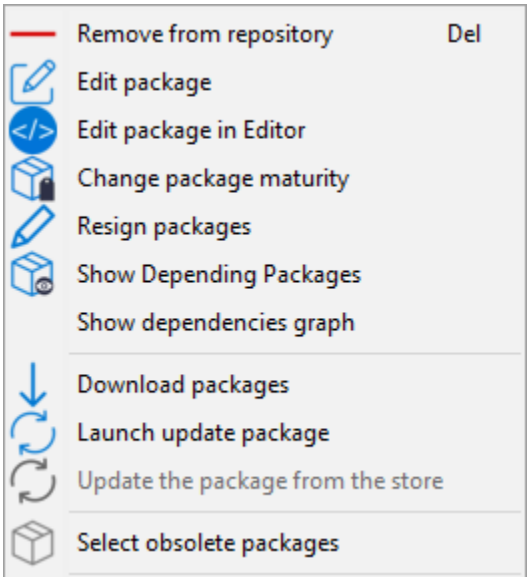


Fig. 22: Adding a forbidden package to a *group* package



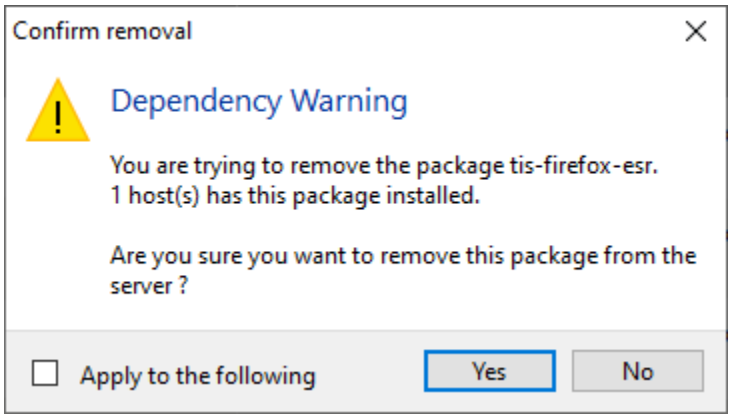
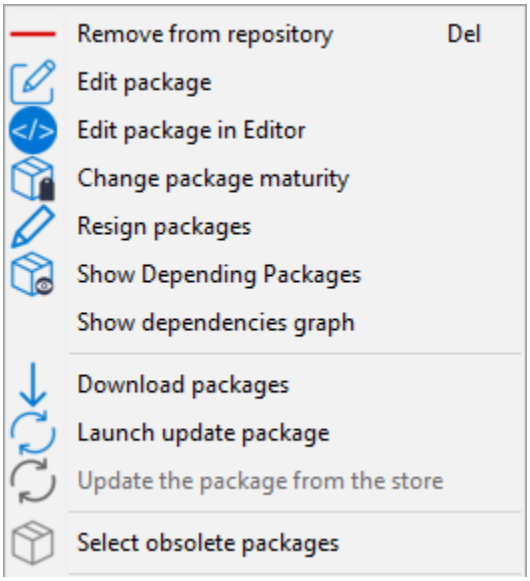


Fig. 23: Window alerting that a package is used by at least one host when attempting to delete a package from the local repository

19.5.9 Editing a WAPT package



To edit a WAPT package, do a *Right-click → Edit package*.

The WAPT package will be downloaded locally in **the base package development folder**, set in the WAPT Console settings.

If **`PyScripter`** is installed, **`PyScripter`** will automatically open the `control` and `setup.py` files.

Once edited you can upload the WAPT package *using the WAPT Console*.

19.5.10 Deploying WAPT packages from the WAPT Console

You can deploy WAPT packages on hosts using multiple methods:

- Directly by *adding a WAPT package to the selected host(s)*.
- By *adding a WAPT package to an Organizational Unit* of which the host is a member.
- By *adding a package to a host profile* that is applied to the host.
- By *adding the package to a group package* of which the host is a member.

19.6 Adding a WAPT package to a host

If you want to add WAPT packages directly on the host, you have to edit the *host* package.

To do so, there are 3 methods:

1. Double-click on the host.
2. Right-click on the host then *Edit host*.
3. Select a host and use the *Edit host* button.

Then, you just have to drag and drop wanted package(s) and confirm.

Fig. 24: Method for adding a WAPT package onto a host

Pressing *Save* does the same thing as doing an *update*.

Pressing *Save and apply* does the same thing as an *update* immediatelly followed by an *upgrade*.

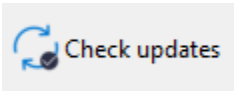
19.7 Adding or removing a WAPT package to several hosts

If you want to add or remove WAPT packages directly on several hosts, select the hosts then on one of them *Richt-click → Edit mulitple hosts packages*. You can add conflicting packages and remove conflicting packages from here too.

A new window will open, check the wanted package then click on *OK*.

After that, do an *update* immediately followed by an *upgrade* on the hosts.

19.8 Checking available updates for a host



This button will execute 2 actions:

1. give current state of the host to the WAPT Server
2. the WAPT Server displays whether the host **MUST** get updates

Every configuration modification require a *Check updates*.

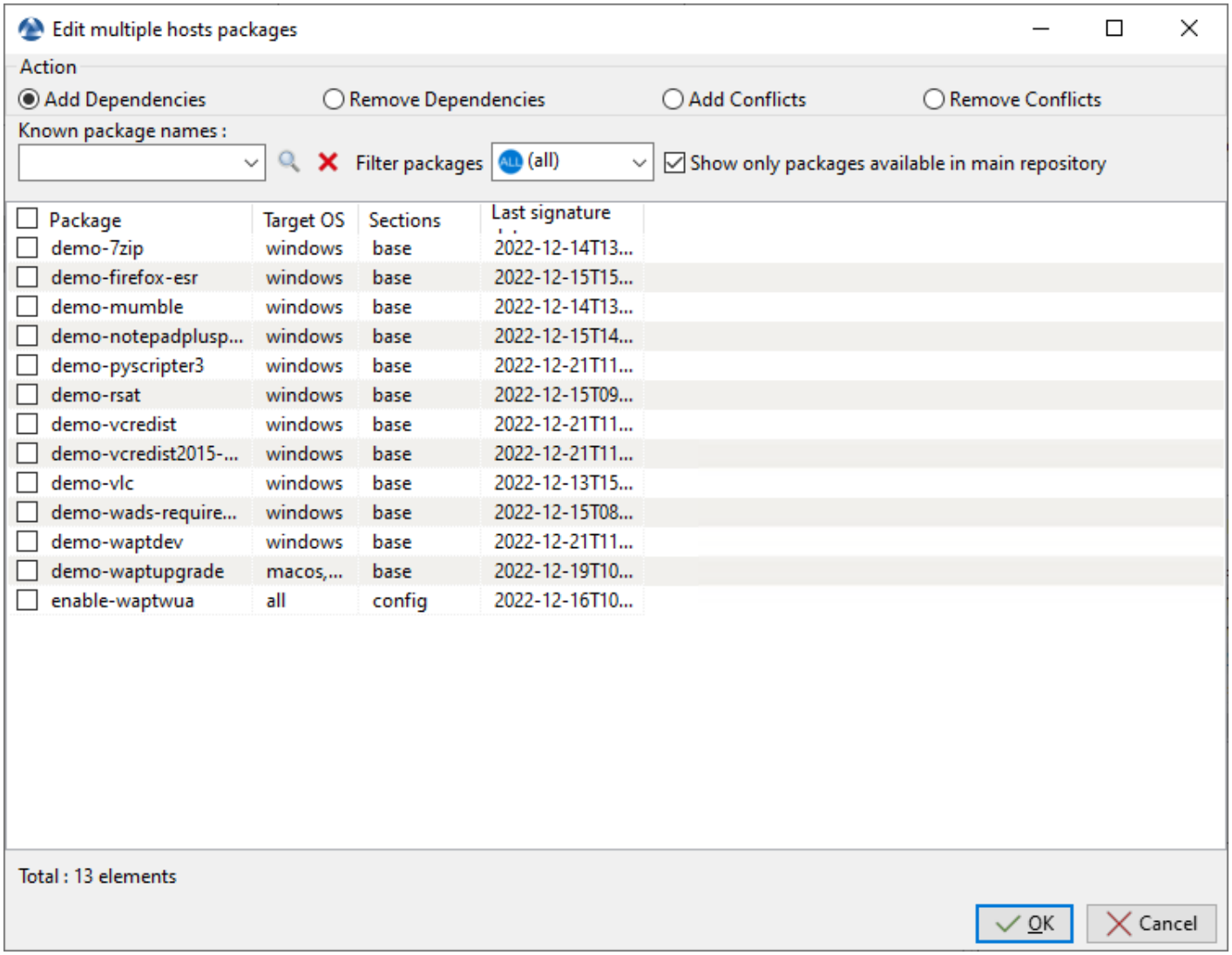
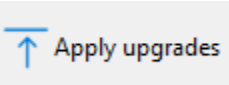


Fig. 25: Method for adding or removing a WAPT package onto several host

19.9 Applying updates on a host



This button allows to apply pending updates on the selected host(s).

Warning: Use with caution, it will force close the software titles that the users are currently running.

One can use instead *Apply upgrade for not running applications* to prevent the users losing their unsaved work.

19.10 Performing a global search on all hosts

Performing global searches on all the criteria presented above is possible.

Choose the filters to check or uncheck.

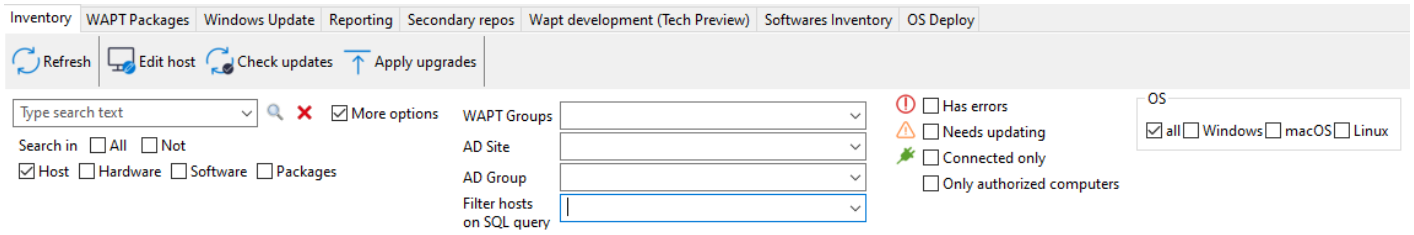


Fig. 26: Advanced search functionalities in the WAPT Console

Table 4: Choice of filters

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.
Packages	Lists WAPT packages installed on the selected hosts.
Has errors	Searches only for hosts for which a task has not finished correctly.
Needs updating	Searches only for hosts needing upgrades.
Connected only	Searches only for connected hosts
Only authorized computers	Searches only for hosts authorized by certificate of current WAPT Console user.
WAPT Group	Filters hosts based on their membership / dependency to a WAPT *group* package.
AD Site	Filter hosts based on their membership / dependency to a Site within Active Directory.
AD Group	Filters hosts based on their membership / dependency to a Active Directory group.
OS	Filters hosts based on their Operating Sytem.

Hint: Filters work with regular expression.

19.11 Creating a configuration package

WAPT **configuration** packages allow to create several WAPT configurations without having to create several WAPT Agents.

To create a configuration package, go to the *WAPT Packages* tab in the WAPT console, then click on the *Make package template from setup file* button and finally choose the *Host agent dynamic configuration* menu item.

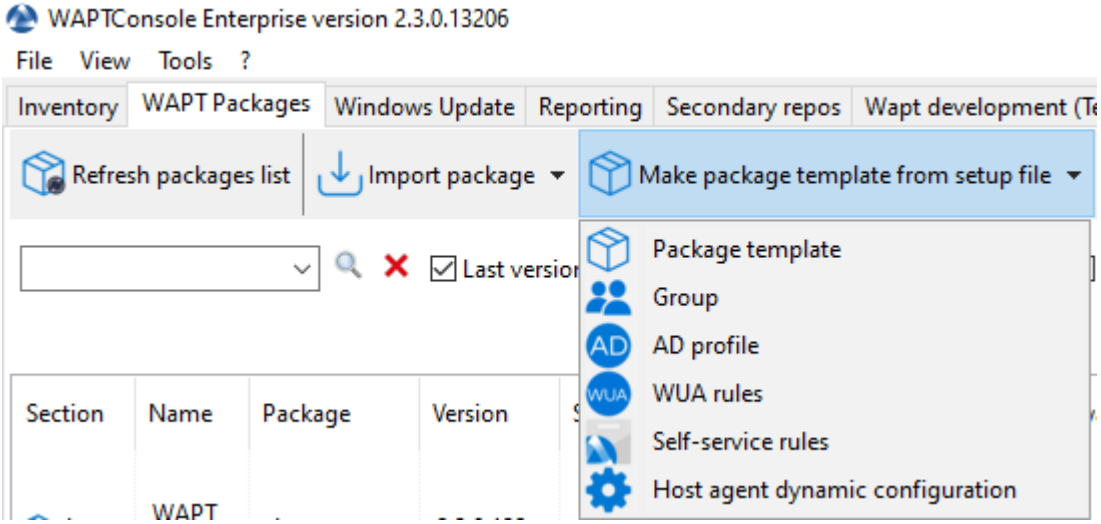


Fig. 27: Package group grid

A new window will open in order to create your new **configuration** package.

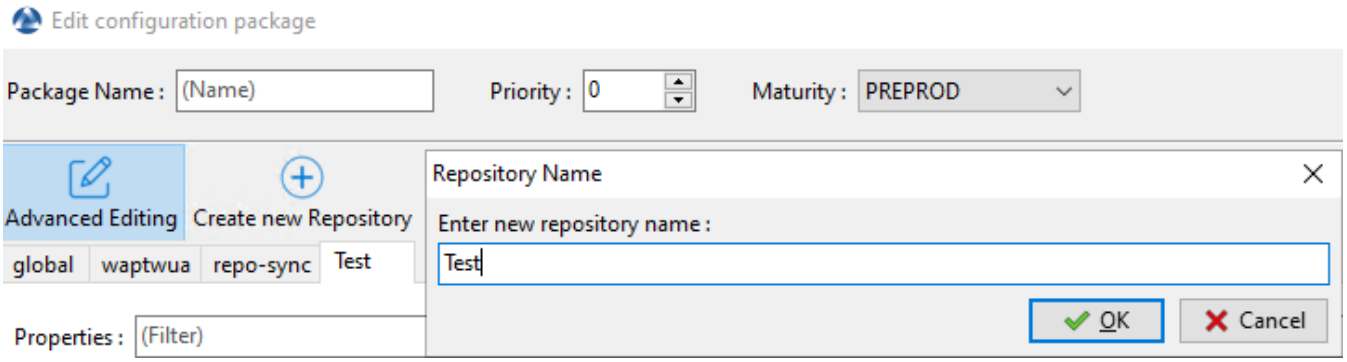
Change the name in the *Package name* field. Now there are several things that can be done.

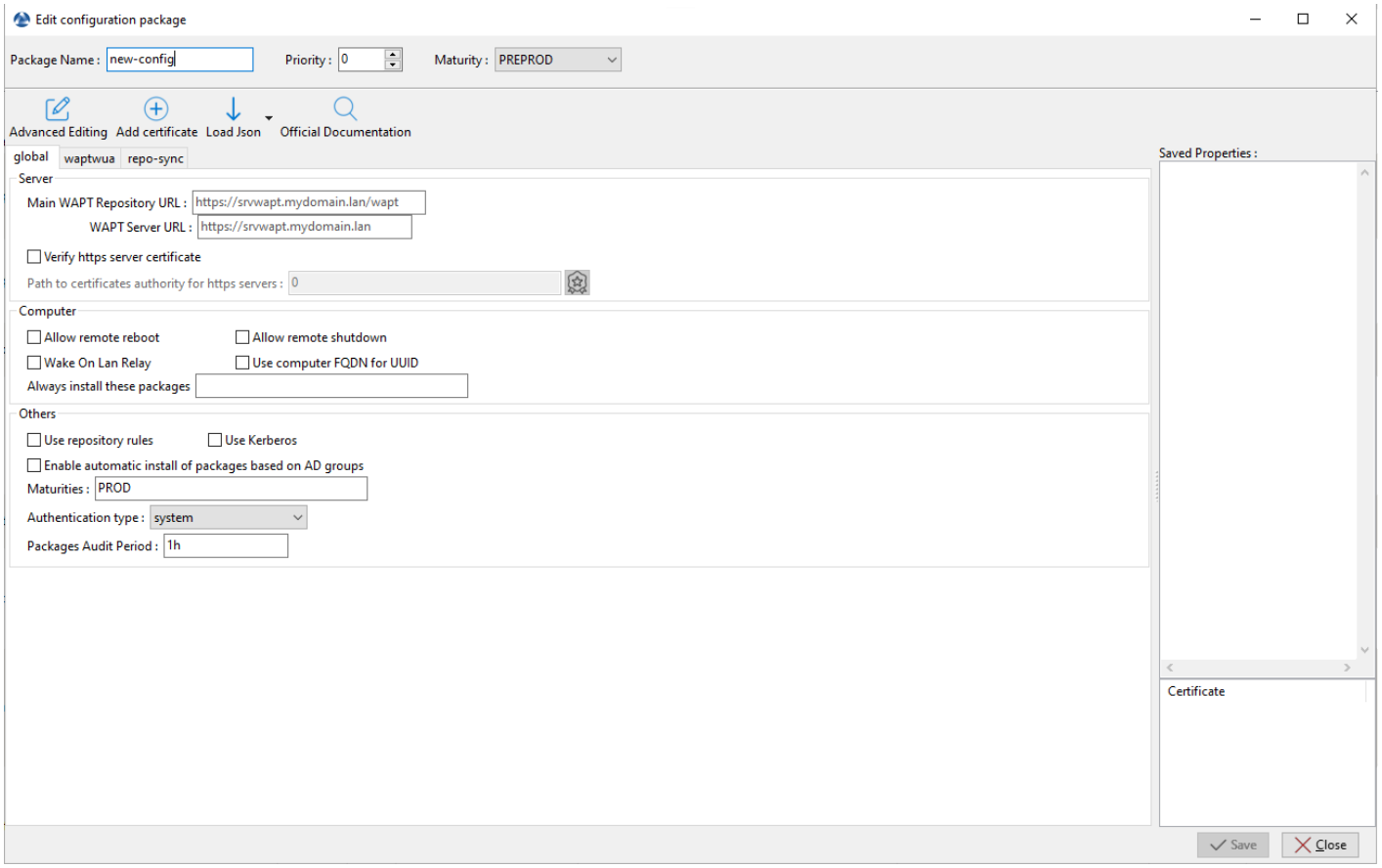
19.11.1 The configuration toolbar

The *Advanced Editing* option will switch the simplified view into a complete list of options. Please see this section for *advanced waptagent options*.

You can use a filter to search the name of an attribute.

- The *Create new Repository* will allow you to create a new WAPT repository if you have another repository than yours. If you do so, a new tab named by your repository will appear in the config package.




Edit configuration package
Package Name : new-config
Priority : 0
Maturity : PREPROD
Advanced Editing
Add certificate
Load Json
Official Documentation
global
waptwua
repo-sync
Server
Main WAPT Repository URL : https://srvwapt.mydomain.lan/wapt
WAPT Server URL : https://srvwapt.mydomain.lan
Verify https server certificate
Path to certificates authority for https servers : 0
Computer
Allow remote reboot
Allow remote shutdown
Wake On Lan Relay
Use computer FQDN for UUID
Always install these packages
Others
Use repository rules
Use Kerberos
Enable automatic install of packages based on AD groups
Maturities : PROD
Authentication type : system
Packages Audit Period : 1h
Saved Properties :
Certificate
Save
Close

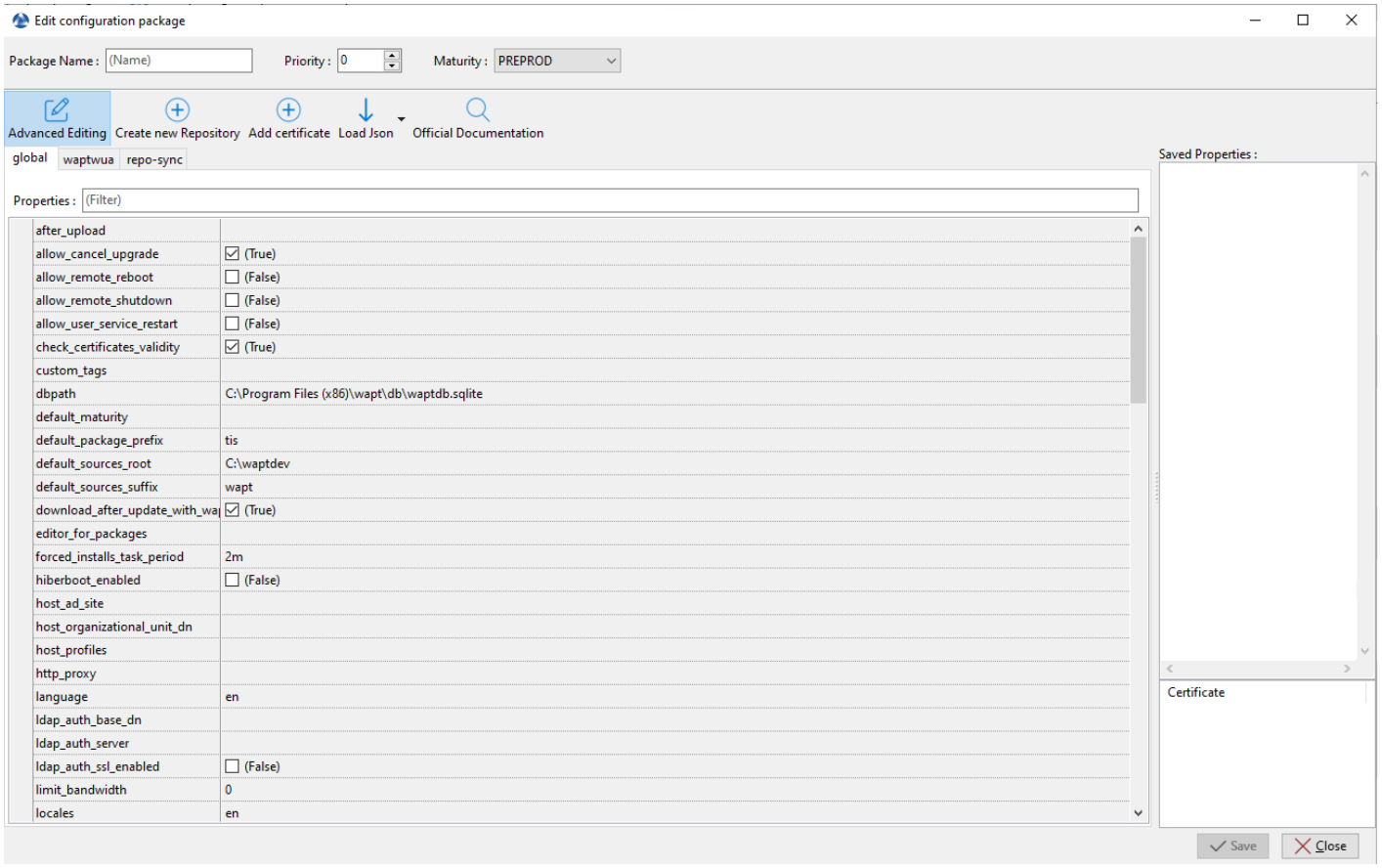


Fig. 28: Advanced WAPT package configuration form

- The *Add certificate* button will open a new window that will allow to select a certificate. The certificate will be included in the bundle of certificates authorized to perform actions on hosts.

Fig. 29: Adding a certificate in a WAPT configuration package

You can remove the certificate with *Right-click → Remove Certificate*.

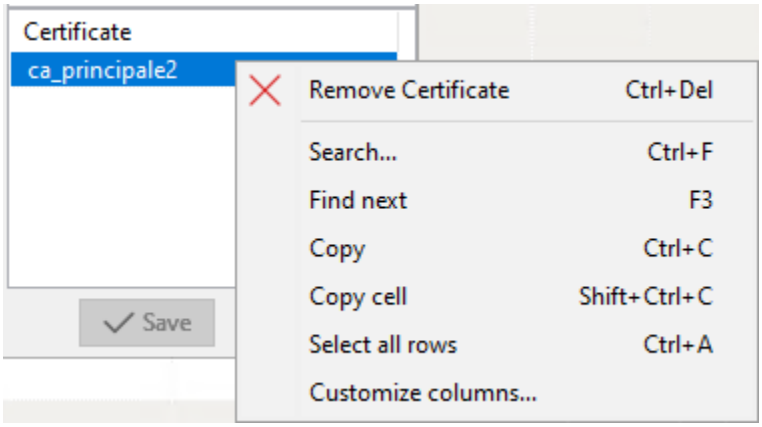


Fig. 30: Removing a certificate in a WAPT configuration package

- The *Load Json* button will import a `json` file which can be a configuration file.

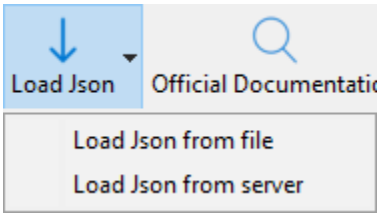


Fig. 31: Loading a json file from a file or from the WAPT server

If you choose to download a `json` configuration file from the WAPT Server, you can choose amongst existing configurations.

- The *Official documentation* button will open a link to the Tranquil IT documentation with all `wapt-get.ini` options.

19.11.2 The configuration tabs

By default, there are 3 tabs: *global*, *waptwua* and *repo-sync*.

- The *global* tab will have the same options when you create a *WAPT Agent*. The right pane summarizes the options that will be set.
- The *waptwua* tab will help you choose the options to use with *WAPT Windows Updates*.
- The *repo-sync* tab will help configure the remote host to become a *remote repository*.

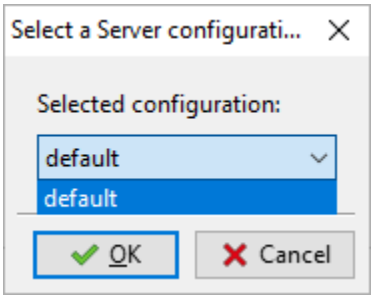


Fig. 32: Selecting a json file from the WAPT server

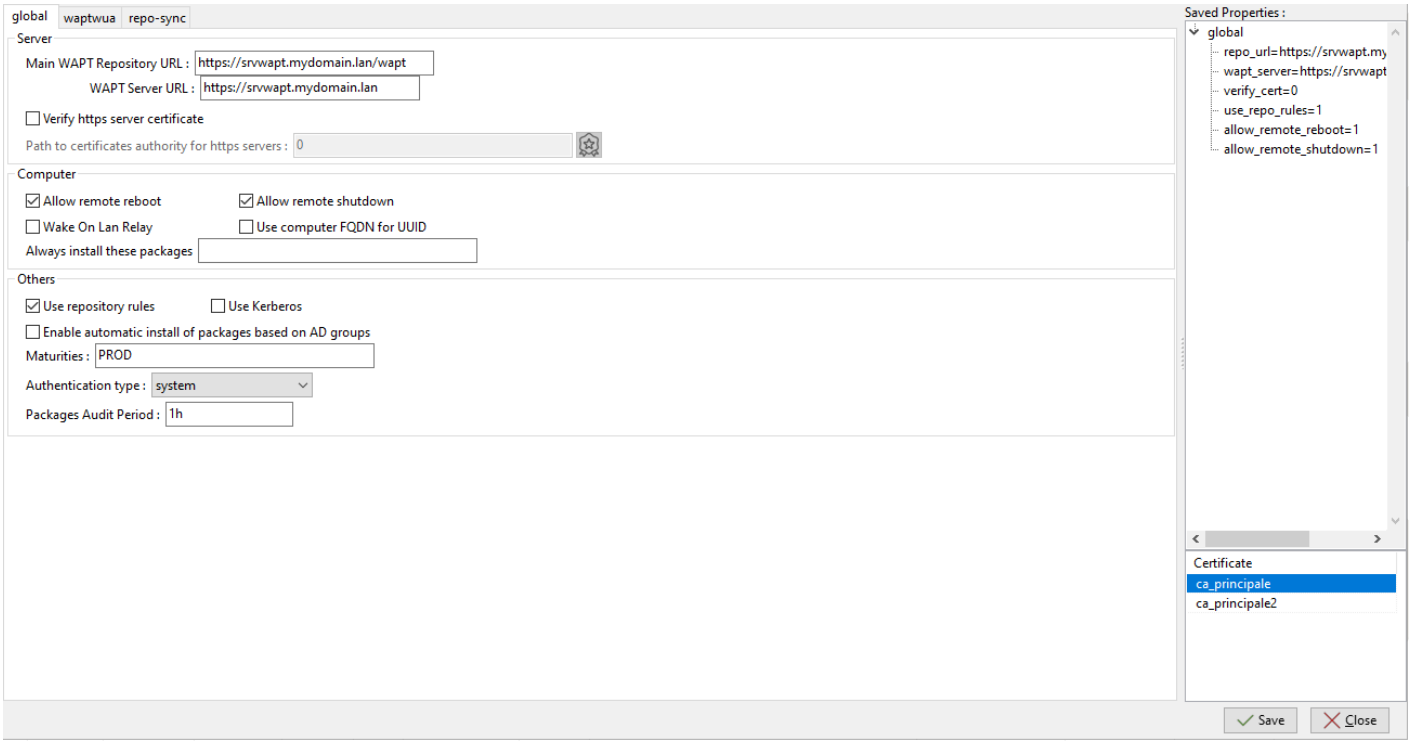


Fig. 33: Global tab for configuring a WAPT package

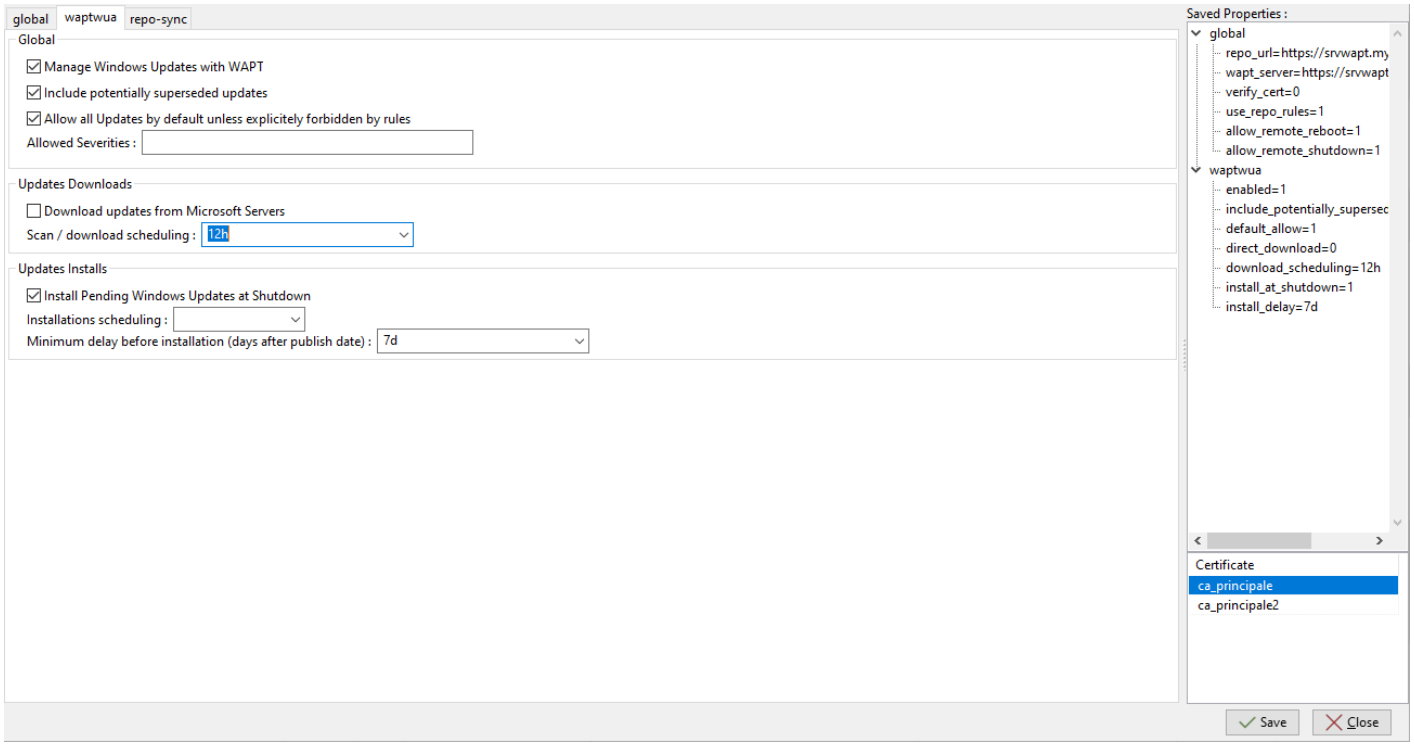


Fig. 34: Tab for enabling a remote host to use WAPT Windows Updates

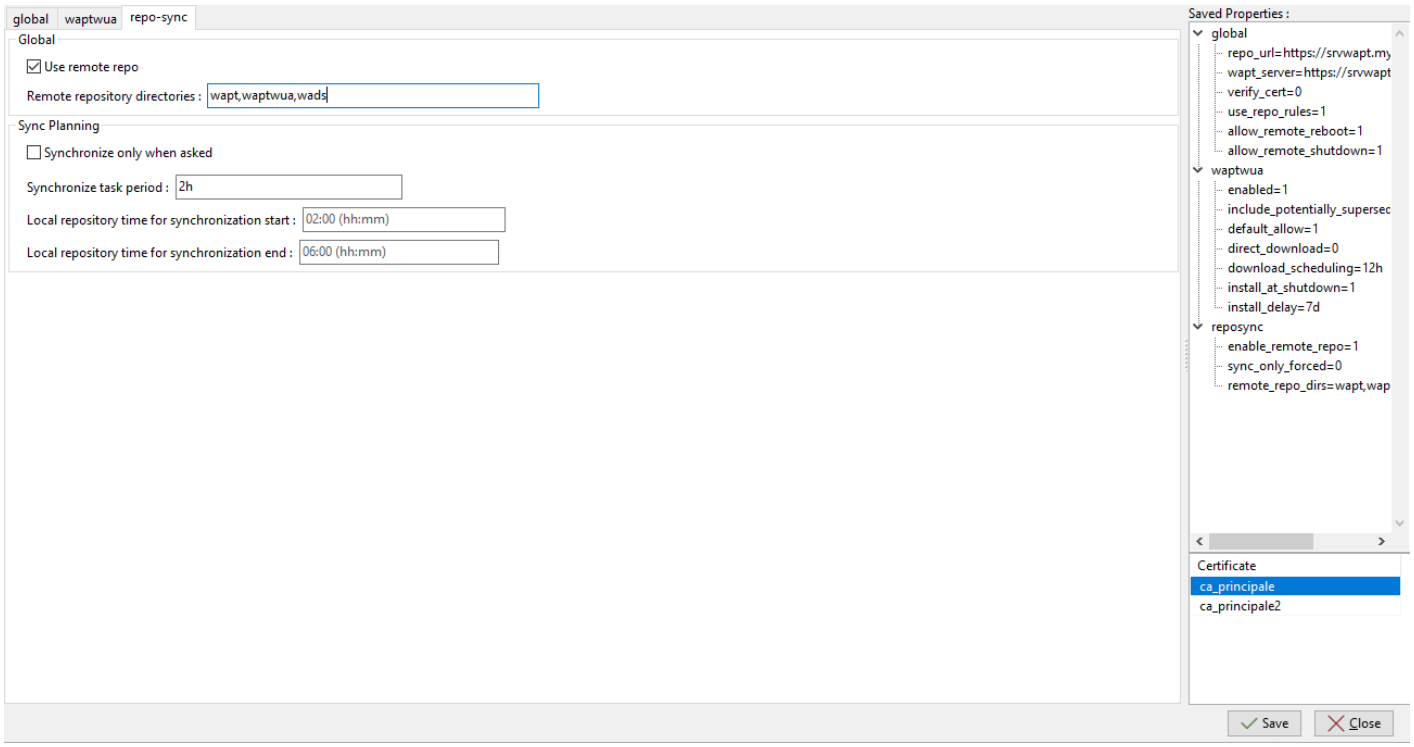


Fig. 35: Tab for configuring a WAPT package to make a WAPT Agent become a remote repository

19.11.3 Removing a configuration section

To delete an option or an entire section, use *Right-click → Delete selected property*.

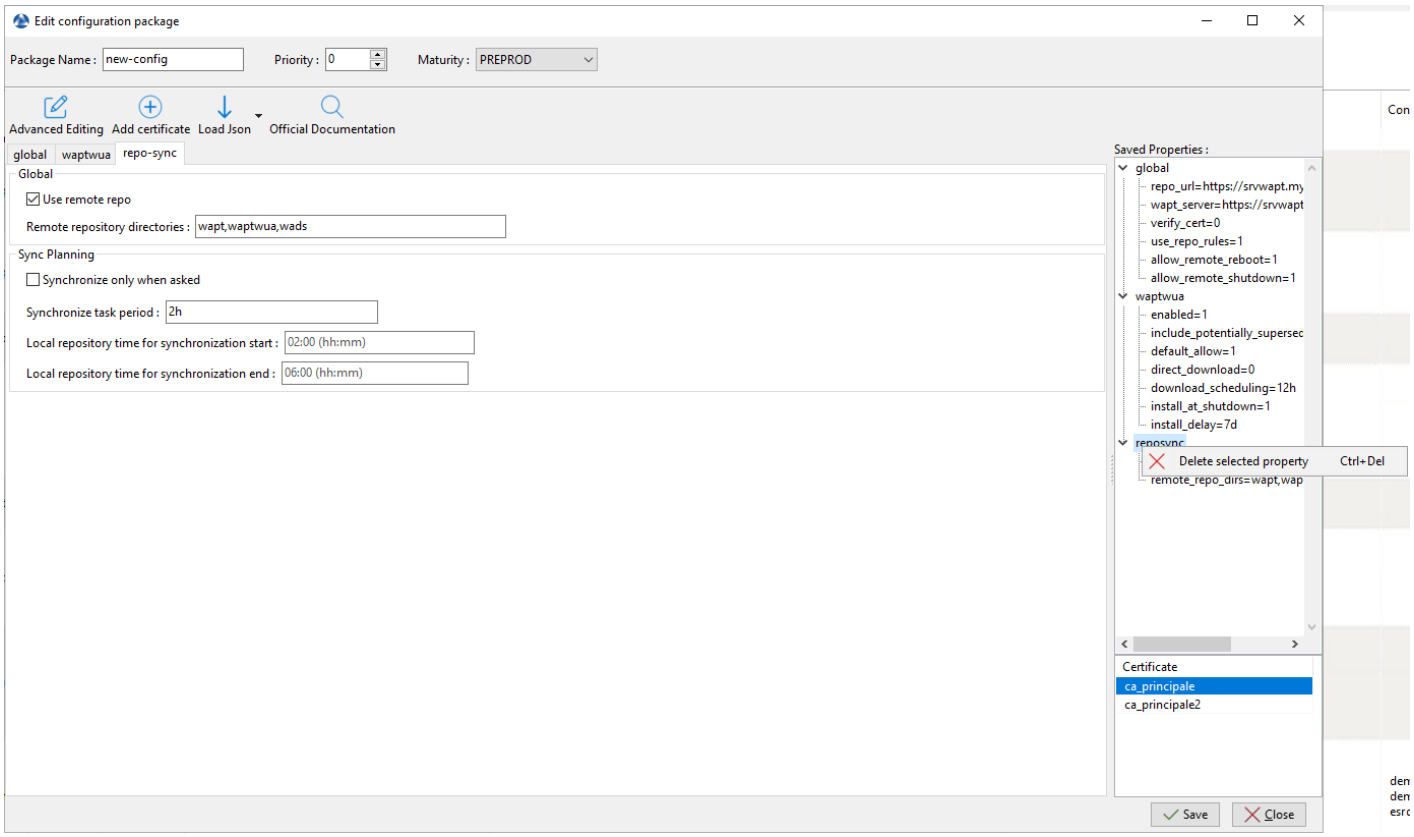


Fig. 36: Deleting a property, an option or a section

19.11.4 Saving the configuration package and applying the new configuration to a remote host

Click on the *Save* button to save the new configuration bundle. The configuration file will be uploaded onto the WAPT Server in `.json` file format. Select the hosts and apply the configuration package like any WAPT package.

On the host, the configuration package will be located in the `conf.d` WAPT installation directory in a `json` format which can look like this:

19.12 Removing the WAPT agent from the WAPT Console

If you want to delete a WAPT Agent from the console, right-click on the target agent then click-right and select *Remove Host* (see above). You will have the possiblity to check *And delete host configuration (Package)*, if you do so, it will delete the targeted *host pakage* and delete all information about the selected computer.

Note: Deleting a host from the WAPT Console **does not** uninstall the WAPT Agent from the computer. Please refer to this *documentation to uninstall the WAPT Agent*. If the WAPT Agent is not properly removed, the computer will register again with the WAPT

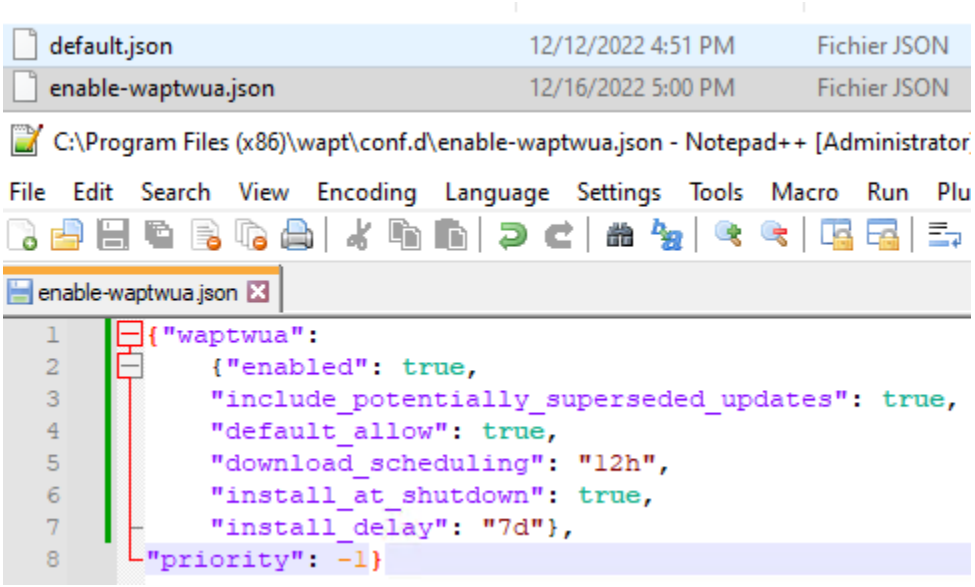
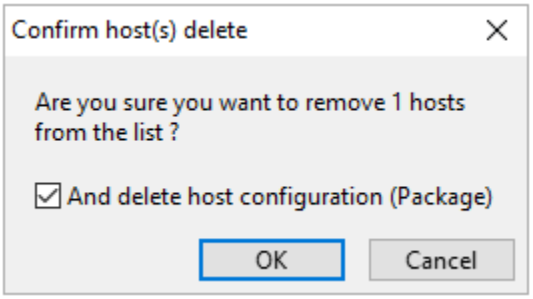


Fig. 37: Example of a configuration file done by a WAPT configuration package



Server.

CHAPTER

TWENTY

USING THE WAPT CONSOLE ADVANCED FEATURES

This page details the advanced use of the WAPT Console.

20.1 Using Organizational Unit packages in WAPT

20.1.1 Working principle

WAPT Enterprise offers Organizational Unit package functionality.

unit packages automate software and configuration installations based on the Active Directory tree. It is a very powerful feature when used properly.

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 are removed.

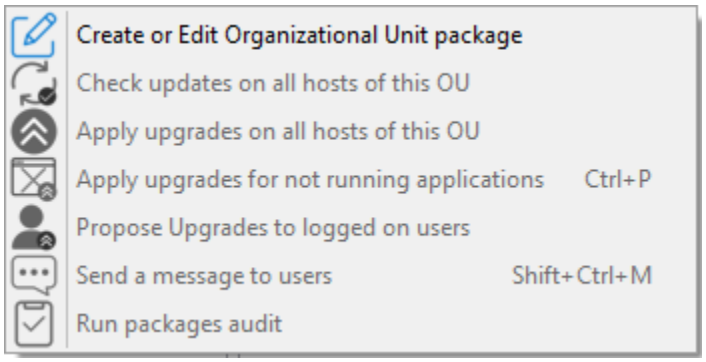
The WAPT Agent is aware of its position in the Active Directory tree structure, therefore it knows the hierarchy of Organizational Units that concerns it, for example:

```
DC=ad,DC=mydomain,DC=lan
OU=Paris,DC=ad,DC=mydomain,DC=lan
OU=computers,OU=Paris,DC=ad,DC=mydomain,DC=lan
OU=service1,OU=computers,OU=Paris,DC=ad,DC=mydomain,DC=lan
```

If a *unit* package is defined on each Organisational Unit level, the WAPT Agent will automatically download WAPT packages and configurations that are attached to each level. Using inheritance, WAPT will apply WAPT packages and dependencies that are attached to each Organizational Unit.

20.1.2 Creating Organizational Unit packages

You can create *unit* packages by *Right-clicking on an OU → Create or Edit Organizational Unit package*.



A window opens and you are prompted to choose which packages to include in the *unit* bundle.

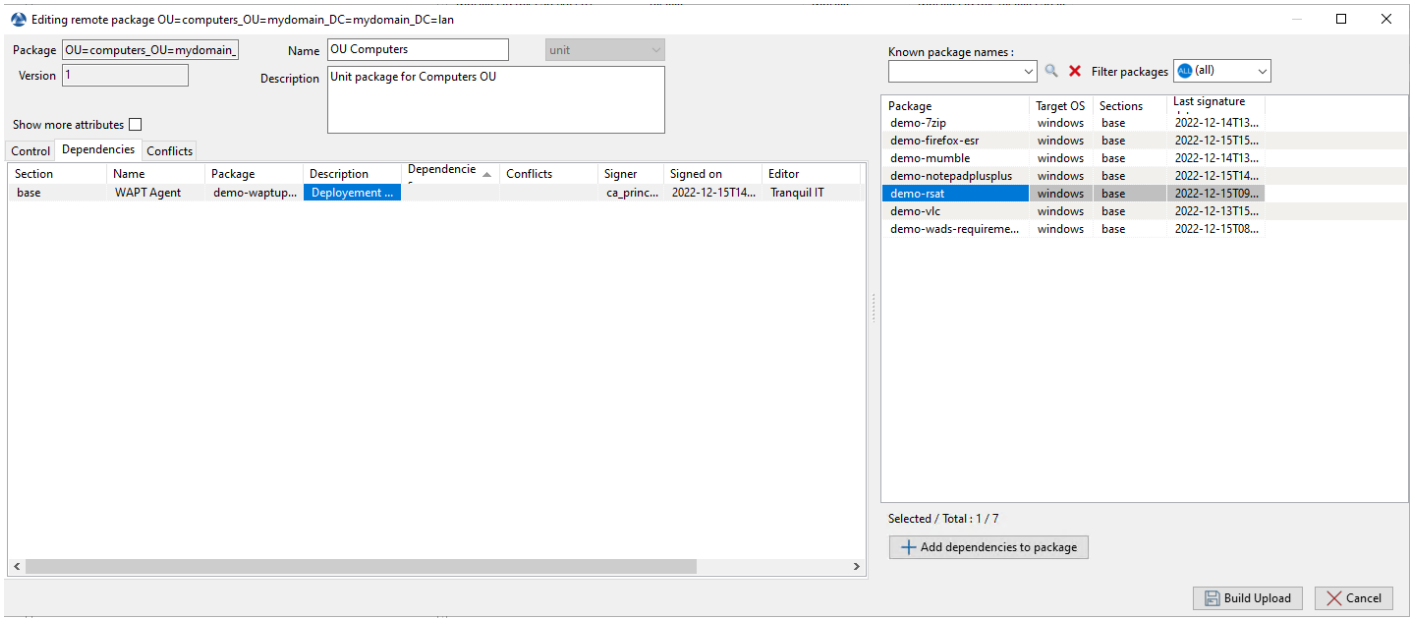
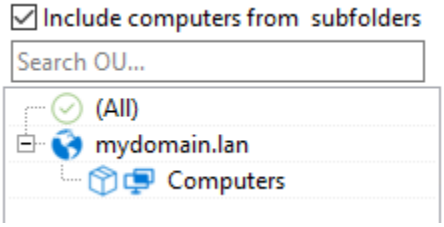


Fig. 1: Adding WAPT packages to a unit bundle

Save the WAPT package and it will be deployed to all hosts belonging to the selected OU.

When you have a **unit** bundle, you will see a cube before the OU name in the WAPT Console.



20.1.3 Actions available with Organizational Units

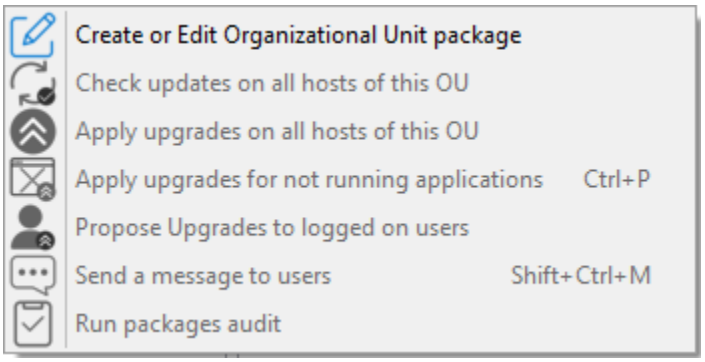
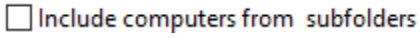


Table 1: Menu items for creating or editing Organizational Unit package

Menu item	Description
The *Create or Edit Organizational Unit package* menu item	*Visit this documentation for more details on creating or editing OU packages*.
The *Check updates on all hosts of this OU* menu item	Allows to upload the current state of the host to the WAPT Server and force the WAPT Server to display whether the hosts in the selected OU have pending updates.
The *Apply upgrades on all hosts of the OU* menu item	Allows to apply waiting WAPT *updates* and *upgrades* on the all hosts in the OU.

Hint: 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.

20.1.4 Faking Organizational Units for WORKGROUP hosts

It can happen that some specific hosts cannot be joined to an Active Directory domain.

Therefore, these hosts do not show up in the Active Directory Organizational Units in the WAPT Console.

To make all hosts show up in the WAPT Console under the right Organizational Unit, whether they are joined to an AD domain or not, WAPT allows to specify a *fake* Organizational Unit in the WAPT Agent configuration file.

The benefits of this very useful trick are:

- You can manage these hosts with WAPT as if they where joined to the Active Directory.
- Out-of-domain and workgroup hosts are now showing up in the Active Directory tree view in the WAPT Console.
- *Unit* packages become usable on these hosts.

To setup a *fake* Organizational Unit on hosts, create an *empty WAPT package*, then use the following code:

```
# -*- coding: utf-8 -*-
from setuphelpers import *
```

(continues on next page)

(continued from previous page)

```
uninstallkey = []

def install():

  print('Setting Fake Organizational Unit')
  fake_ou = "OU=REAL_AD_SUB_OU,OU=REAL_AD_OU,DC=MYDOMAIN,DC=LAN"
  inifile_writestring(WAPT.config_filename,'global','host_organizational_unit_dn',fake_ou)

  print('Reload WAPT configuration')
  WAPT.reload_config_if_updated()

def update_package():
  pass
```

The `host_organizational_unit_dn` will be like below in `wapt-get.ini`:

```
[global]
host_organizational_unit_dn=OU=REAL_AD_SUB_OU,OU=REAL_AD_OU,DC=MYDOMAIN,DC=LAN
```

Note:

- Stick to a specific case with your `host_organizational_unit_dn` (do not mix "dc"s and "DC"s, "ou"s and "OU"s ...).
- Follow the letter case used in the *DN/computer_ad_dn* fields in the hosts inventory grid.

20.2 Using profile bundles in WAPT

20.2.1 Working principle

WAPT Enterprise offers an Active Directory *profile* bundle functionality.

The *profile* bundle automates the installation of WAPT packages and configuration packages on hosts based on their membership to Active Directory Computer Security Groups.

The WAPT Agent will report to the WAPT Server the Active Directory groups to which the host belongs.

If a *profile* package has the same name as an Active Directory group, then the WAPT agent will install automatically the *profile* package for the Active Directory group of which the host is a member.

If the host is no longer a member of its Active Directory group, then the matching *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 WAPT upgrades.

Note: For performance reasons, this feature is enabled only if the `use_ad_groups` option is enabled in the `wapt-get.ini` configuration file of the WAPT Agent.

Important: **The Active Directory Computers security groups and sub-groups contain Computers, not Users**.

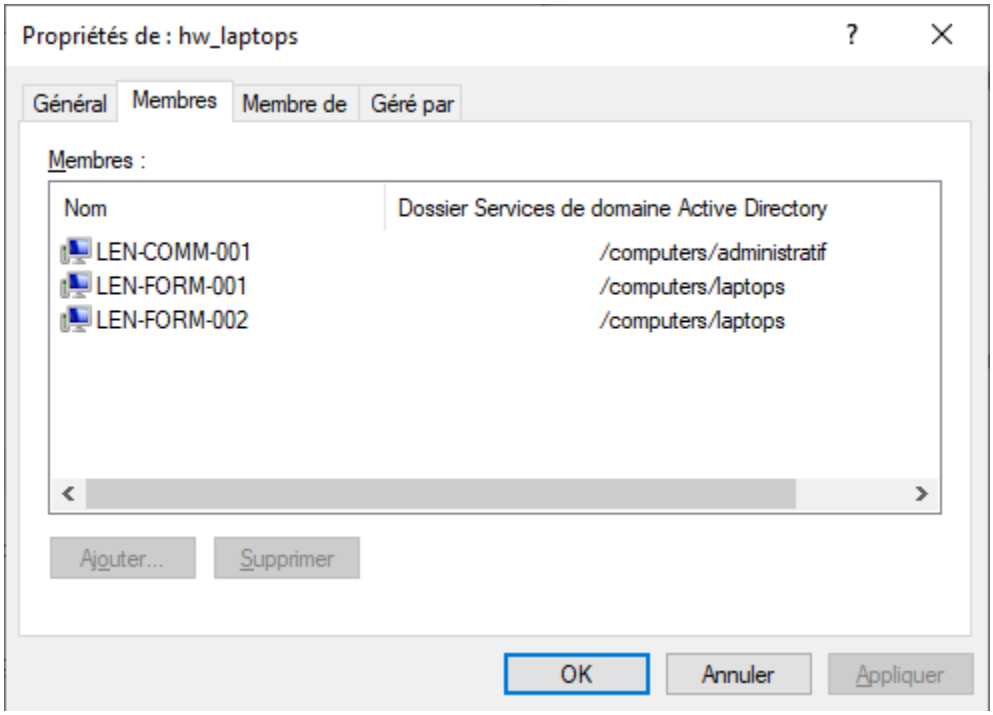


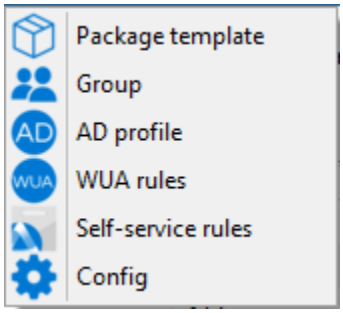
Fig. 2: Window showing the Computers group in Active Directory

Warning: Automatically installing software and configurations based on user and user group membership is not implemented with WAPT and such implementation is not desirable. The use case of installing software based on user profile is better served with the differentiated *self-service* feature that is also available with WAPT Enterprise.

The name of the group **MUST** be lower case in Active Directory and in the WAPT Console.

20.2.2 Creating WAPT *profile* bundles in the WAPT Console

You can create *profile* bundle WAPT packages by clicking on *Make package template from setup file → AD profile*.



Important: Requirements:

- The *profile* AD group name and the *profile* package **MUST** be all lower case.

Example:

- AD Security group: **hw_laptops**;
- WAPT profile bundle: **hw_laptops**.

A window opens and you are prompted to choose which WAPT packages are to be included in the newly created **profile** bundle.

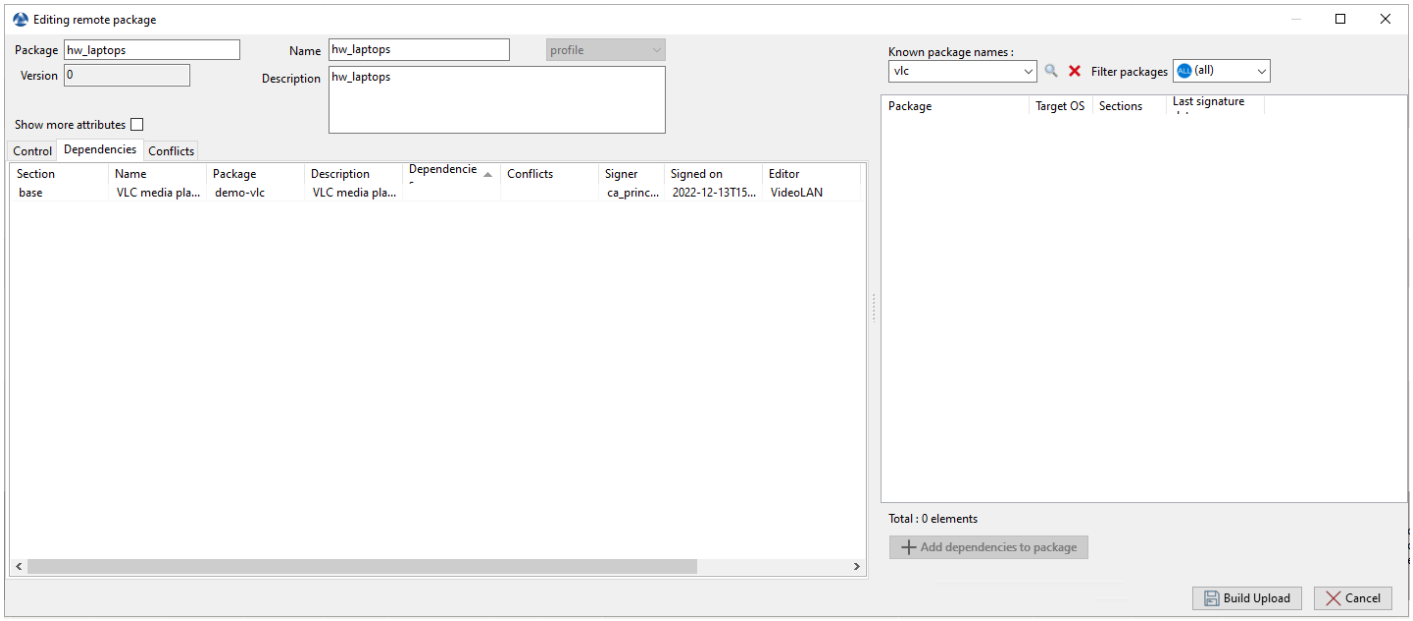


Fig. 3: Adding WAPT packages to a *profile* bundle in the WAPT Console

Save the *profile* bundle and it will be uploaded to the WAPT Server.

20.3 Adding plugins in the WAPT Console

To add custom plugins, go to the *Tools* → *Preference* → *Plugins* Tab.

Click *Add* to add a plugin, then edit the corresponding columns.

Column	Description
Name	Name that will appear in the menu.
Executable	Path of the executable that will be executed.
Arguments	Arguments passed to the executable. All the parameters that are diplayed in the grid can be used, like {ip}, {uuid} or {computer_fqdn}. To get the parameter name, you may right-click on the colum header, and the name will be displayed in paranthesis beside the column name.

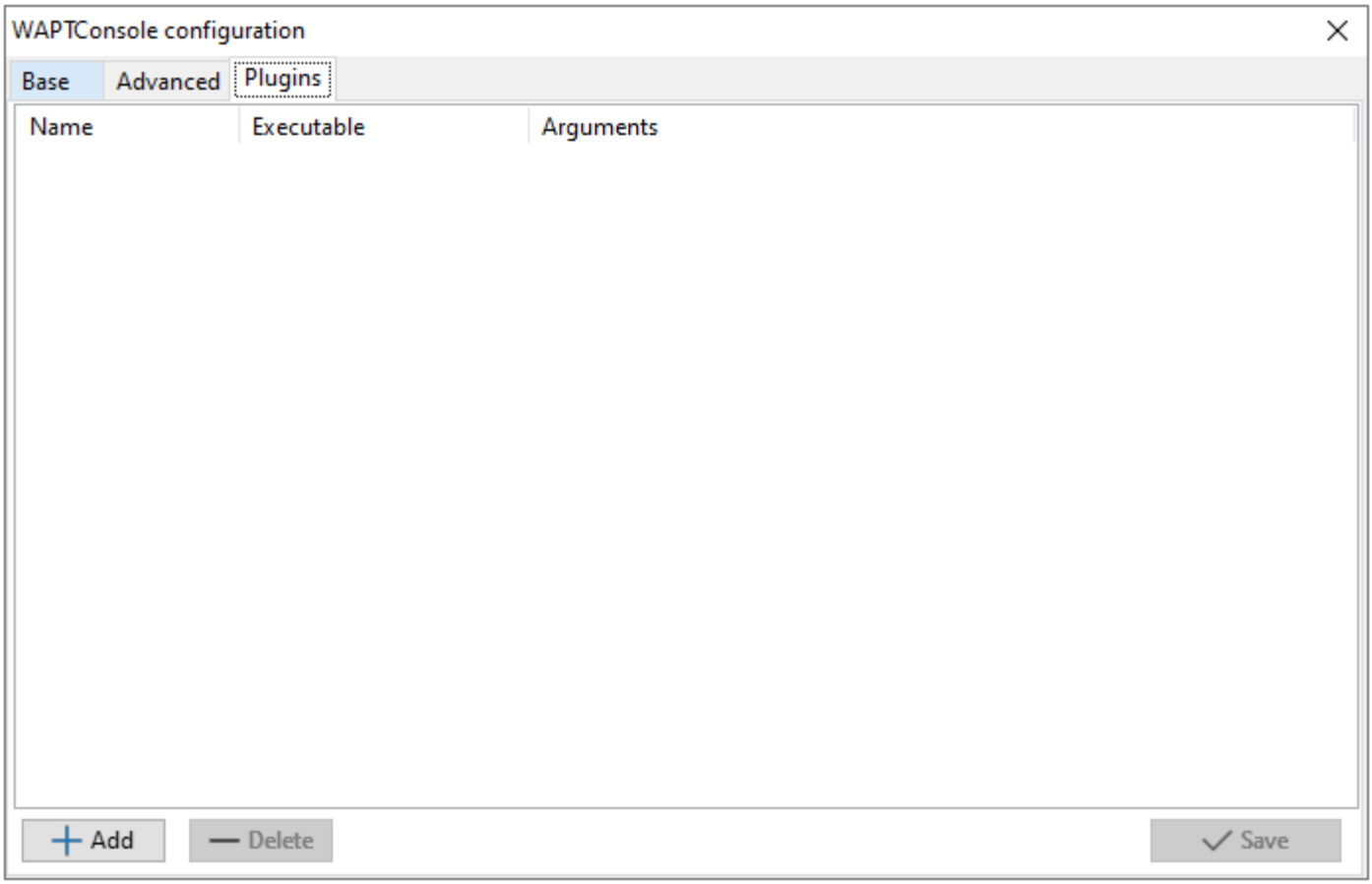


Fig. 4: Creating a custom plugin in the WAPT Console

Plugins will then appear in the menu:

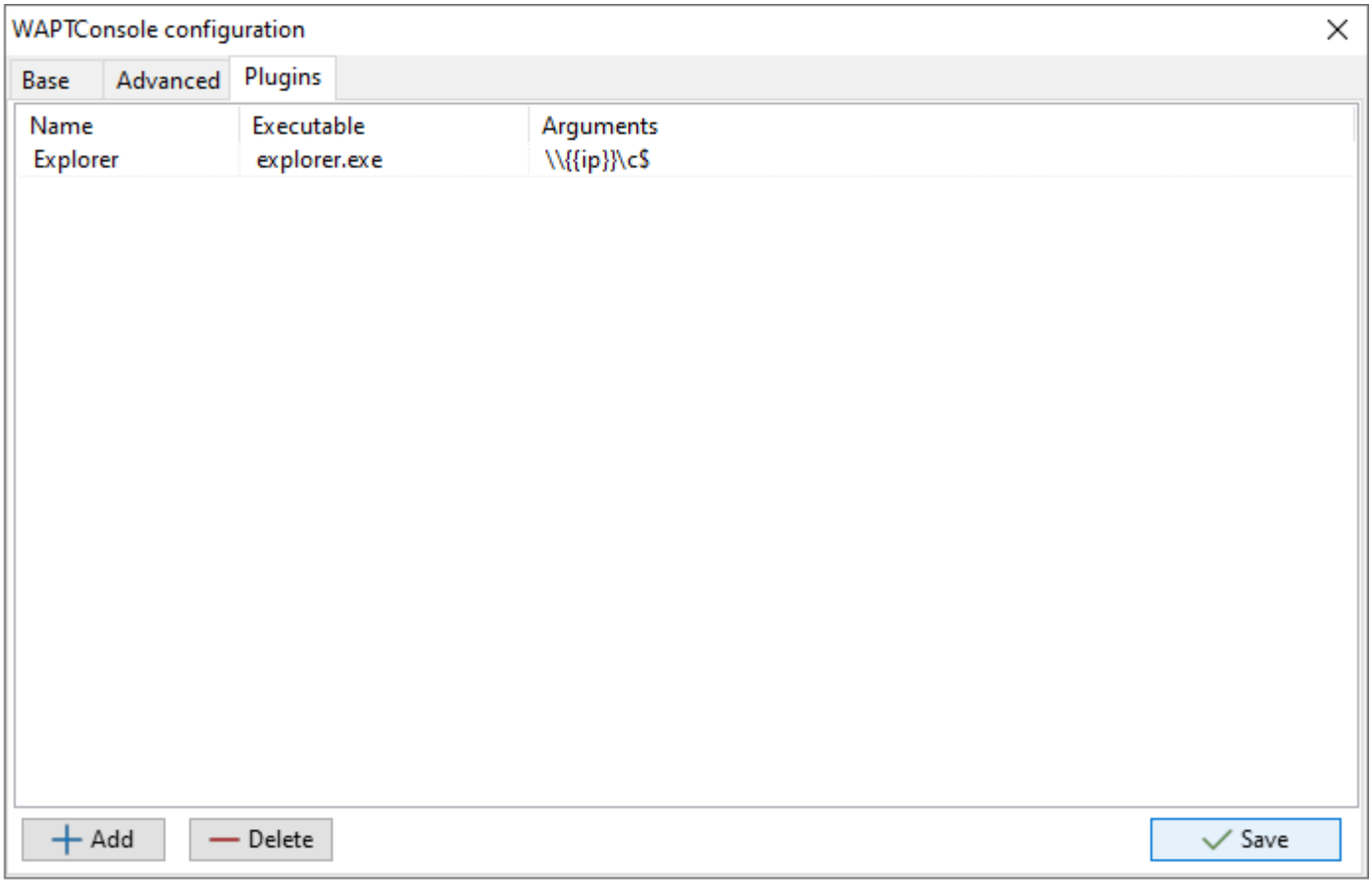


Fig. 5: Creating a custom plugin in the WAPT Console

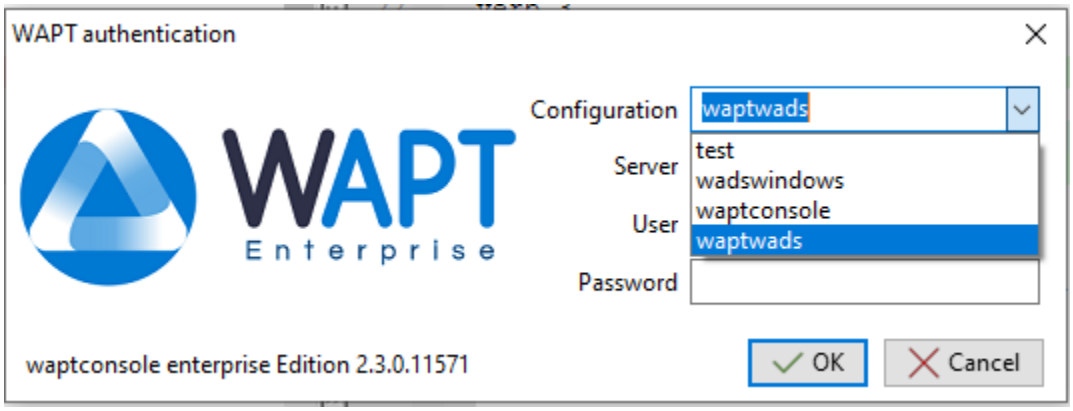
20.4 Managing several WAPT Server profiles in the WAPT Console

You can connect the WAPT Console to several WAPT Servers.

To do so, go to `%localappdata%waptconsole`, copy the `waptconsole.ini` file and rename it, for example `waptconsole2.ini`. Modify the new file with the second WAPT Server parameters (ex: IP / DNS, prefix, etc).

Then, when you re-open the WAPT Console, you can select one WAPT Server or the other.

Hint: You can have several WAPT Server connection profiles but the WAPT Servers do not communicate among them.



20.5 Using the WAPT System Tray utility

The WAPT System Tray utility is a systray program working in user context.

The WAPT System Tray utility launches at logon if the option has been ticked during WAPT Agent installation. The icon will show up in the Windows tray toolbar.

One can also launch the WAPT System Tray utility manually on `C:\Program Files (x86)\wapt\wapttray.exe`.

20.5.1 Functionalities of the WAPT System Tray utility

Main functions

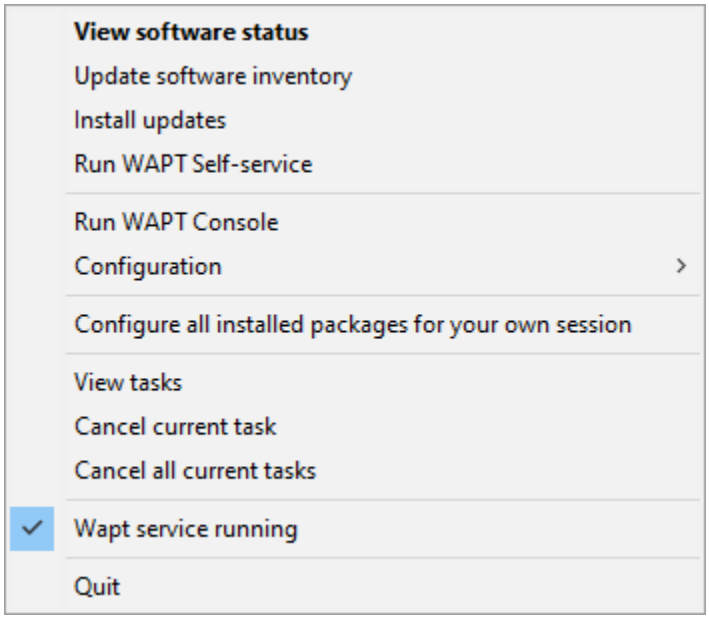


Table 2: List of functionalities of the WAPT System Tray utility

Action	Description
View software status	Launches the local web interface in a web browser.
Update software inventory	Refreshes the list of available WAPT packages. Double-clicking on the tray icon brings about the same effect.
Install updates	Launches the installation of pending upgrades.
Run WAPT Self-service	Launches the WAPT Self-Service.
Run WAPT Console	Launches the WAPT Console.
Configuration	See following table for detailed list of options.
Configuring all installed packages for your own session	Launches a **`session-setup`** to configure user environment for all packages installed on the host.
View tasks	Display the task list on the local web interface in the web browser.
Cancel current task	Cancel a running task on WAPT Agent.
Cancel all current tasks	Cancel all running tasks on WAPT Agent.
WAPT service running	Stops and reloads the WAPT service.
Quit	Closes the tray icon without stopping the local WAPT service.

Configuration functions

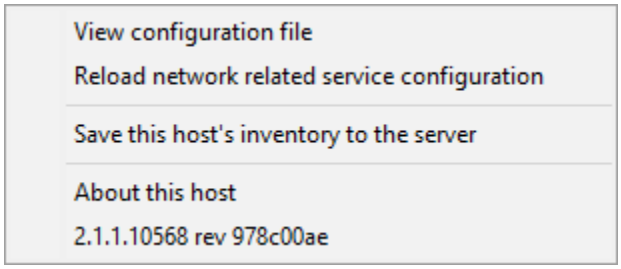


Table 3: List of configuration options for the WAPT System Tray utility

Action	Description
View configuration file	Opens the `C:\Program Files (x86)\wapt\wapt-get.ini` file with *Local Administrator* privileges (credentials may be asked).
Reload network related service configuration	Reloads the connection to the WAPT Server in the event of a network reconfiguration.
Save this host to the WAPT Server	Updates the host's inventory with the WAPT Server.
About this host	Launches the local web interface in a browser file with *Local Administrator* privileges (credentials may be asked) to display the host inventory.

20.5.2 Video demonstration

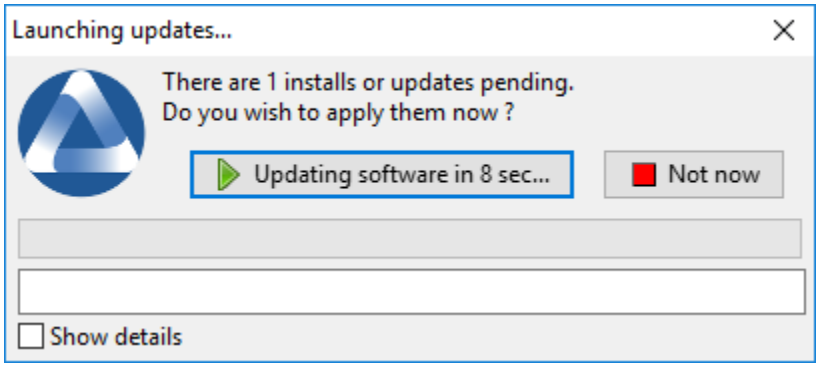
https://youtu.be/9iG36IeHuVc

20.6 Using the WAPT Exit utility

The WAPT Exit utility 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 WAPT packages are waiting to be upgraded, they will be installed.

The WAPT Exit method is very effective in most situation because it does not require the intervention of the *User* or the *Administrator*.



The WAPT Exit utility executes by default on shutdown, it is installed with the WAPT Agent.

The behavior of the WAPT Exit utility is customizable in *wapt-get.ini* of the WAPT Agent.

Warning: If a WAPT task is running, the shutdown of the host is suspended until the task has completed or timed-out.

The WAPT Exit utility can be manually executed by running `C:\Program Files (x86)\wapt\waptexit.exe`.

20.6.1 Triggering the WAPT Exit utility with a scheduled task

One can deploy a GPO or a WAPT package that will trigger the WAPT Exit utility at a pre-scheduled time.

Triggering the WAPT Exit utility 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 WAPT Exit utility script at the most appropriate time.

You can use the following script for your scheduled task, adapted to your need:

```
waptpython -c "from 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.
- The WAPT Exit utility may fail to upgrade a software program if a software that you are upgrading is in the `impacted_process` list of the `control` file. See *below* for more information.

- The method of triggering the WAPT Exit utility at a scheduled time is the least recommended method for desktops. It is better to let the WAPT Exit utility execute at shutdown or on user request.

20.6.2 The WAPT Exit utility settings in wapt-get.ini

It is possible to *modify the behavior of the WAPT Exit utility* in the `wapt-get.ini`.

It is also possible to modify the behavior of the WAPT Exit utility directly from the command line, see the next points.

20.6.3 The WAPT Exit utility options with the command line

Avoiding the cancellation of upgrades

To disable the interruption of the installation of updates you can run the WAPT Exit utility with the argument:

```
waptexit.exe -allow_cancel_upgrade = True
```

Increasing the trigger time in the WAPT Exit utility

To specify the wait time before the automatic start of the installations you can start the WAPT Exit utility with the argument:

```
waptexit.exe -waptexit_countdown = 10000
```

Avoiding to interrupt user activity

To tell WAPT not to run an *upgrade* of software titles currently running on the host (`impacted_process` attribute of the WAPT package), the WAPT Exit utility may be run with the argument `-only_if_not_process_running`.

```
waptexit.exe -only_if_not_process_running = True
```

If not specified, the WAPT Exit utility will take the value indicated in `C:\Program Files (x86)\wapt\wapt-get.ini`.

Launching the installation of WAPT packages with a special level of priority

To tell WAPT to only upgrade WAPT packages with a specific priority, you can run the WAPT Exit utility with the argument `-priorities`.

```
waptexit.exe -priorities = high
```

Registering/ unregistering the WAPT Exit utility

To register or unregister the WAPT Exit utility in local shutdown group strategy scripts, use:

- to enable the WAPT Exit utility at host shutdown:

```
wapt-get add-upgrade-shutdown
```

- to disable the WAPT Exit utility at host shutdown:

```
wapt-get remove-upgrade-shutdown
```

Video demonstration

20.7 Customizing WAPT for better user acceptance

It is possible to customize WAPT with your company colors to improve user acceptance.

3 components of WAPT are customizable:

- the WAPT Exit utility;
- the WAPT Self-Service;
- the WAPT Message utility.

It's possible to use the same logo for all programs.

Place the image in `<wapt_folder>\templates`.

The logo **MUST** be named `wapt-logo.png`

The recommended size of the logo is 200X55 and the format `.png`

For a different logo per program, see next points.

20.7.1 The WAPT Exit utility

It is possible to customize the WAPT Exit utility by placing the image you want in `<wapt_folder>\templates`

The logo **MUST** be named `waptexit-logo.png`

The recommended size of the logo is 200X55 px and the format `.png`

If it is not defined, WAPT uses `wapt-logo.png`. If it does not exist, use a default WAPT logo.

20.7.2 WAPT Self-Service

It is possible to customize the WAPT Exit utility by placing the image you want in `<wapt_folder>\templates`

The logo **MUST** be named `waptself-logo.png`

The recommended size of the logo is 200X55 px and the format `.png`

If it is not defined, WAPT uses in order `waptexit-logo.png`, `waptself-logo.png` and finally the default WAPT logo.

20.7.3 WAPT Message

It is possible to customize the WAPT Exit utility by placing the image you want in `<wapt_folder>\templates`

The logo **MUST** be named `waptmessage-logo.png`

The recommended size of the logo is 200X55 px and the format `.png`

If it is not defined, WAPT uses in order `waptexit-logo.png`, `waptself-logo.png` and finally the default WAPT logo.

20.8 Customizing the WAPT Console with its configuration file

Hint: the WAPT Console configuration is stored in 2 locations:

- `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.
- `C:\Users\%username%\AppData\Roaming\waptconsole\waptconsole.ini`.

These files are automatically generated when the **`waptconsole`** is first launched and it is generated from the `wapt-get.ini` file configured on the *Administrator*'s workstation;

20.8.1 Description of available sections

Table 4: Description of available sections for the WAPT Agent

Section	Description
`[global]`	Defines the global WAPT Console options
`[sections]`	Defines external repository options. `[wapt-template]` *repositories*
`[waptwua]`	WUA options

All sections are detailed below.

Others sections present on `C:\Users\%username%\AppData\Roaming\waptconsole\waptconsole.ini` are not editable manually, therefore they are not detailed.

Attention: For parameters both present in `wapt-get.ini` and `waptconsole.ini`, values are set in `wapt-get.ini` and copied to `waptconsole.ini`. Do not edit manually these parameters.

20.8.2 Description of available options by section

[global]

Several options are available in the `[global]` section of the `waptconsole.ini` file.

Table 5: Description of available options in `AppData\Local`

Options (Default Value)	Description	Example
`advanced_mode` (default `False`)	Launches the WAPT Console in debug mode.	
`allow_remote_reboot` (default `False`)	Allows to reboot the selected host(s) remotely from	
`allow_remote_shutdown` (default `False`)	Allows to shut down the selected host(s) remotely f	
`client_certificate` (default `None`)	Defines whether the remote repository is using Clie	
`client_private_key` (default `None`)	Defines whether the remote repository is using Clie	
`check_certificates_validity` (default `False`)	Forces the package certificate's date and CRL to be	
`default_maturity` (default `None`)	Defines the default upload maturity for WAPT pack	
`default_package_prefix` (default `tis`)	Defines the default prefix for new or imported pack	
`default_sources_root` (default `C:\waptdev` on Windows or `~/waptdev` on Linux)	Defines the directory for storing packages while in	
`grid_hosts_plugins` (default `W10=`)	Lists *external plugins* for the WAPT Console. Defa	
`host_profiles` (default `None`)	Defines a WAPT package list that the WAPT Agent	
`hiberboot_enabled` (default `False`)	Disables Hiberboot on Windows 10 to make **wapte**	
`http_proxy` (default `None`)	Defines the address of the proxy server in the WAP	
`last_usage_report` (default `None`)	Provides the date when the WAPT Console was las	
`lastwaptserveruser` (default `None`)	Provides the last user logged on this WAPT Consol	
`max_gpo_script_wait` (default `180`)	Defines the timeout for GPO execution at computer	
`personal_certificate_path` (default `None`)	Defines the path to the certificate associated with th	
`pre_shutdown_timeout` (default `180`)	Defines the timeout for scripts at computer shutdow	
`repo_url` (default your WAPT repo address)	Defines the address of the main WAPT repository.	
`send_usage_report` (default `True`)	Allows the WAPT Console to send anonymous stat	
`sign_digests` (default `sha256`)	Lists allowed signature algorithms for the WAPT p	
`use_ad_groups` (default `False`)	Allows using *unit packages*.	
`use_fqdn_as_uuid` (default `False`)	Allows using the FQDN rather than the BIOS UUI	
`use_kerberos` (default `False`)	Allows using kerberos authentication for initial reg	
`use_hostpackages` (default `False`)	Allows using *host packages*.	
`use_http_proxy_for_repo` (default `False`)	Allows using a proxy to connect to the main WAPT	
`use_http_proxy_for_server` (default `False`)	Allows using a proxy to connect to the WAPT Serv	
`use_repo_rules` (default `False`)	Allows using *replication for repositories*.	
`verify_cert` (default `False`)	Allows *verifying SSL / TLS certificate*.	
`wapt_server` (default `None`)	Defines the address of the WAPT Server.	

Table 6: Description of available options on `AppData\Roaming`

Options (Default Value)	Description	Example
`advanced_mode` (default `False`)	Launches the WAPT Console in debug mode.	advanced_mode = True
`enable_external_tools` (default `False`)	Displays the actions that call external applications (RDP, Windows tools etc…).	enable_external_tools = True
`enable_management_features` (default `False`)	Displays the button to create self-signed certificates or to create the WAPT Agent's installer.	enable_management_features = True
`hide_unavailable_actions` (default `False`)	Hides actions that are not available for the WAPT Agent	hide_unavailable_actions = True
`HostsLimit` (default `2000`)	Limits hosts displayed in the WAPT Console.	HostsLimit = 300
`language` (default language on the WAPT Client)	Forces the default langage for GUI (not for package filtering)	language = en
`lastappinifilename` (default `None`)	Defines the *`.ini`* file used to store the WAPT Console configuration.	lastappinifilename = C:\Users\%username%\AppData\Roaming\waptconsole\wap
`show_host_audit_data_tab` (default `False`)	Displays the *Audit data* tab on host inventory.	show_host_audit_data_tab = True
`use_ad_groups` (default `False`)	Allows using *unit packages*.	use_ad_groups = True
`use_fqdn_as_uuid` (default `False`)	Forces the use of the FQDN instead of the uuid BIOS as the unique host identifier in WAPT.	use_fqdn_as_uuid = True
`waptconsole.version` (default `None`)	Displays the version of the WAPT Console.	waptconsole.version = 2.0.0.9424
`waptwua_enabled` (default `False`)	Allows displaying the *Windows Update* tab on the WAPT Console.	waptwua_enabled = True

[sections]

You may add several external repositories by adding `[sections]` in `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

Attention: This parameter can be configured both in the WAPT Agent configuration and in the WAPT Console configuration `C:\Users\%username%\AppData\Local\waptconsole\waptconsole.ini`.

For information on configuring the WAPT Agent, please refer to *this point*.

See available parameters and configurations by visiting *this documentation on setting up multiple repositories*.

CHAPTER

TWENTYONE

CONFIGURING WAPT REPOSITORIES

21.1 Repository location on the WAPT Server

Operating System	Value
Debian / Ubuntu	`/var/www/wapt/`
Redhat and derivatives	`/var/www/html/wapt/`
Windows	`C:\wapt\waptserver\repository`

21.2 Replicating a repository

21.2.1 Functional overview

Hint: The method explained below is for the Enterprise version only.

The deprecated and **unsupported** Syncthing method may be used for the **Discovery** version of WAPT.

WAPT Agent replication role

Repository replication can be enabled using a WAPT Agent installed on an existing host, a dedicated appliance or a Virtual Host.

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 handled by WAPT Agents natively.

It is based on a `sync.json` file which indexes every files present in these folders:

- `wapt`;
- `waptwua`;
- `wapt-host`;
- `wads`.

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 `wapt-get.ini` configuration file.

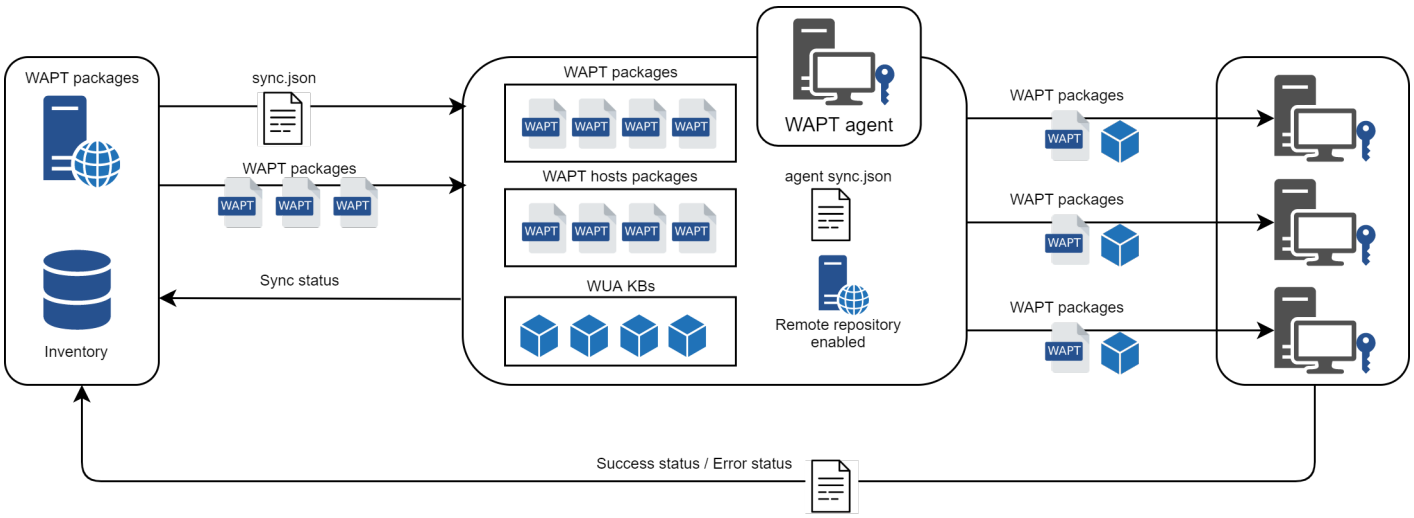


Fig. 1: Flow diagram of the replication behavior of the WAPT Agent

21.2.2 WAPT Agent configuration

To enable replication on an *existing WAPT Agent* (Linux / Windows), you need to set in the `[repo-sync]` section in the `wapt-get.ini` configuration file of the WAPT Agent.

Hint: If you use DNS, please remind to create a DNS entry for your WAPT agent.

Table 1: WAPT Agent replication configuration

Options (Default Value)	Definition	Example
`enable_remote_repo` (default `False`)	Enables remote repository to synchronize with the main repository.	`enable_remote_repo` (default `True`)
`local_repo_path` (default `WAPT root dir/repository`)	Sets the path to the root directory of the local repository for WAPT packages.	`local_repo_path = /var/www/`
`local_repo_time_for_sync_start` (default `None`)	Sets synchronization start time (HH:MM / 24h format).	`local_repo_time_for_sync_start = 22:30`
`local_repo_time_for_sync_end` (default `None`)	Sets synchronization stop time (HH:MM / 24h format).	`local_repo_time_for_sync_end = 05:30`
`local_repo_sync_task_period` (default `None`)	Sets synchronization periodicity (minutes).	`local_repo_sync_task_period = 25`
`local_repo_limit_bandwidth` (default `None`)	Sets synchronization allowed bandwidth (Mbits/s).	`local_repo_limit_bandwidth = 2`
`remote_repo_dirs` (default `wapt, waptwua`)	Defines folders to synchronize.	`remote_repo_dirs = wapt, waptwua,wapt-host`
`use_repo_rules` (default `False`)	Enables for use *repository rules*.	`use_repo_rules = True`
`sync_only_forced` (default `False`)	Synchronizes the repository only if forced.	`sync_only_forced = True`

Warning: If you modify manually `wapt-get.ini` on the remote repository, you need to restart the **WAPT service**.

Note: A ready-to-use WAPT package is available in **Tranquil IT public store** to enable repository replication on Windows or Linux based WAPT Agents.

This way, the desktop of the welcome desk in a remote office of any organization may become a WAPT repository to distribute WAPT packages to the fleet of computers in the remote office.

This special package:

- Installs and enables the **`Nginx web server`** on the remote repository.
- Configures **`Nginx`** virtualhost environment.
- Enables remote repository configuration in `wapt-get.ini`.

It is possible to automatically configure repositories with your own preferred values by editing this package.

Below is an example `wapt-get.ini` file for a WAPT Agent.

```
[global]
...
```

(continues on next page)

(continued from previous page)

```
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
```

21.2.3 WAPT Server configuration

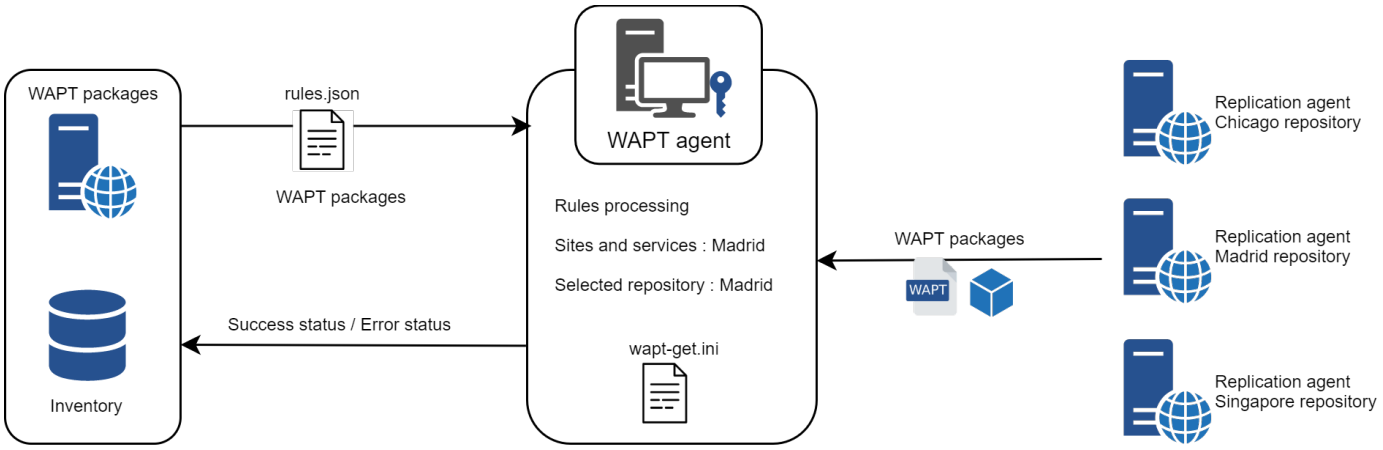
By default, the WAPT Server will know which WAPT Agents are configured as remote repositories and it will list them in the WAPT Console.

21.2.4 Repository rules

When a WAPT Agent has been configured as a repository, it will automatically retrieve its `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 remote WAPT Agents, so they may connect to their most appropriate repositories.

If no rules can be matched, the WAPT Agent will fallback to the `repo_url` attribute defined in its `wapt-get.ini` configuration file.



WAPT Agent

Warning: If you have configured GeoIP redirects on Nginx, you should disable it as it might conflict with repository rules.

To enable WAPT Agent repository rules, you **MUST** enable this setting in the `[global]` section of the `wapt-get.ini` configuration file of the WAPT Agent.

Options (Default Value)	Description	Example
`use_repo_rules` (default `False`)	For using *replicating repository*.	`use_repo_rules = True`

Below is an example `wapt-get.ini` file for a WAPT Agent.

```
[global]
...
use_repo_rules = True
```

Note: It is possible to enable this option when *generating a WAPT Agent*.

WAPT Server

On the WAPT Server, remote repositories functionality is automatically enabled.

To verify, edit `waptserver.ini` and read `remote_repo_support` value.

Options (Default Value)	Example value	Definition
`remote_repo_support`	`True`	Enables the WAPT Server to serve as a repository.

WAPT Console

Repository rules can be managed from the WAPT Console and are based on several parameters:

Table 2: Available parameters for repository rules

Options	Example value	Description
Agent IP	`192.168.85.0/24`	Defines a repository rule based on Agent IP sub-network.
Domain	`ad.mydomain.lan`	Defines a repository rule based on Active Directory domain name.
Hostname	`desktop-04feb1`	Defines a repository rule based on the hostname of the WAPT Agent.
Public IP	`256.89.299.22/32`	Defines a repository rule based on the public IP address (NATed hosts).
Site	`Paris-HQ`	Defines a repository rule based on Active Directory Sites and Services.

Adding a repository rule

To add a new repository rule, go to the *Repositories* tab in the WAPT Console and click on the *Add rule* button.

Table 3: Options for repository rules

Options	Example value	Description
Name	repo25	Defines the name for the repository rule.
Condition	AGENT IP	Defines the condition to match for the repository rule to apply (see above).
Value	192.168.25.0/24	Defines the value when the condition applies. If *NOT* is checked, the value applies to the reverse of the condition.
Repository URL	https://repo25.mydomain.lan	Defines the list of available remote repositories. The list includes http://download.windowsupdate.com/microsoftupdate/v6/wsusscan/ to allow directly downloading of Windows Updates by the remote repositories to preserve WAPT Server bandwidth.
Package type	WAPT	Defines what *types of packages* are replicated.
Other	No fallbacks	The option *No Fallback* will prevent from falling back to the main WAPT Server and will avoid undesired network congestion if the remote repository becomes temporarily unavailable.

- The option *Proxy* will need to be set if the remote repository is required to connect via a proxy.

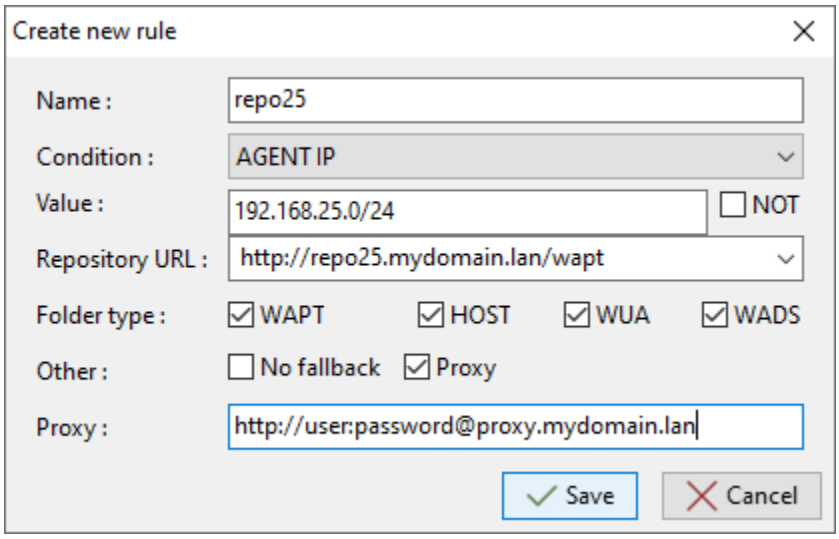


Fig. 2: Window for setting repository rules in the WAPT Console

You can then choose from the different above parameters and affect values to a specific secondary WAPT repository.

Warning: The rules are applied from top to bottom.

The first rule that matches the conditions overrides all the other rules placed under.

Danger: Do not forget to save the replication rules.

21.3 Multiple repositories

Similar to Debian repositories, it is possible for the WAPT Agent to use multiple repositories for updating package. The WAPT Agents will check all repositories.

Danger: If you use this functionality, **KNOW WHAT YOU ARE DOING**.

When using repositories with different signers, the additional signer's public certificates **MUST** be added to `C:\Program Files (x86)\wapt\ssl` on Windows or `/opt/wapt/ssl` on Linux and macOS, therefore, you **MUST** trust their work and their signature.

You then **MUST** deploy the WAPT Agents with both keys.

Please refer to the documentation on *creating the WAPT Agent* to add certificates.

21.3.1 WAPT Agent configuration

This parameters are modifiable on `wapt-get.ini` file.

Description of available parameters

Table 4: Options for defining multiple repositories

Options	Example value	Description
`[global]`	repositories = wapt-templates,private	Defines the repositories, for example `wapt-templates` and `private`, where their settings are set in additional `[section]` sections of the `wapt-get.ini` file.
`[section]`	[wapt-templates] repo_url=https://store.wapt.fr/wapt verify_cert = True [private] repo_url=https://srvwapt.mydomain.lan/wapt verify_cert = False	Defines the properties of each repository declared in the `[global]` section of the `wapt-get.ini` file.

Table 5: Options for repository properties

Options (Default Value)	Description	Example
`http_proxy` (default `None`)	Defines the HTTP proxy address.	http_proxy = http://user:pwd@host_fqdn:port
`repo_url` (default `None`)	Defines the address of the main WAPT repository.	repo_url = https://srvwapt.mydomain.lan/wapt
`timeout` (default `None`)	Defines the timeout when connecting to remote repositories (in milisecconds).	timeout = 5000
`use_http_proxy_for_repo` (default `False`)	Defines whether a proxy needs to be set to access the repositories.	use_http_proxy_for_repo = True
`verify_cert` (default `None`)	Defines whether *HTTPS certificates of the repository needs to be verified*, and if so defines the path to the certificate bundle.	verify_cert = True

Note: The WAPT Agent will look for updates in all repositories defined in its `wapt-get.ini` configuration file when doing a **`wapt-get search`**.

More info on *using WAPT with the command line interface*.

21.3.2 Configuring the WAPT Console for using multiple repositories

After having configured the WAPT Agent for using multiple repositories, we can make the repositories show up in the WAPT Console.

To do so, modify the `%appdata%\local\waptconsole\waptconsole.ini` file.

Example:

```
[wapt-template]
repo_url = https://wapt.tranquil.it/wapt
http_proxy =
verify_cert = True
public_certs_dir =
client_certificate =
client_private_key =
timeout = 5

[private]
repo_url = https://srvwapt.mydomain.lan/wapt
http_proxy =
verify_cert = False
public_certs_dir =
client_certificate =
client_private_key =
timeout = 5
```

Table 6: Options for external repositories in the WAPT Console

Options (Default Value)	Description	Example
`client_certificate` (default `None`)	Defines the folder that contains the certificates used to authenticate downloaded external packages.	client_certificate = `C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt (on Windows)`
`client_private_key` = None	Defines the folder that contains the private key.	client_private_key = C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.pem (on Windows)
`http_proxy` (default `None`)	Defines the HTTP proxy address.	http_proxy = http://user:pwd@srvproxy.mydomain.lan:port
`public_certs_dir` =	Defines the folder that contains the certificates used to authenticate downloaded external packages.	public_certs_dir = C:\private
`repo_url` (default `None`)	Defines the address of the main WAPT repository.	repo_url = https://srvwapt.mydomain.lan/wapt
`timeout` (default `None`)	Defines the timeout when connecting to remote repositories (in miliseconds).	timeout = 5000
`verify_cert` (default `None`)	Defines whether *HTTPS certificates of the repository needs to be verified*, and if so defines the path to the certificate bundle.	verify_cert = True

CHAPTER

TWENTYTWO

USING WAPT SELF-SERVICE

22.1 Presentation

With WAPT your users can have a selfservice for software installation.

It's different in the **Discovery** and **Enterprise** versions.

Functionality	Discovery	Enterprise
Access to self-service	✓	✓
Deploying self-service packages	✓	✓
Filtering self-service packages	✗	✓
Management tab	✗	✓

22.2 Working principle

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.

22.2.1 Discovery

Only Local Administrators and members of the *waptselfservice* group can access self-service on the hosts.

Attention: These users have acces to all packages in your repository.

22.2.2 Enterprise

You can filter the list of self-service packages available for your users.

A *self-service* package may be deployed on hosts to list the different self-service rules that apply to the host.

The *self-service* packages are based on user groups.

Your users will be able to install a selection of WAPT packages without having to be a *Local Administrator*.

22.3 Using the self-service feature

22.3.1 Configuration Discovery Mode

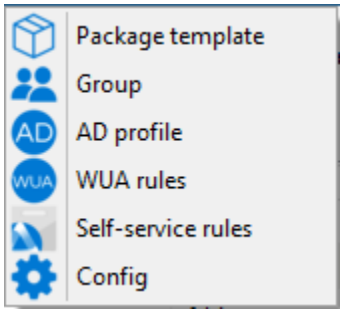
On Discovery create a *waptselfservice* security group on your Active Directory and add your users.

Note: **ALL** users in the *waptselfservice* security group and **ALL** Local Administrators will have access to **ALL** WAPT packages in the repository.

It is not possible to filter the WAPT packages made accessible to the users in Discovery mode.

22.3.2 Configuration Enterprise Mode

In the WAPT Console go to the *WAPT Packages* tab and select the *Self-service rules* menu item.

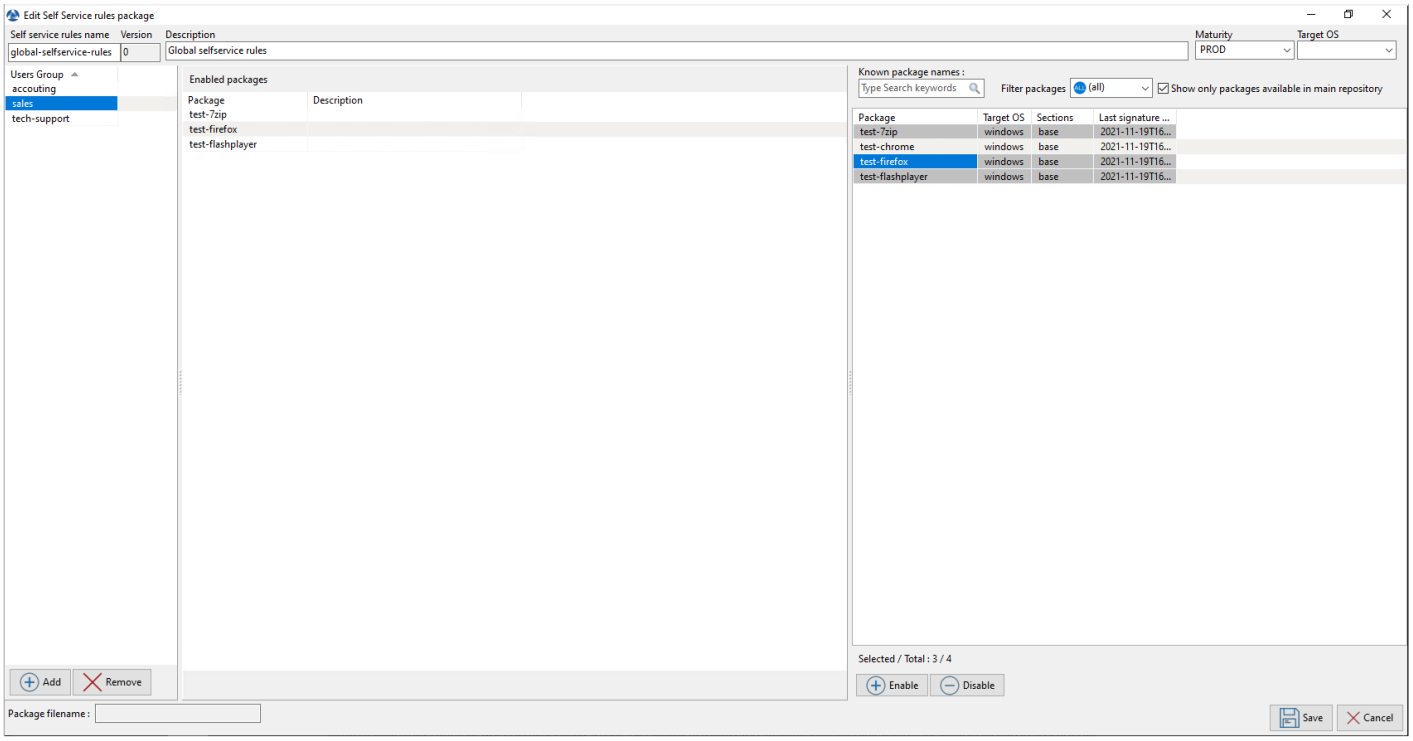


You can now create your *self-service* rules package.

1. Give a name to the *self-service* package.
2. Give a Description.
3. Click on the *Add* button to add the group (at the bottom left).
4. Name the *self-service* group (with `F2` or type directly into the cell).
5. Select Maturity *self-service* package
6. Select the target OS for which the *self-service* package is designed.
7. Drag and drop the allowed software and configuration packages for this *self-service* group into the central panel.
8. Add as many groups as needed to be included to the WAPT *self-service* package.
9. Save the WAPT package and deploy on the selected hosts.

Note:

- The name of the *self-service* package **MUST** be the same as the name of the **Active Directory user security group** to which the *self-service* rules will apply..
- If a group appears in multiple *self-service* packages, then the rules are merged.
- The authentication used is system authentication by default, it is possible to authenticate with *Active Directory*.



- Once the *self-service* package is deployed, only allowed WAPT packages listed in the *self-service* group(s) of which the *User* is a member will be shown to the logged in *User*.

22.4 Using WAPT Self-Service

WAPT Self-service is accessible in the Windows start menu under the name *Self-Service software WAPT*.

It is also available directly in the WAPT directory `<base>\waptself.exe`.

Note: The login and password to enter when launching the self-service are the User's credentials (local or Active Directory credentials).

The WAPT Self-service then displays a list of packages available for installation.

- The user can have more details on each WAPT package by clicking the + button.

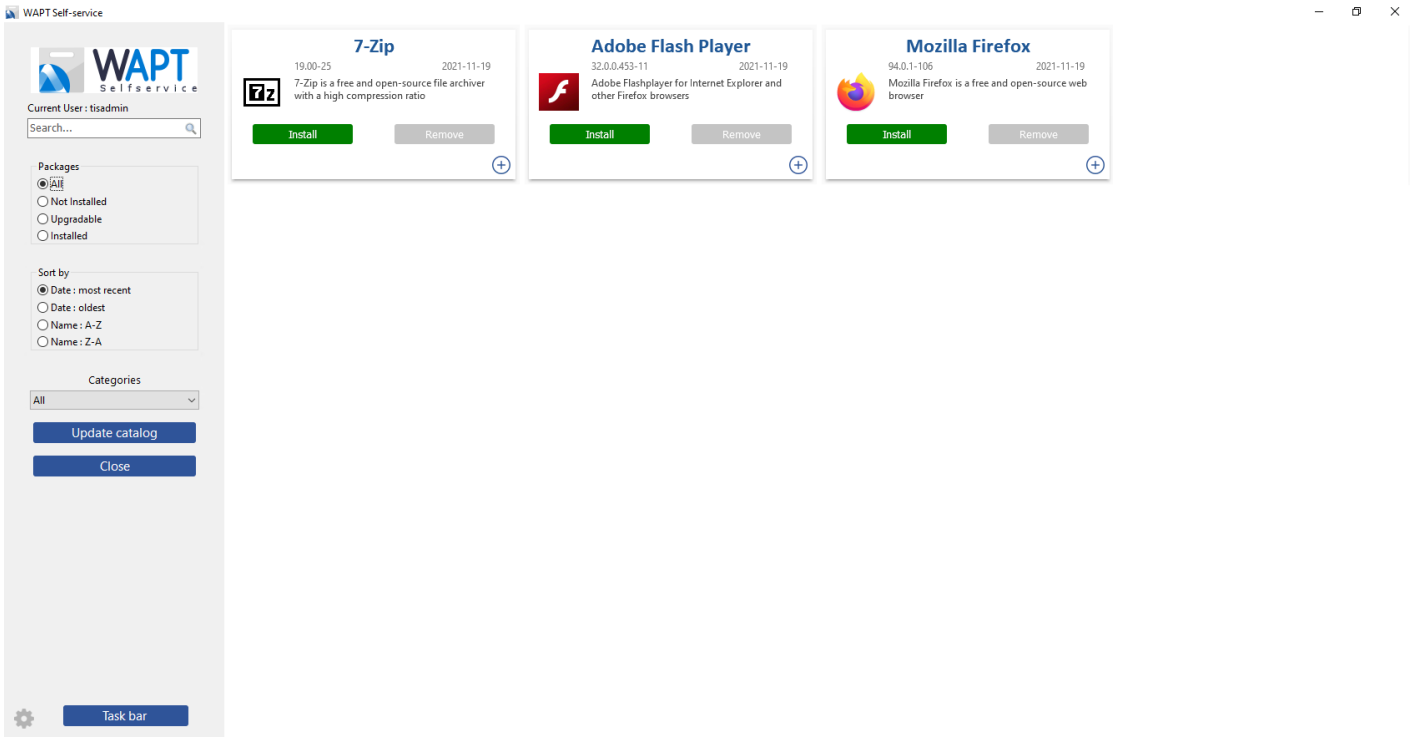
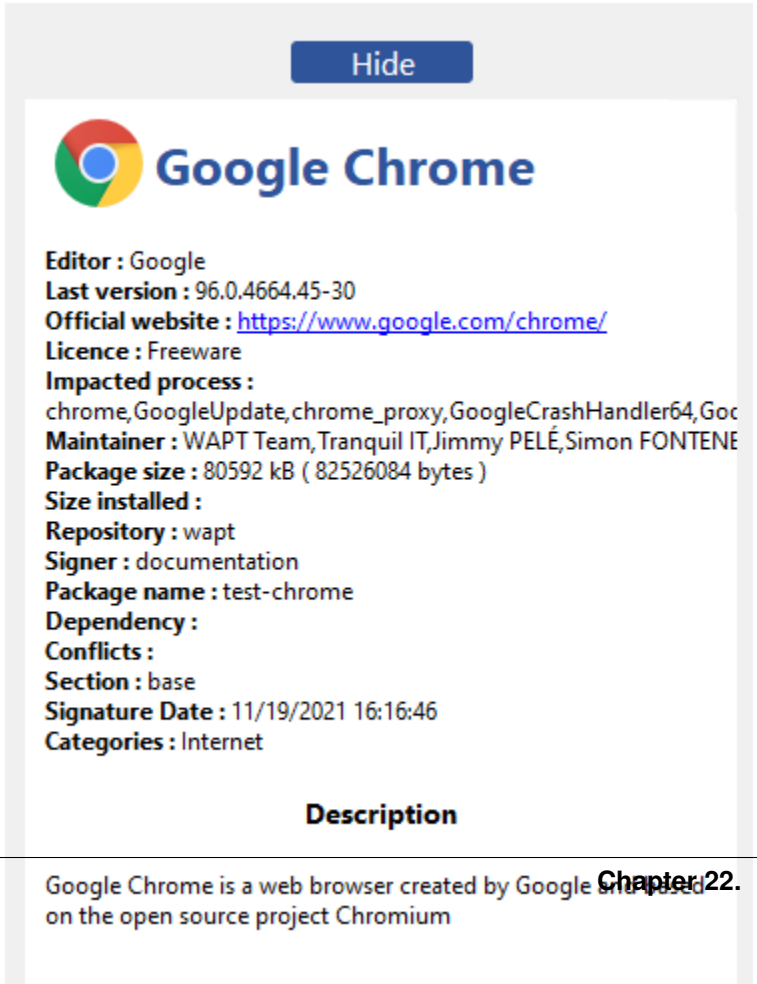
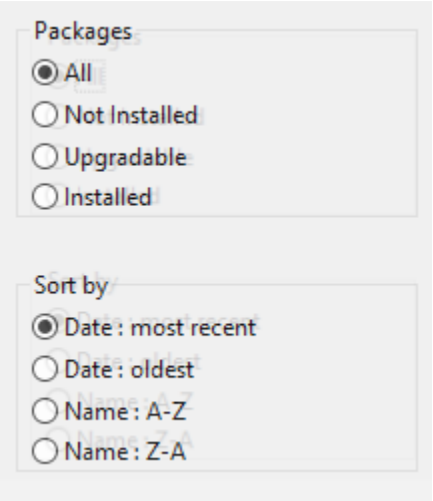


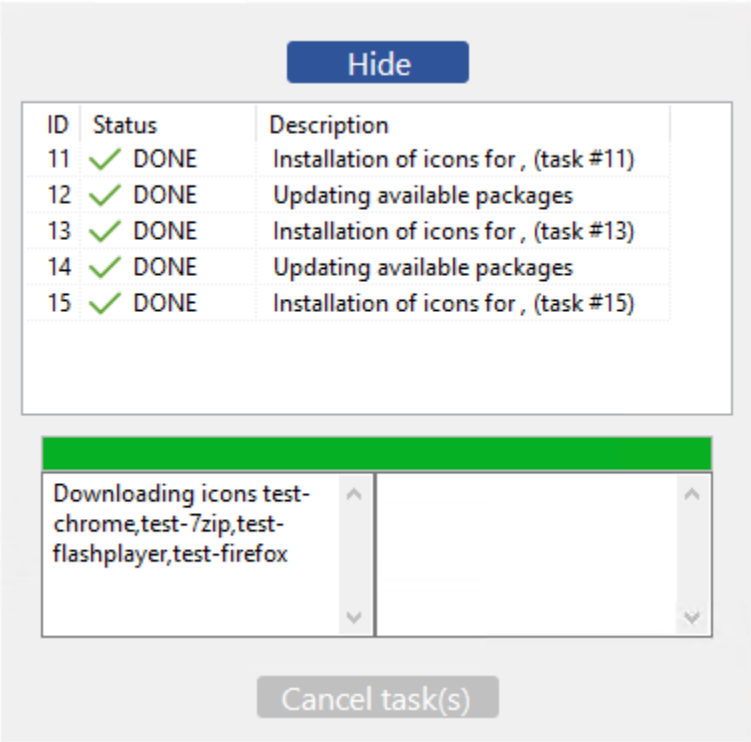
Fig. 1: Main window of the WAPT Self-service



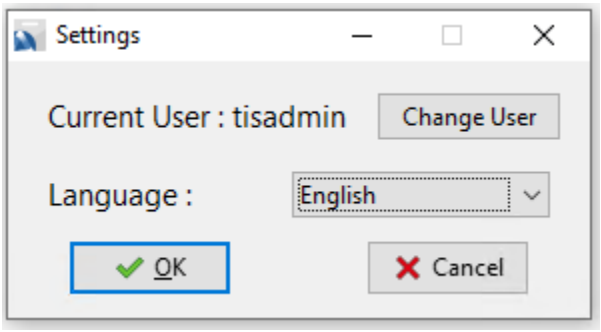
- Different filters are available for the user on the left side panel.



- The *Update Catalog* button is used to force a **`wapt-get update`** on the WAPT Agent;
- The current task list of the WAPT Agent is available by clicking the *task bar* button;



- It is possible to change the language of the interface with the button at the bottom left.



22.4.1 Default package categories available

By default, WAPT manage these categories of packages:

- Internet;
- Utilities;
- Messaging;
- Security;
- System and network;
- Storage;
- Media;
- Development;
- Office;
- Education.

It is possible to *add other categories* to the WAPT packages that you design.

22.5 WAPT Agent settings for WAPT Self-Service

WAPT Agent can be configured to allow WAPT self-service.

22.5.1 Configuring a different authentication method for the self-service

By default, authentication on WAPT service is configured in system mode.

This behavior is defined with the value of `service_auth_type` in *wapt-get.ini*:

Value	Description
`system` *Default value*	WAPT service transmits the authentication directly to the operating system; it also recovers the groups by directly interrogating the operating system.
`waptserver-ldap`	This mode allows authentication to the WAPT Server. The WAPT Server will make a LDAP request to verify authentication and groups. For this to work, you **MUST** have configured *LDAP authentication* on the WAPT Server.
`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. For this to work, you **MUST** have configured *LDAP authentication* on the WAPT Server.

You may be interested in looking up this article describing the *settings for WAPT Self-Service and the WAPT service Authentification* for more options.

Note: For the system authentication under GNU/Linux 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.

Warning: In `system` mode we assume that *Local Administrators* can see all the WAPT packages. To change this behavior see the next point.

22.5.2 Configuring the authentification for Administrator

By default WAPT Self-Service uses the `system` authentification.

In this mode, the *Local Administrators* can see all the packages of WAPT Server repository.

If you do not want this behavior there are **2** possibilities:

- Block the view of all packages for *Local Administrators*.
- All packages are only visible for a specific user group.

Block Local Administrator on self-service

To block all packages from being displayed to *Local Administrators* you have to add the parameter `waptservice_admin_filter` in `wapt-get.ini`.

Value	*True*	*False*
`waptservice_admin_filter`	Enable *selfservice package* view filtering for Local Administrators.	Disable *selfservice package* view filtering for Local Administrators.

User group self-service Administrator

It is possible to use a special user group to define a list of administrators in the Self-Service.

Create a user security group named `waptselfservice` and add members.

All members of this group can view all packages on the WAPT Self-Service.

With `waptservice_admin_filter` parameter, you have secured the administrator acces of WAPT Self-Service.

22.6 Video demonstration

https://youtu.be/-_sm8KBwDOw

CHAPTER

TWENTYTHREE

USING WAPT WINDOWS UPDATE AGENT (WAPTWUA)

Hint: WAPT is able to manage Windows Updates on your endpoints and replace automatic Windows Updates or a WSUS Server.

Note: WAPTWUA works with the WUA (Windows Update Agent) Windows API.

For more information: https://docs.microsoft.com/en-us/windows/win32/wua_sdk/using-the-windows-update-agent-api.

Attention: WAPTWUA can not operate concurrently with the Windows store.

23.1 Working principle

Video demonstration:

https://youtu.be/x36gAaT31Ko

Each Patch Tuesday (Patch Tuesday is an unofficial term used to refer the second Tuesday of each month when Microsoft releases software patches for their software products.), the WAPT Server downloads an updated `wsusscn2.cab` file from official Microsoft servers.

By default, downloads are scheduled once a day and no download is triggered if the `wsusscn2.cab` file has not changed since the last download.

Hint: In order to make WAPTWUA work, The WAPT Server must have access to:

- windowsupdate.microsoft.com
- ..windowsupdate.microsoft.com
- ..update.microsoft.com
- windowsupdate.com
- download.windowsupdate.com
- download.microsoft.com
- download.windowsupdate.com

- wustat.windows.com
- ntservicepack.microsoft.com
- go.microsoft.com
- dl.delivery.mp.microsoft.com

Even though you may choose other sources for Windows updates, ports 443 and 80 need to accept incoming trafic on the WAPT Server.

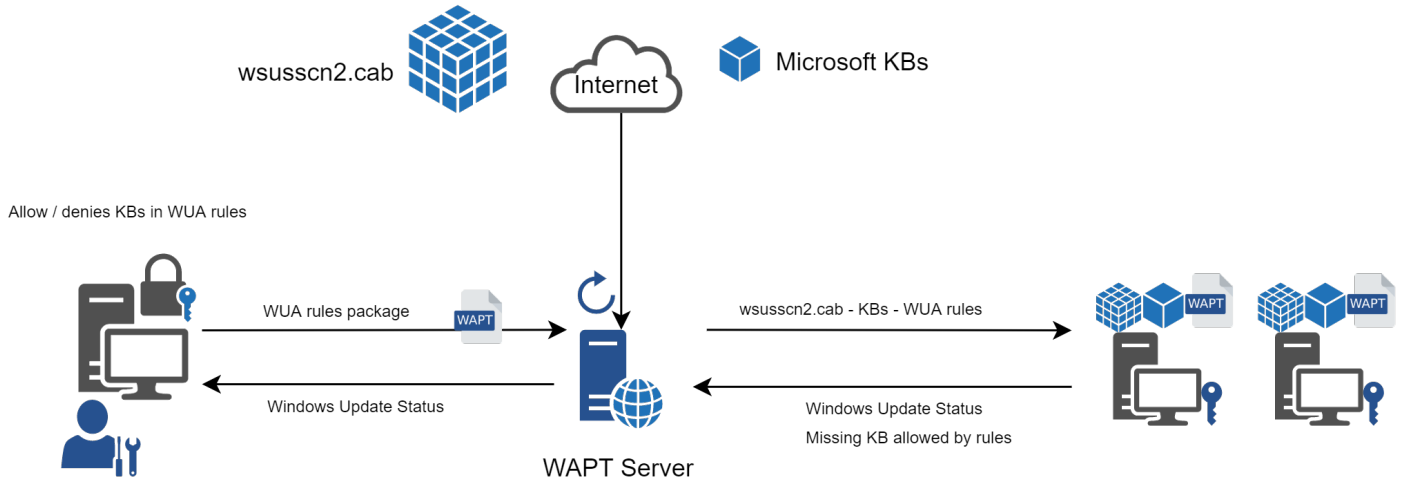


Fig. 1: Flow diagram of the WAPT Windows Updates

The `wsusscn2.cab` file is then downloaded by the WAPT Agent from the WAPT Server repository and then passed on to WUA Windows utility to crunch the update tree for the host.

Regularly, the host will analyze the available updates using the `wsusscn2.cab` file and send its list of needed updates to the WAPT Server.

If an update is pending on the host and if that update is not present on the WAPT Server, the WAPT 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: On the WAPT Server, 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`:

Note:

- If repository replication is used, it will synchronize WAPT Windows Update out of the box. For this, the `waptwua` folder needs to be included in the folder to synchronize.

If a proxy is required to access Internet, then be sure to *set the proxy server in the waptserver.ini file*.

23.2 Differences 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.

WAPT Windows Update only downloads updates that have been requested by at least one client computer. 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.

23.3 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 1903 to Windows 10 20H2.

Major version upgrades are not handled in the same way as minor OS upgrades. Major upgrades are handled via the downloading 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.

23.4 Driver upgrades

Driver upgrades via WSUS are not recommended 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 recommended 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 *Make package template from setup file → Package template → Windows Update packages (.msu)* 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.

23.5 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.

Just select the `.msu` file and click *create package* in the WAPT Console to launch the wizard to create a simple package.

To do so, follow *this documentation* on packaging `.msu` files for these *Out-of-band* updates.

Attention: You have to be careful 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.

23.6 Configuring WAPTWUA on the WAPT Agent

WAPTWUA is configured in `wapt-get.ini` in `[waptwua]` section.

You then have several options:

Table 1: Configuration options in the `[waptwua]` section in the `wapt-get.ini`

Options (Default Value)	Description	Example
`enabled` (default `False`)	Enables or disables WAPTWUA on this host.	enabled = True
`direct_download` (default `False`)	Defines whether updates are downloaded directly from Microsoft servers.	direct_download = True
`default_allow` (default `False`)	Defines whether missing update are authorized by default.	default_allow = True
`download_scheduling` (default `None`)	Defines the Windows Update scan recurrence (Will not do anything if *waptwua* package rule or `wsusscn2.cab` file have not changed).	download_scheduling = 1d
`install_scheduling` (default `None`)	Defines the Windows Update install recurrence (Will do nothing if no update is pending).	install_scheduling = 2h
`install_at_shutdown` (default `False`)	Defines whether updates are triggered on host shutting down.	install_at_shutdown = True
`install_delay` (default `None`)	Defines a deferred installation delay before publication in the repository.	install_delay = 15d
`allowed_severities` (default `None`)	Defines a severity list that will be automatically accepted during a WAPT windows update scan. ex: *Important*, *Critical*, *Moderate*.	allowed_severities = Important
`waptexit_disable_skip_windows_update` (default `False`)	Define if the skip Microsoft Windows Update tick box in WaptExit window to skip Windows Update is available (value False) or not (value True)	waptexit_disable_skip_windows_update = True
`include_potentially_superseded_updates` (default `False`)	Define whether the Windows Update agent will show both the latest KB and the superseeded ones (True), or only the lastest KB (False).	include_potentially_superseded_update = True

Hint: These options can be set when generating the WAPT Agent.

Example `[waptwua]` section in `wapt-get.ini` file:

```
[waptwua]
enabled = True
default_allow = False
direct_download = False
download_scheduling = 7d
install_at_shutdown = True
install_scheduling = 12h
install_delay = 3d
```

When creating the `waptagent.exe` from the WAPT Console, these options are equivalent to this:

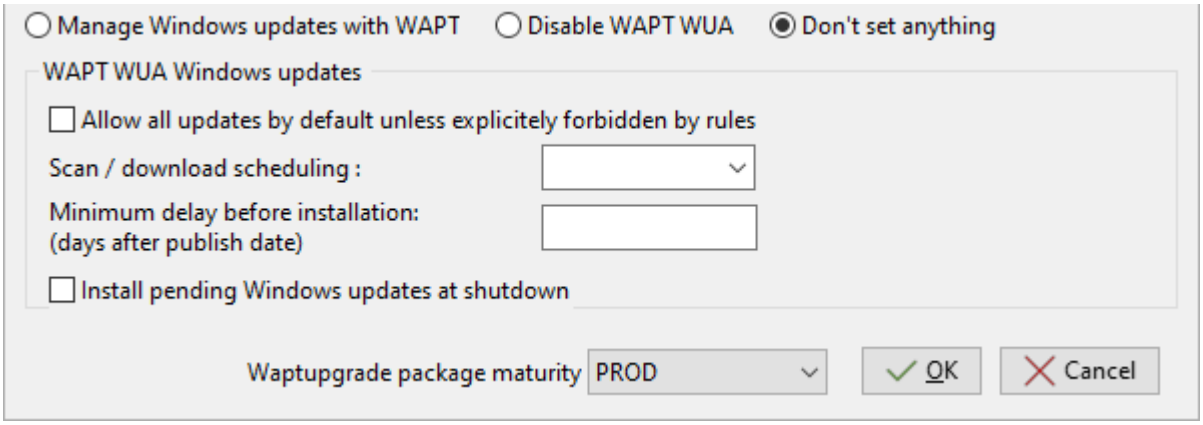


Fig. 2: Menu options for the WAPT Windows Update Agent

Example source code to modify `[waptwua]` settings with a WAPT package:

```
def install():

  inifile_writestring(WAPT.config_filename,'waptwua','enabled','true')
  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')

  print('Reload WAPT configuration')
  WAPT.reload_config_if_updated()
```

23.7 Using WAPTWUA from the WAPT Console

The WAPTWUA is managed with two tabs in the WAPT Console.

WUA Rules sub-tab in WAPT Package tab

The *WUA Rules* tab allows you to create *waptwua* rule packages.

- When a *waptwua* package is installed on a host, it indicates to the WAPTWUA Agent what are the authorized or forbidden KBs (Knowledge Base articles).
- When several *waptwua* packages are installed on a host, 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`.

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`.

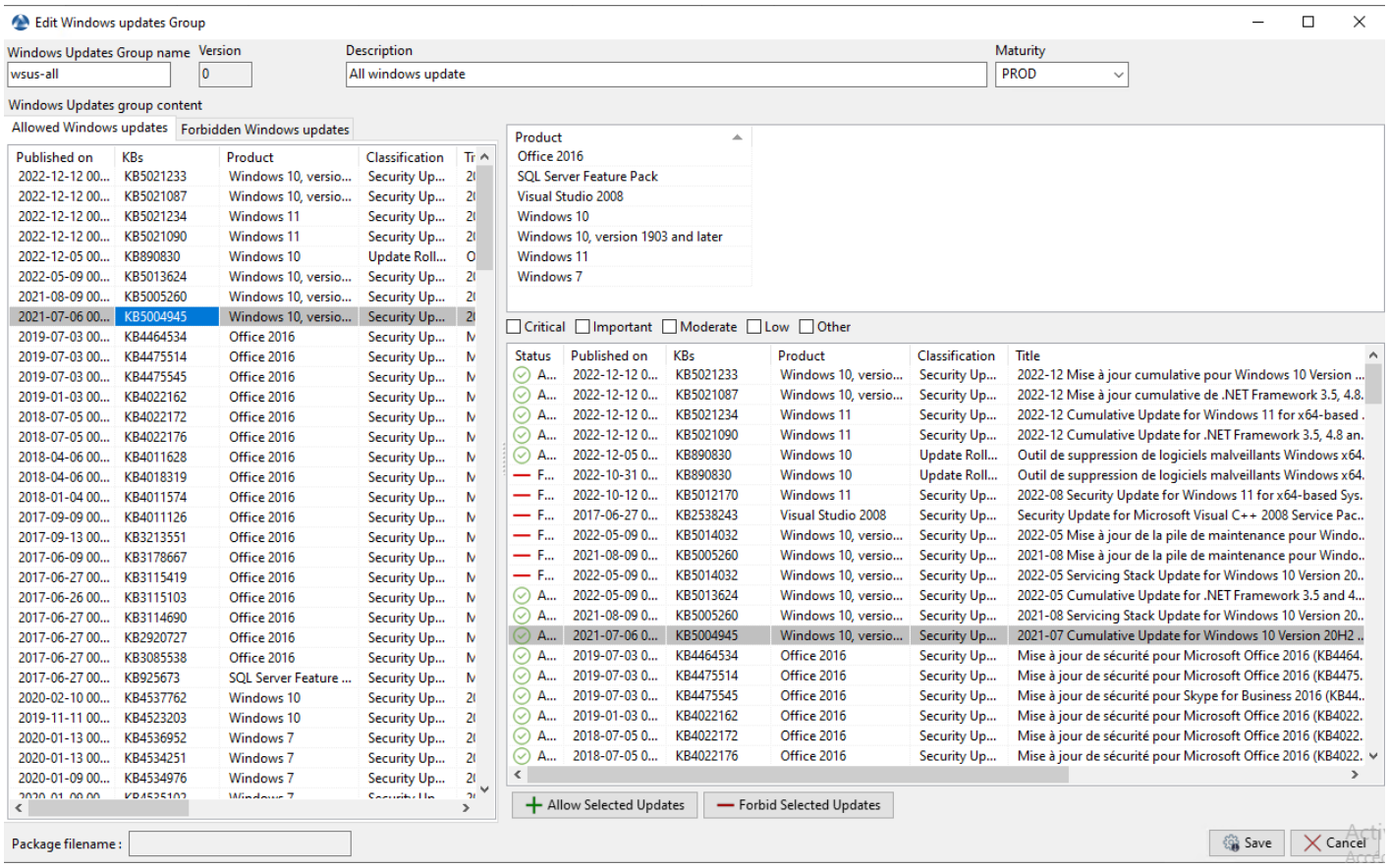


Fig. 3: Creating a waptwua package in the WAPT Console

Hint:

- To test updates on a small set of computers, you can set WAPTWUA default maturity to `PREPROD`.
- You can then test the Windows Updates on a small sample of `PREPROD` hosts and if everything is good, you can release the updates to the entire fleet of computers.

Windows Updates tab

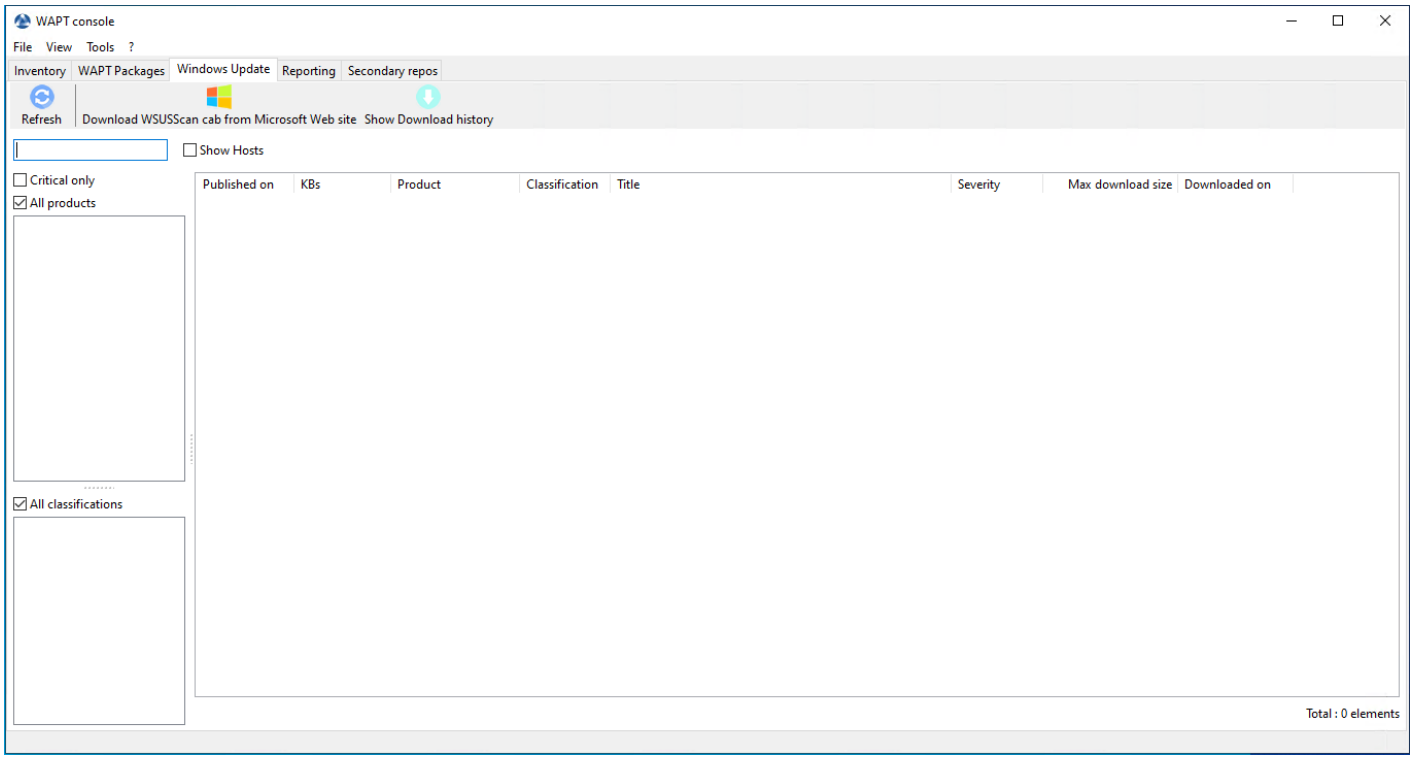


Fig. 4: Windows Updates tab in the WAPT Console

The *Windows Update* tab lists all needed Windows Updates.

Important: The WAPT Server does not scan the `wsussc2.cab` itself, it lets the Windows Update Agent utility present on all Windows hosts do it. If an update seems to be missing from the list, you **MUST** run a scan on one of the hosts present in the WAPT Console. If you run a WAPT WUA scan on a Windows 10 client, the CAB and Windows 10 files will be displayed on the *Windows Update* tab.

The left pane displays update 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).

- To force download an update, do *Right-click* → *Download*.
- To force download the `wsusscn2.cab` file, click on the *Download WSUSScan cab from Microsoft Web site* button.
- To see the Windows Updates downloaded on the WAPT Server, click on the *Show download task* button.

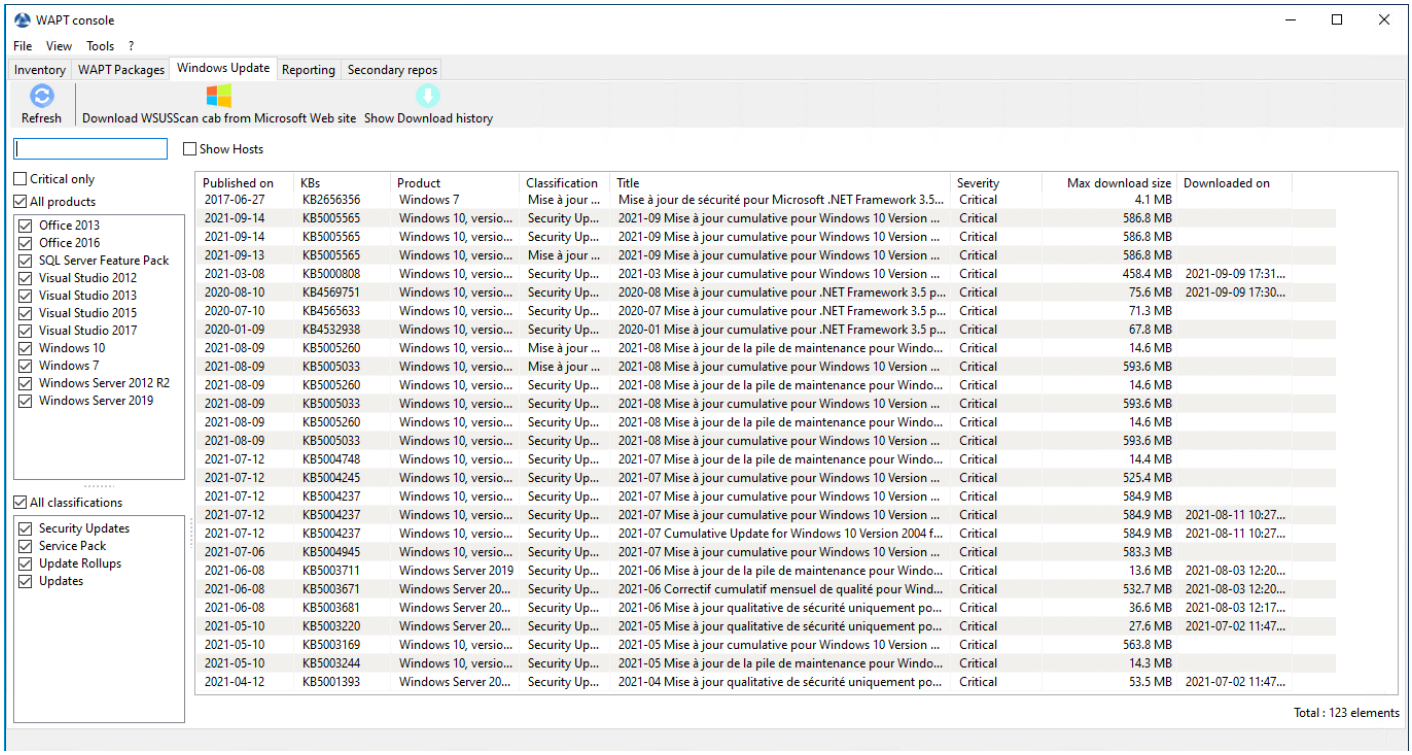


Fig. 5: Listing of Windows Updates in the WAPT Console

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 a Windows update has been requested by a WAPT Client and the requested Windows Update is not cached, 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 the tab *Windows Updates* → *Windows Updates list*

23.7.1 Cleaning old Windows updates

You can run the cleanup either manually or automatically.

Automatically

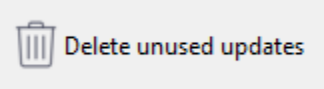
If the KB is not installed on the host, it is automatically deleted on the WAPT Server between 2:30 am and 3:30 am every day. It is possible to disable the automatic deletion of KB with the `cleanup_kbs` option in the `waptserver.ini` configuration file of the WAPT Server.

Add this setting on *the WAPT Server configuration file*:

```
cleanup_kbs = False
```

From the WAPT Console

To cleanup the `waptwua` folder, go to the *Windows update* tab and click on the *Delete Unused KB* button. Pressing the button will delete all useless KB stored on the WAPT Server.



From the WAPT Server

It is possible to delete manually from the WAPT Server any Windows Update file that is no longer required.

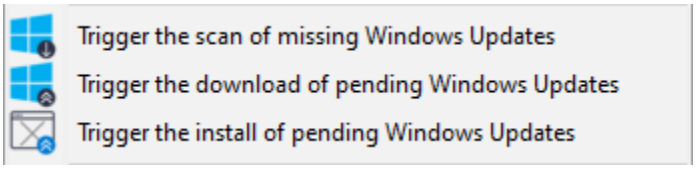
The WAPT Server will only re-download deleted updates if any of the WAPT Agents requests it.

On the WAPT Server, downloaded updates are stored:

- On Linux hosts in `/var/www/waptwua`.
- On Windows hosts in `C:\wapt\waptserver\repository\waptwua`.

23.7.2 Launching WUA on clients

From the WAPT Console you have three options.



- The *Trigger the scan of pending Windows Updates* button will launch the scan on the client and list all updates flagged for the OS.
- The *Trigger the download of pending Windows Updates* button will launch the downloading of pending updates on the client.
- The *Trigger the install of pending Windows Updates* button will launch the install of downloaded updates on the client.

Hint: When pending updates stored in cache need to be installed, the WAPT Agent triggers the WUA service.

The WAPT Agent will then enable and start the WUA Service temporarily to install the updates. When updates are installed, the WAPT service will stop and disable the WUA service until the next cycle.

23.7.3 State of Windows Update on the host

Windows updates can have 4 states on a host.

Status	Description
OK	A Windows update has installed correctly.
MISSING	A Windows update has not yet been downloaded to the WAPT Server.
PENDING	The WAPT Server knows it has to download an update from official Microsoft servers.
DISCARDED	A Windows update was forbidden by rules.

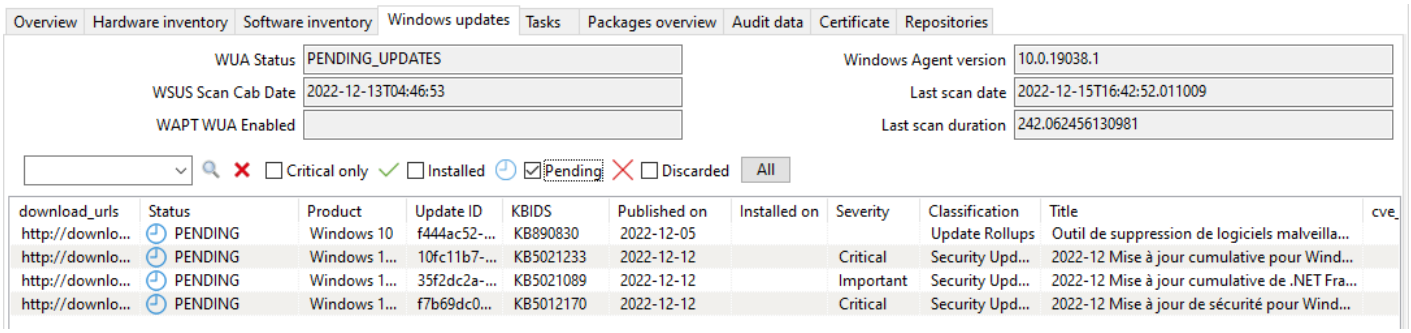


Fig. 6: Pending Windows Updates showing in the WAPT Console

23.7.4 Notion of UpdateID

In WAPT we do not use *kbids* but instead we use **updateids**.

This allows us to be finer in the management of updates.

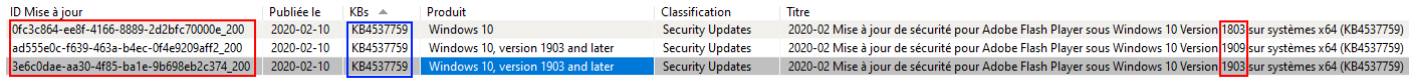


Fig. 7: List showing duplicate KB in the WAPT Console

In this example, KB4537759 appears multiple times because there are 3 different *updateids*:

- win10 1803;
- win10 1903;
- win10 1909;

You should therefore authorize *updateids* and not *KB ids*.

23.8 WAPT does not force Windows to uninstall a Windows Update

Attention: Uninstalling a Windows update can be dangerous for the host.

When an update is detected as forbidden by WAPT, its removal will **NOT** be forced.

If you really want to uninstall an update, you should package the KB that you want to uninstall as a standard WAPT package.

Here is an example:

```
from setuphelpers import *

uninstallkey = []

def install():
  with EnsureWUAServRunning():
    run('wusa /uninstall /KB:4023057')
```

CHAPTER

TWENTYFOUR

SIMPLIFYING THE DEPLOYMENT OF YOUR WORKSTATIONS

Many companies and administrations include software and configurations in the OS images they deploy on their fleets of hosts.

But from now on this is no longer the recommended method for several reasons:

- 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, you will have to keep track of the changes in a text document, a spreadsheet, or a change management tool.
- OS editors (notably Microsoft) advise the use of raw ISO images and their parameterization in post-install.
- Finally, if you introduce in your image security configurations, network configurations, or configurations to limit the intrusion of telemetry, these configurations can disrupt the normal functioning of WAPT, it will complicate future diagnostics.

With WAPT this is no longer necessary

24.1 Recommendations

Tranquil IT recommends:

- To make only one raw image per OS type with MDT, Fog (win10, win2016, etc) or *WAPT WADS* 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 create as many Organizational Units as you have host 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 by 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.

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.

24.1.1 Deployment scenario

- You receive or the IT manager at the remote site receives a new computer in its box.
- You configure the host's MAC address in DHCP so that it gets the right system image and is positioned in the right Organizational Unit at the end of the deployment process.
- The expected system image is downloaded on the host in masked time, the host is placed in the right Organizational Unit.
- The WAPT Agent registers the host with the WAPT Server, it appears in the 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 hosts or host by host, you finalize the configuration of the hosts 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).

24.2 Deploying your workstations via WADS

WADS for WAPT Automated Deployment Services was developed to provide a simple solution for Operating System deployments via WAPT.

The OS Deployment is available for Windows, Debian and its derivates and for Redhat and its derivates.

24.2.1 WADS mode of operation

Schematically, deploying an OS involves **3** steps:

1. Importing the different media and files required for the deployment, such as Operating System `.iso` images, driver packs and configuration files.

1. Creating the boot support.

3. Launching the deployment on the target host using the network or a USB stick.

24.2.2 Difference between WADS and other solutions

- **Classic** deployment solution.
- **WADS** deployment solution.

Hint:

- The WADS operating mode respects the recommanded method of the OS vendor.
- With WADS, all functionnalities are grouped on the same WAPT Server.

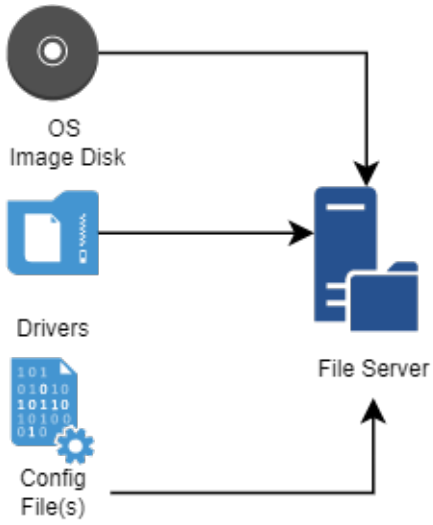


Fig. 1: Flow diagram for importing the files required for the WADS deployment

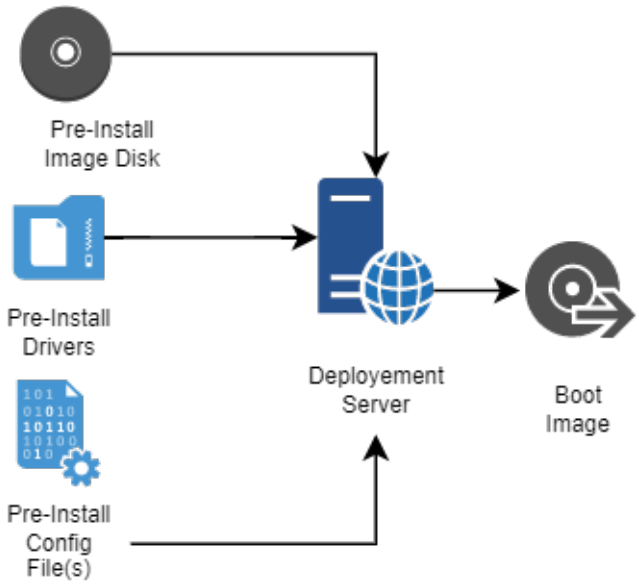


Fig. 2: Flow diagram for creating the booting support for the WADS deployment

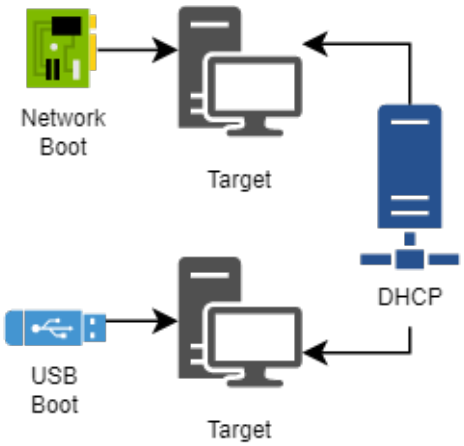


Fig. 3: Flow diagram for using the boot support in the WADS deployment

- Therefore, there is no need to set up any additional infrastructure other than the WAPT Server.

Software differences

Table 1: Differences between WADS and other methods

WADS deployment Server	Standard MDT method	Benefit
Uses iPXE	Uses CIFS (Common Internet File System) file server protocol	No need to setup a file server and no need to open additional ports
No OS image configuration needed	Requires manually editing an answer file configuration	Simplicity, all configurations are provided by WAPT
Uses HTTPS to download the Windows OS image	Uses CIFS to download the Windows OS image	The target hosts may be deployed over the internet using the USB stick method
The WADS method embeds all necessary files	The MDT method requires assembling files from different sources	The deployment, the configuration and the OS updates are bundled into one WAPT software package

24.3 Installing and configuring TFTP and DHCP for WADS

24.3.1 Installing and configuring a TFTP server

Warning: If you have installed another tftp server on the WAPT Server, please uninstall it first.

This documentation is for WAPT 2.2.1 and higher

Choose your distribution

Linux Debian / Ubuntu / Redhat

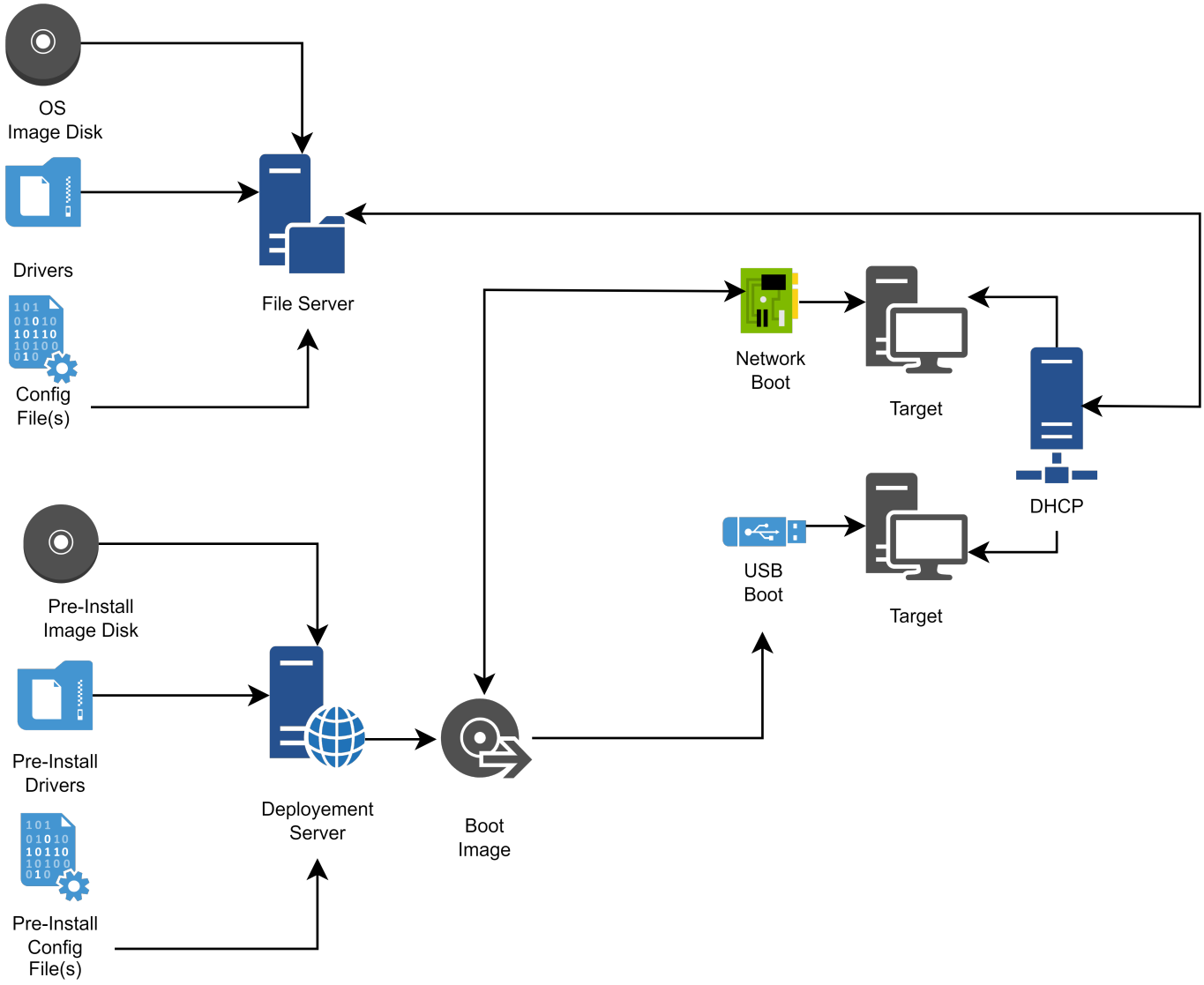


Fig. 4: Flow diagram for a classic OS deployment

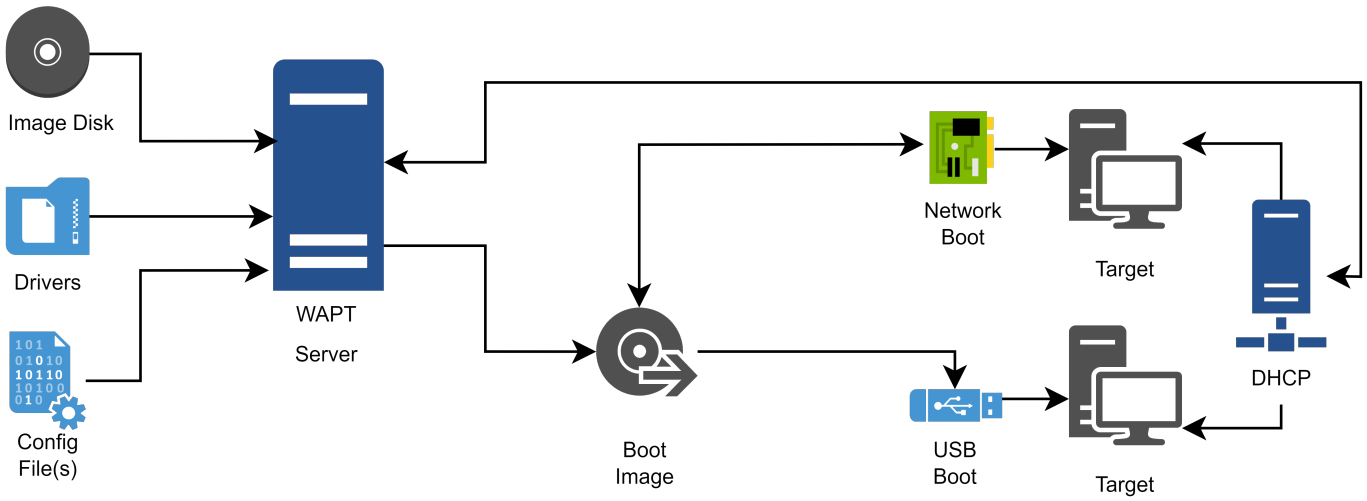


Fig. 5: Flow diagram for a WADS deployment

- Enable and start tftp serverInstall the TFTP server.

```
systemctl enable wapttftpserver
systemctl start wapttftpserver
```

- You may test that the tftp server works properly using a tftp client and test download the ipxe.efi file. If you are testing the following command on a Redhat based machine other than the waptserver, beware of the local outbound firewall that blocks outgoing tftp client requests.

```
cd ~
tftp srvwapt.mydomain.lan
  binary
  get ipxe.efi
  quit
ls -l ipxe.efi
```

Windows

- When installing the server, tick the WADS tftp check mark. You can re-run the installer if it was not done at that time. You can check that the service is configured and running with the command

```
sc query wapttftpserver
```

- If the server is installed but not started, you can start it with:

```
net start wapttftpserver
```

24.3.2 Installing and configuring a DHCP server

The PXE booting is a two step process. First the UEFI/BIOS bootloader will download iPXE binary from the tftp server, then iPXE binary will download the iPXE script and boot binaries from http. This is why we need to have a two step PXE DCHP configuration.

DHCP server

For example:

```
<!-- global options -->
next-server 192.168.1.30;

option ipxe-url code 175 = text;
option client-architecture code 93 = unsigned integer 16;

<!-- subnet mydomain.lan netmask 255.255.255.0 -->

if option client-architecture = 00:00 {
  if exists user-class and option user-class = "iPXE" {
    filename "http://srvwapt.mydomain.lan/api/v3/baseipxe?uefi=false&keymap=fr";
  }
  else{
    filename "undionly.kpxe";
  }

} else {
  if exists user-class and option user-class = "iPXE" {
    option ipxe-url "http://srvwapt.mydomain.lan:80/";
    filename "http://srvwapt.mydomain.lan/api/v3/baseipxe?keymap=fr";
  }
  else{
    filename "ipxe.efi";
  }
}
```

For more information you can refer to https://ipxe.org/howto/dhcpd

DNSMASQ server

For example:

```
dhcp-match=set:ipxe,175 # iPXE sends a 175 option.
dhcp-boot=tag:!ipxe,undionly.kpxe,IP_WAPTSERVER
dhcp-boot=tag:ipxe,http://srvwapt.mydomain.lan/api/v3/baseipxe?uefi=false
```

For example for one machine:

```
dhcp-match=set:ipxe,175 # iPXE sends a 175 option.
dhcp-mac=set:waptserver,MAC_ADDRESS_TARGET_COMPUTER
dhcp-boot=tag:!ipxe,undionly.kpxe,waptserver,IP_WAPTSERVER
dhcp-boot=tag:ipxe,http://srvwapt.mydomain.lan/api/v3/baseipxe?uefi=false,waptserver
```

Windows

You can use the following PowerShell command line to configure iPXE booting on your network. Please adapt the *$url_waptserver* and *$waptserver_ipaddress_tftp* depending on your current installation. *keymap* is the keyboard language

```
$waptserver_ipaddress_tftp = "192.168.154.13"
$url_waptserver = "http://srvwapt.mydomain.lan"
$keymap = "fr"

Add-DhcpServerv4Class -Name "legacy_bios" -Type Vendor -Data "PXEClient:Arch:00000"
Add-DhcpServerv4Class -Name "iPXE" -Type User -Data "iPXE"

Set-DhcpServerv4OptionValue -OptionId 66 -Value "$waptserver_ipaddress_tftp"

Add-DhcpServerv4Policy -Name "wapt-ipxe-url-legacy" -Condition AND -UserClass EQ,iPXE -VendorClass␣
↪EQ,legacy_bios*
Set-DhcpServerv4OptionValue -PolicyName "wapt-ipxe-url-legacy" -OptionID 67 -Value "$url_waptserver/
↪api/v3/baseipxe?uefi=false&keymap=$keymap"

Add-DhcpServerv4Policy -Name "wapt-ipxe-url-uefi" -Condition AND -UserClass EQ,iPXE -VendorClass NE,
↪legacy_bios*
Set-DhcpServerv4OptionValue -PolicyName "wapt-ipxe-url-uefi" -OptionID 67 -Value "$url_waptserver/
↪api/v3/baseipxe?keymap=$keymap"

Add-DhcpServerv4Policy -Name "ipxe.efi" -Condition AND -UserClass NE,iPXE -VendorClass NE,legacy_
↪bios*
Set-DhcpServerv4OptionValue -PolicyName "ipxe.efi" -OptionID 67 -Value "ipxe.efi"

Add-DhcpServerv4Policy -Name "undionly.kpxe" -Condition AND -UserClass NE,iPXE -VendorClass EQ,
↪legacy_bios*
Set-DhcpServerv4OptionValue -PolicyName "undionly.kpxe" -OptionID 67 -Value "undionly.kpxe"

For more information, you can refer to https://ipxe.org/howto/msdhcp
```

24.4 Deploying a Windows OS via WADS

24.4.1 Deployment process

1. **Using BIOS/UEFI**:
   - the host makes a *DHCP* request to obtain an *IP* and the *PXE configuration* (TFTP server IP & iPXE file name), or
   - the host boots from a USB stick which embeds the *PXE configuration*
2. **Using BIOS/UEFI**:
   - the host makes a *TFTP* request to get *iPXE* and her configuration, or
   - the host runs the *iPXE configuration* from the USB stick.
3. Then, using **iPXE**, the host makes a *HTTPS* request to the WADS Server to obtain the BCD (Boot Configuration Data) and the `WinPE` file.

4. Finally, using **WinPE**, the host contacts the WADS Server via *HTTP* to obtain the OS iso file and its associated configuration files.

24.4.2 Requirements before starting

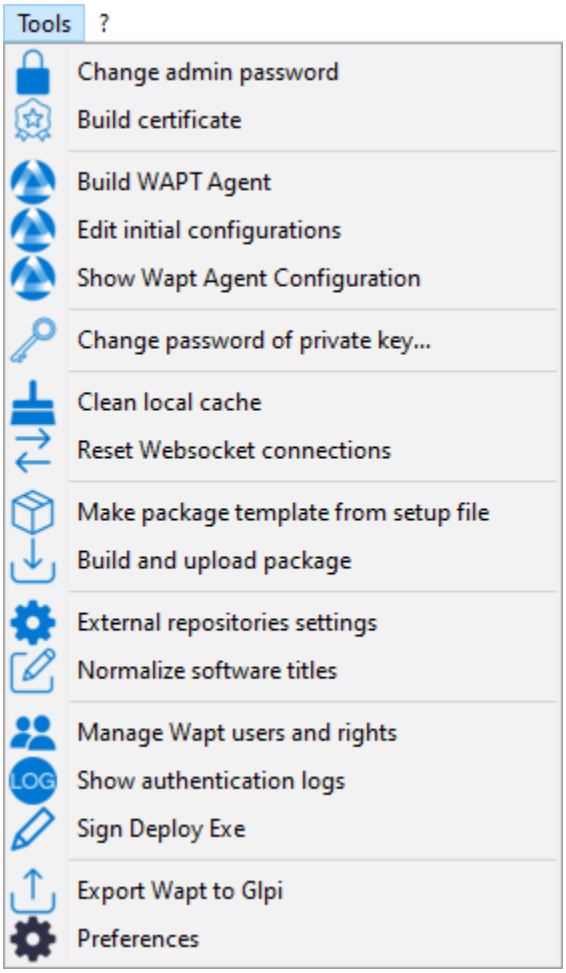
1. To use WADS on your WAPT Console, you need to install a specific package on your management station.

Two packages are available, only one is needed. Choose according to your needs:

- This package integrates the **minimal requirements** for creating a WinPE file.
- This package installs **Windows ADK**, all the tools to create and modify WinPE.

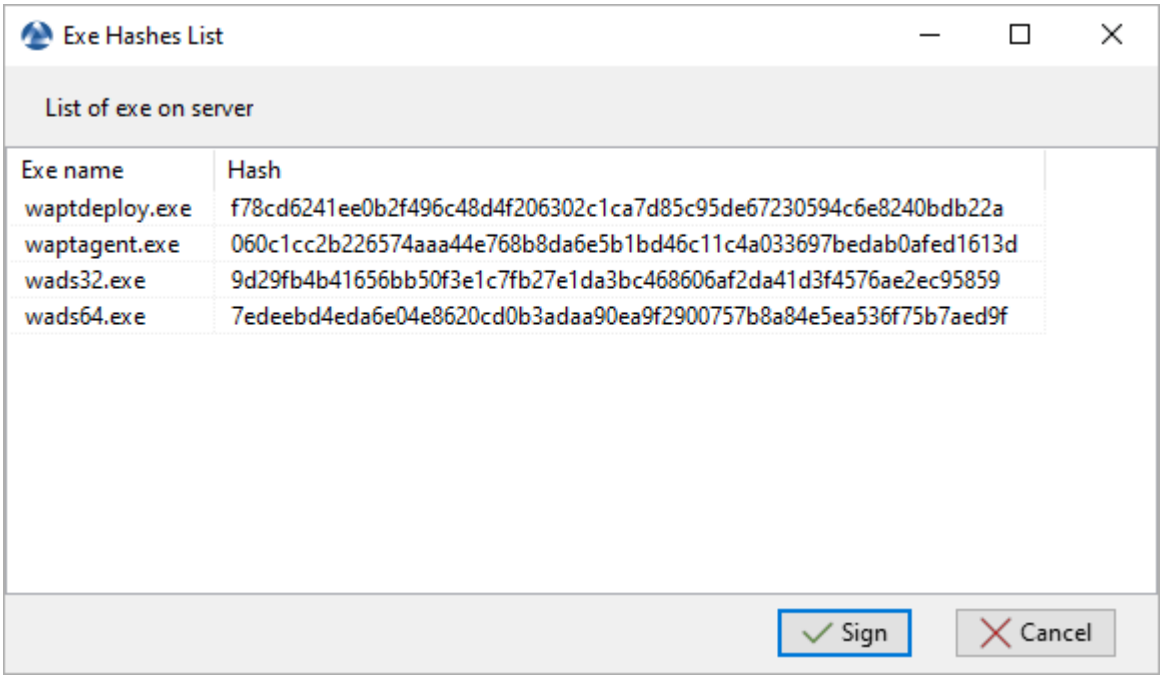
2. As of 2024-09-20, the user account using the WADS Console **MUST** have Local Administrator rights in the *WAPT Access Control Lists*.
3. Signing WADS with your certificate:

- Go to the *Tools* → *Sign Deploy Exe*.



- Click on the *Sign* button:

4. Go to the *OS Deploy* tab:



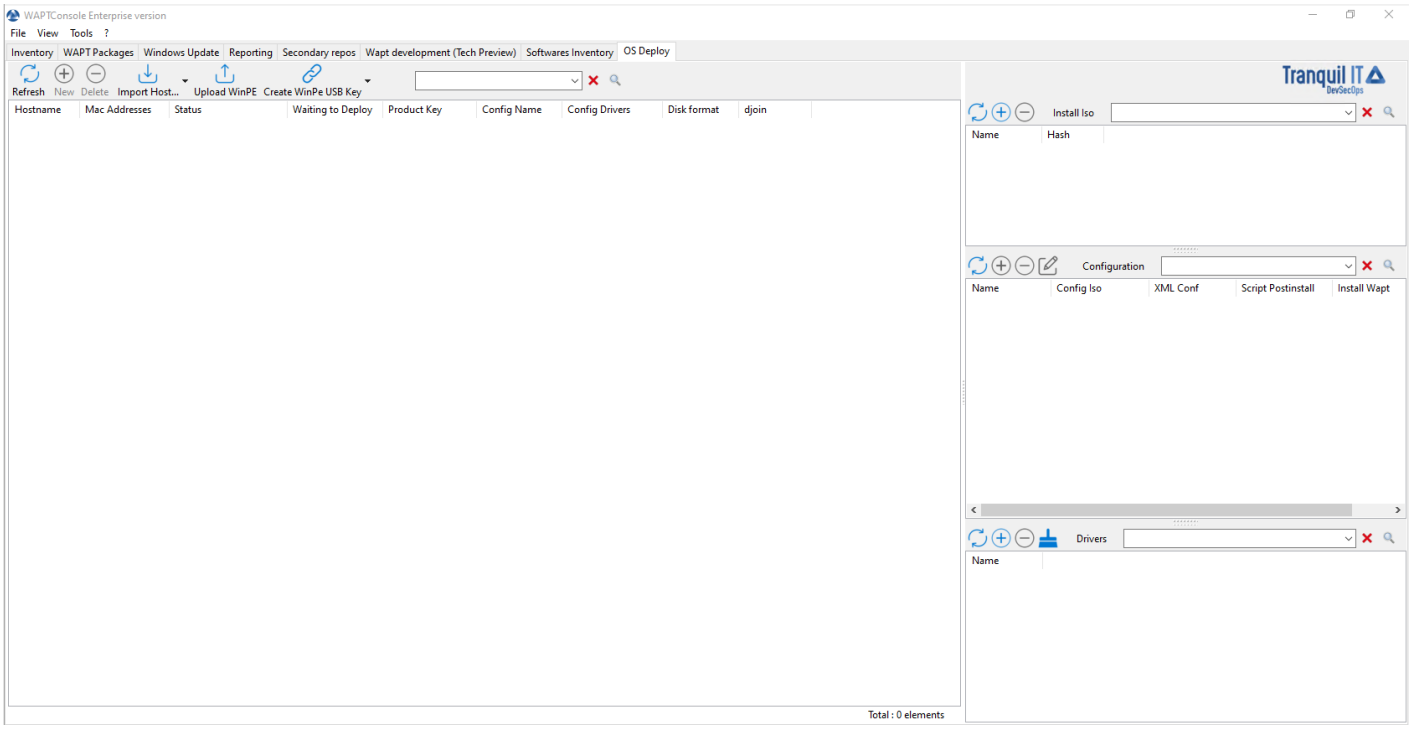


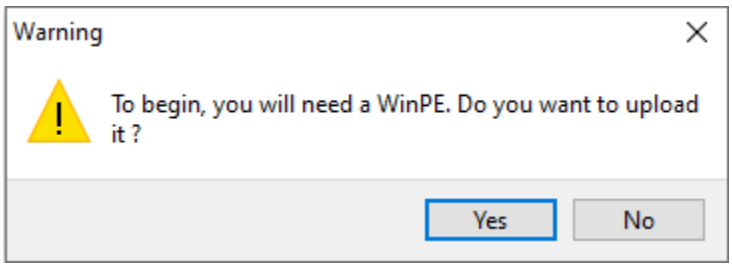
Fig. 6: Main window of the WADS Console

24.4.3 Adding the WinPE files

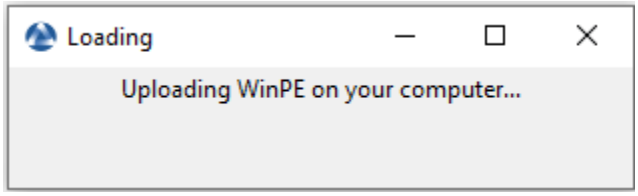
WinPE is a minimal operating system used to install, deploy, and repair Windows.

On WADS, WinPE is used to bootstrap the deployment of Windows.

- If no WinPE file exists, then this pop-up will appear.



- Then click on *Upload WinPE*.
- Choose the keyboard layout. **This step is important because you will type in the hostname in WinPE using the keyboard layout chosen with this step**.
- Select the certificate with which to sign the USB stick files
- If needed, please add network drivers in order to boot with PXE
- Wait while the `WinPE` file uploads onto the WAPT administration computer:



- Wait while the `WinPE` file uploads to the WADS Server:

If The `WinPE` file has been successfully uploaded to the WADS Server.

Hint: After each upgrade, you'll have to resign your WinPE. Don't forget to keep up-to-date network drivers if needed.

24.4.4 Adding the Operating System ISO

The next step is to add the Operating System `.iso` file to use for deploying Windows.

- Use the latest official Windows release from Microsoft as the `.iso` file.
- In the *Install ISO* section in the main WADS Console, click on the + button to upload the selected `.iso` file.
- Select the `.iso` file and give it a name.
- When uploaded, the `.iso` file is signed with the selected certificate:
- After the signing step has successfully completed, the `.iso` file is uploaded to the WADS Server:
- After the uploading step has successfully completed, the `.iso` file appears in the *Install iso* section in the main WADS Console:

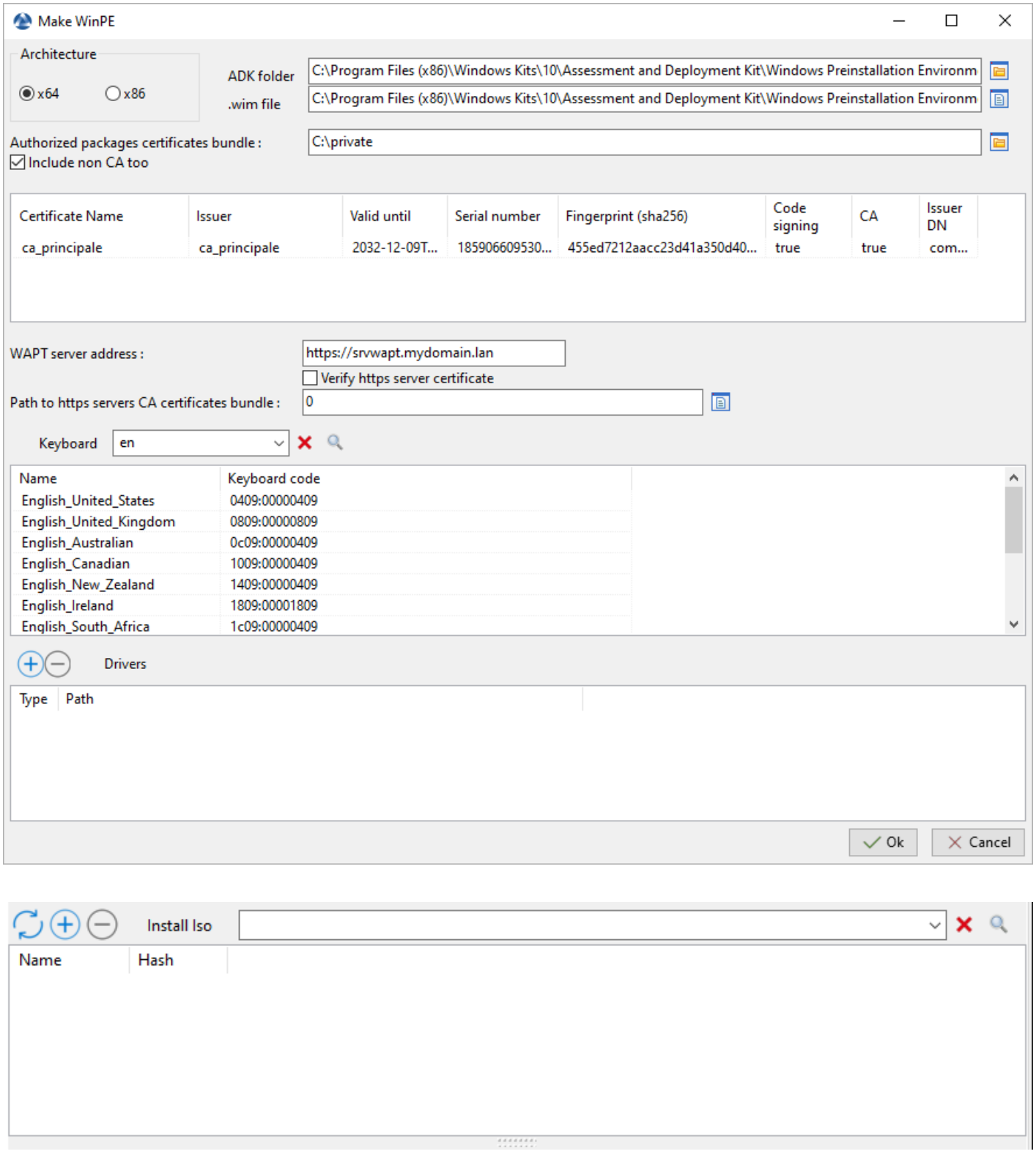


Fig. 7: ISO section of the WADS Console

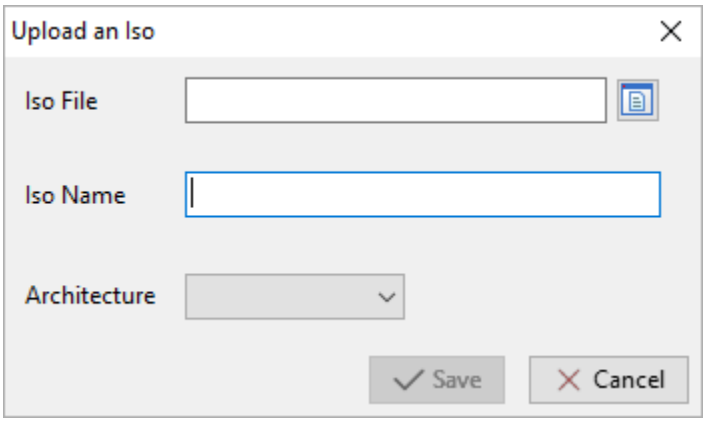


Fig. 8: Dialog box for selecting the ISO file to upload to the WADS Server

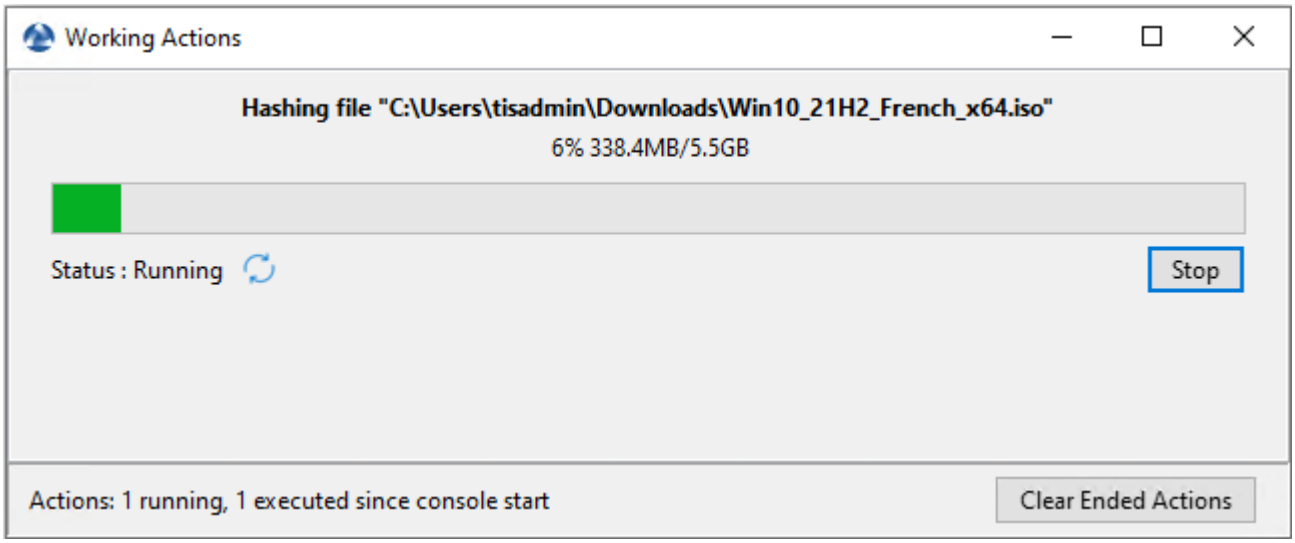


Fig. 9: Dialog box informing of the signing progression of the ISO file in the WADS Console

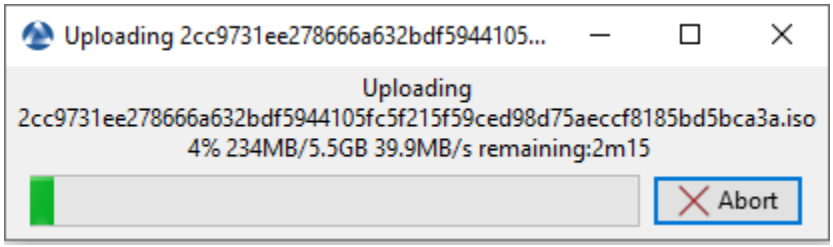


Fig. 10: Dialog box informing of the uploading progession of the ISO file in the WADS Console

Name	Hash
Windows	2cc9731ee278666a632bdf5944105fc5f215f59ced98d75aeccf8185bd5bca3a

Hint: It is possible to upload several `.iso` versions of Windows for different use cases.

24.4.5 Adding the Configuration answer file

The next step is to add the Configuration answer file that will be used to configure the deployment of the Windows Operating System.

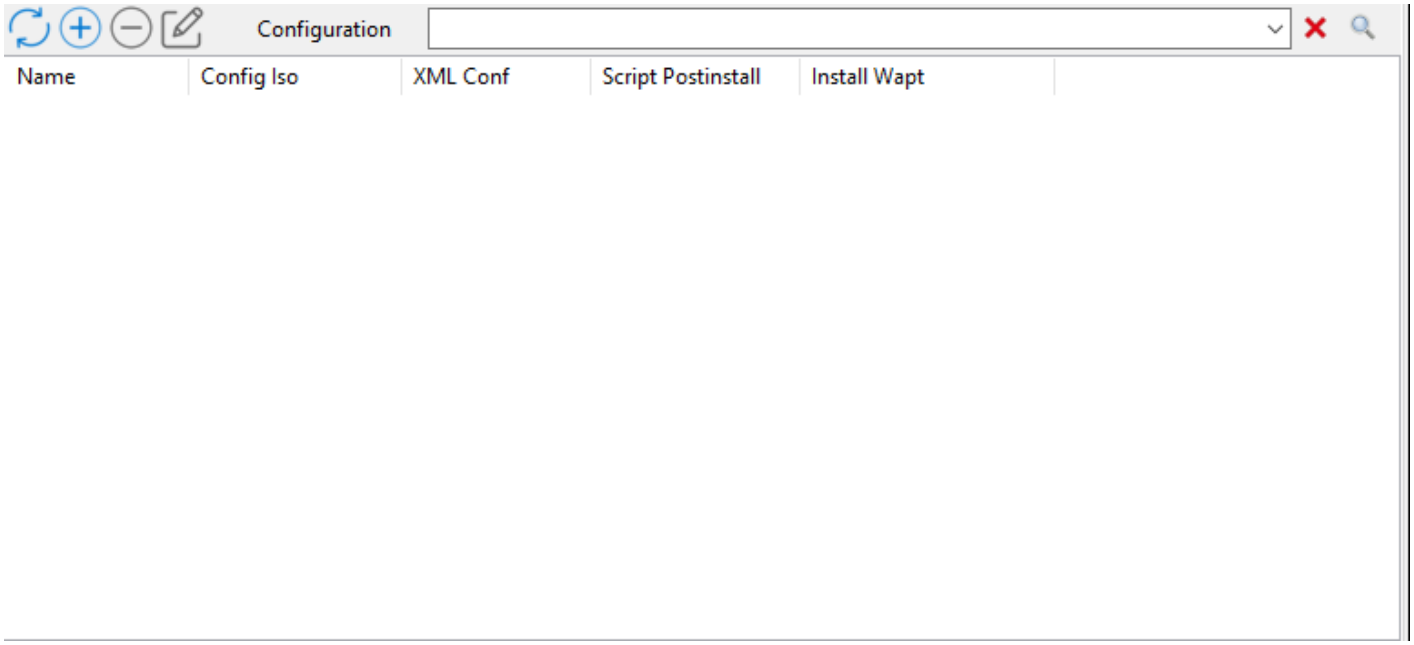


Fig. 11: Answer file section of the WADS Console

- In the *Configuration* section click on the + button to configure the answer file.

Table 2: Options for the answer file in the WADS Console

Options	Description
Config Name	Defines the name of the XML answer file.
ISO Name	Defines the `.iso` file to associate to the XML answer file.
For Windows	Defines whether you install a Windows OS or Linux if unchecked.
Install Wapt	Defines whether to install the **WAPT agent** after the installation of the Operating System.
Configuration file	Defines the XML answer files template to use for Windows or the configuration file for Linux.
Post install Script	Defines a `.bat` post-install script to be run after the installation of the Operating System.

- Insert into the *Config Name* field the name of the answer file.
- Select with the *Iso Name* dropdown the ISO file to association to the deployment configuration.
- Check or uncheck the *Install WAPT* checkbox to install the WAPT Agent by default.

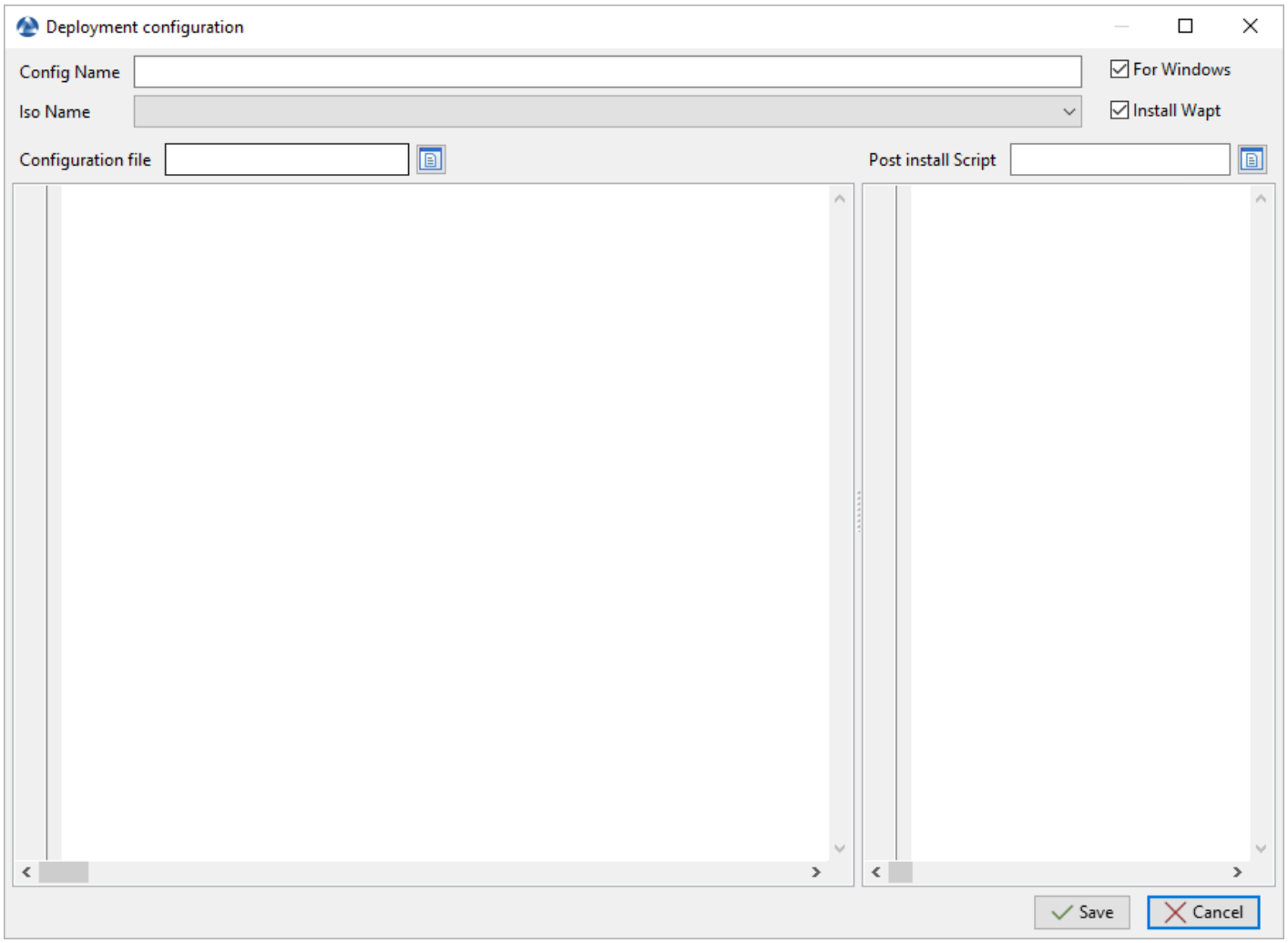


Fig. 12: Window for creating the answer configuration file in the WADS Console

- Check or uncheck the *For Windows* checkbox to install a Windows OS.
- Select the answer file template to associate to the deployment configuration with the *Configuration File* field if it's OS Windows else, select the configuration file for Linux.
- If necessary, set the post-install script in *Post install Script*, for example:

```
"C:\Program Files (x86)\wapt\wapt-get.exe" install tis-firefox-esr
```

- Click on the *Save* button to create the answer file.
- When done, the configuration appears in the *Configuration* section.

Name	Config Iso	XML Conf	Script Postinstall	Install Wapt
Windows 10	Windows	<!--********** ...		True

Fig. 13: Answer file added to the WADS Server in the WADS Console

Hint: It is possible to create several answer file configurations for different versions of Windows / Linux and for different use cases.

24.4.6 Joining the host to an Active Directory domain

You can use your own answer file with WADS but by default, WADS integrate *2* types of answer files for Windows:

- **Offline** to join a computer with the DirectAccess Offline Domain Join (Djoin) method
- **Online** to join a computer on the AD

Online method

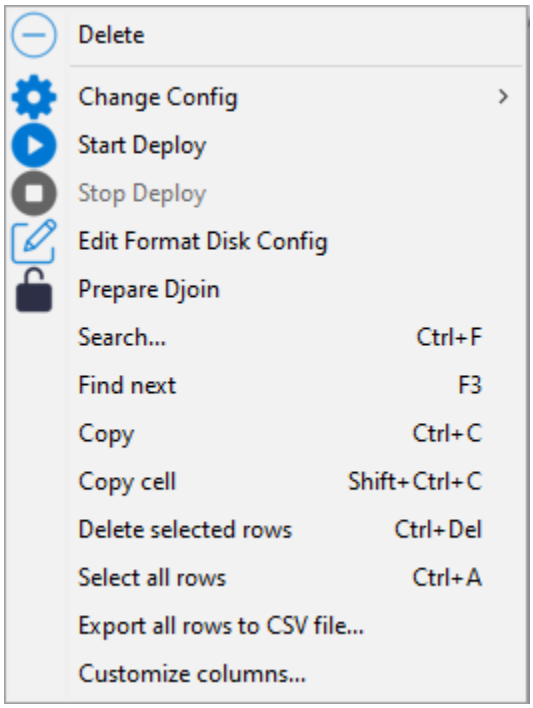
Update this part with your **join service account**, you can give a specific OU if you want. If not, just delete the line *MachineObjectOU*.

```
<Identification>
  <Credentials>
    <Domain>mydomain.lan</Domain>
    <Password>password</Password>
    <Username>wadsjoin</Username>
  </Credentials>
  <JoinDomain>mydomain.lan</JoinDomain>
  <MachineObjectOU>OU=MyOu,OU=MyParentOu,DC=MyDomain,DC=lan</MachineObjectOU>
</Identification>
```

Offline method

The offline method uses the Djoin method.

- Right-click on the host to open the menu list.



- Click on *Prepare Djoin*.
- Select the OU to which to attach the host (or define it manually) and click on *Save*.

You can check *Do not use current user* if your current user can not or must not join a computer to the domain. If checked, you have to give manually **Domain**, **Host OU**, **User** (just the sAMAccountName, not the UPN nor the DOMAIN\user) and **password**.

You can check *Overwrite the existing machine* in order to join anew a computer.

- The `Djoin` file is ready to be used to join the host as a member to the Active Directory domain.

24.4.7 Adding drivers

The next step is to add driver bundles that will be used during the deployment of the Windows Operating System.

- In the *Drivers* section click on the + button to add a driver pack to the WADS Server.

This window allows you to upload the driver bundles to associate to the Windows deployment.

Table 3: Options for the driver bundles in the WADS Console

Options	Description
Choose Dir	Defines the path to the folder containing the driver bundles.
Name	Defines the name of the driver bundle.

- Click on the *Save* button, the uploading of the driver bundles starts.
- When uploaded, the drivers pack appears in the *Drivers* section of the WADS Console.

It is possible to create several driver packs for different versions of Windows and for different use cases.

It is possible to use the `.cab` files from OEM (Original Equipement Manufacturers).

It is also possibe to export the drivers from an existing well functioning host using a **`Powershell`** command.

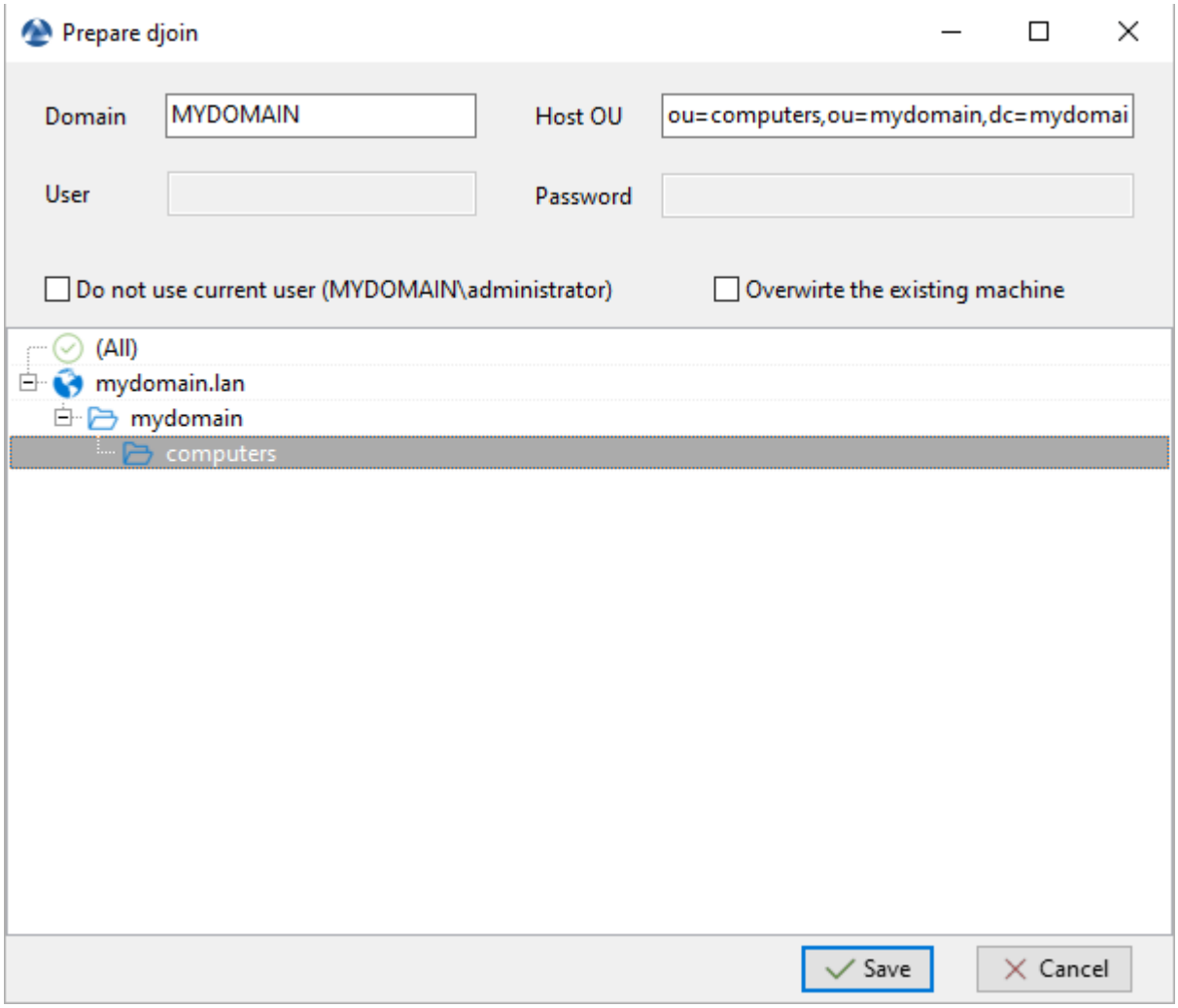


Fig. 14: Selecting the Organizational Unit to which to automatically attach the re-imaged host

Fig. 15: Drivers section of the WADS Console

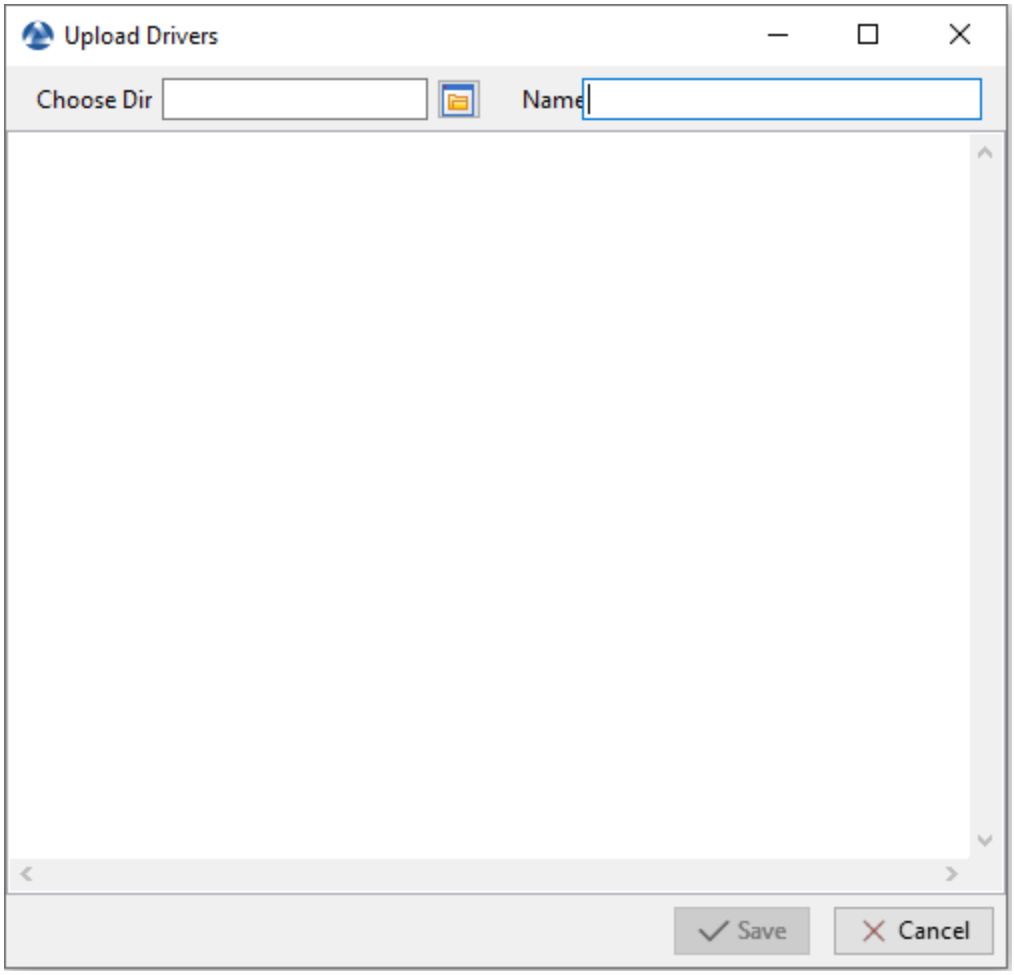


Fig. 16: Window for creating the driver bundles in the WADS Console

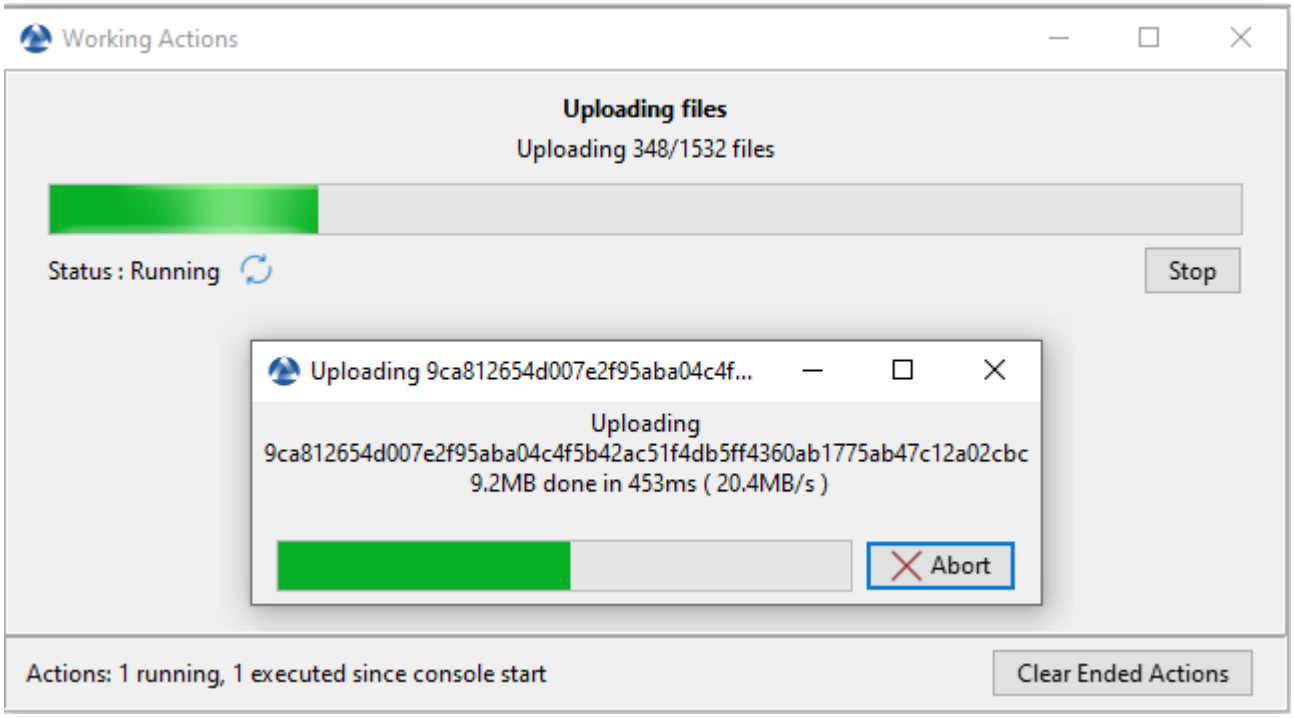


Fig. 17: Dialog box informing the uploading progression of the driver bundles in the WAPT Console

Fig. 18: The drivers pack has been uploaded to the WADS Server

```
Export-WindowsDriver -Online -Destination D:\Drivers
```

24.4.8 Booting the host to re-image with WADS

WADS allows **2** methods boot the host to re-image:

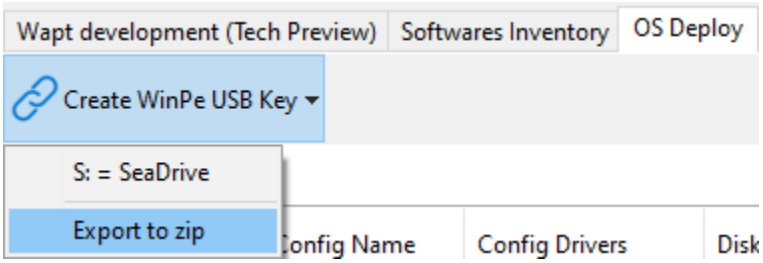
- *Locally with a USB key*.
- *Via LAN with a TFTP server*

Booting the host with a USB stick

Note: The USB key used **MUST** be FAT32 formatted and empty.

- Insert the USB stick in the WAPT adminsitration workstation and click on the *Create WinPE USB Key* button to start the process.
- Choose the keyboard layout. **This step is important because you will type in the hostname in WinPE using the keyboard layout chosen with this step**.
- Select the certificate with which to sign the USB stick files
- Click on the *Upload WinPE* to format the USB stick and copy the WinPE file.
- Boot to the computer's boot menu using the USB stick option and go to the *run the deployment* step.

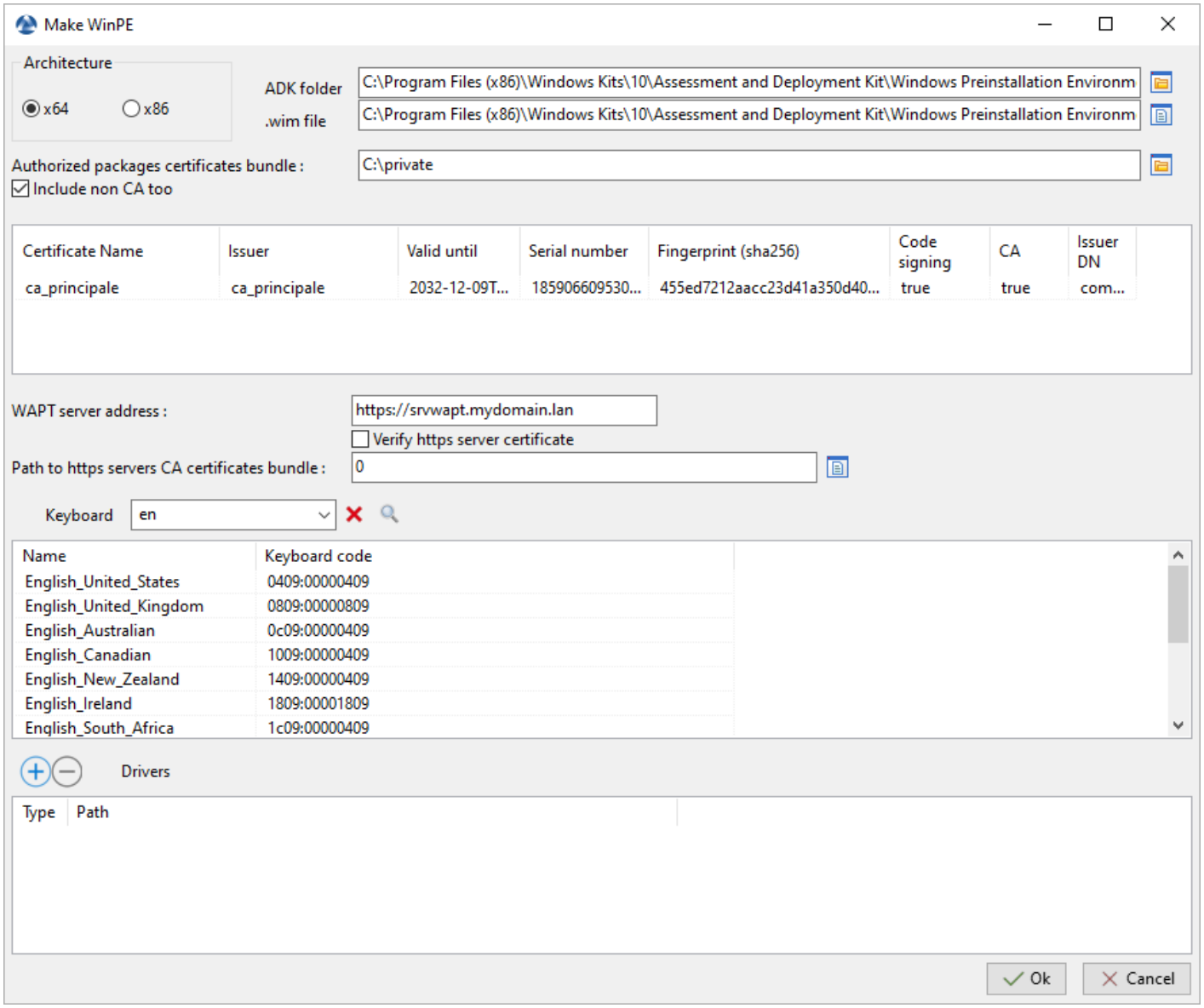
Note: You can *Export to zip* when you create a WinPE USB Key if you can not use a USB key and then burn it onto a CD / DVD instead.



Booting the host with the network

Booting from the LAN requires:

- A properly working *TFTP server*;
- A properly working *DHCP server*;
- Having port 69 open on the WAPT Server for inbound traffic, and having tftp conntrack enabled on intermediate firewalls if you have firewalls between the server and the client computer.
- Boot to the computer's boot menu using the LAN option and go to the *run the deployment* step.


Make WinPE
Architecture
x64
x86
ADK folder
C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation Environm
.wim file
C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Windows Preinstallation Environm
Authorized packages certificates bundle :
C:\private
Include non CA too
Certificate Name
Issuer
Valid until
Serial number
Fingerprint (sha256)
Code signing
CA
Issuer DN
ca_principale
ca_principale
2032-12-09T...
185906609530...
455ed7212aacc23d41a350d40...
true
true
com...
WAPT server address :
https://srvwapt.mydomain.lan
Verify https server certificate
Path to https servers CA certificates bundle :
0
Keyboard
en
Name
Keyboard code
English_United_States
0409:00000409
English_United_Kingdom
0809:00000809
English_Australian
0c09:00000409
English_Canadian
1009:00000409
English_New_Zealand
1409:00000409
English_Ireland
1809:00001809
English_South_Africa
1c09:00000409
Drivers
Type
Path
Ok
Cancel

24.4.9 Deploying the Windows image

There are **3** choices when booting with iPXE:

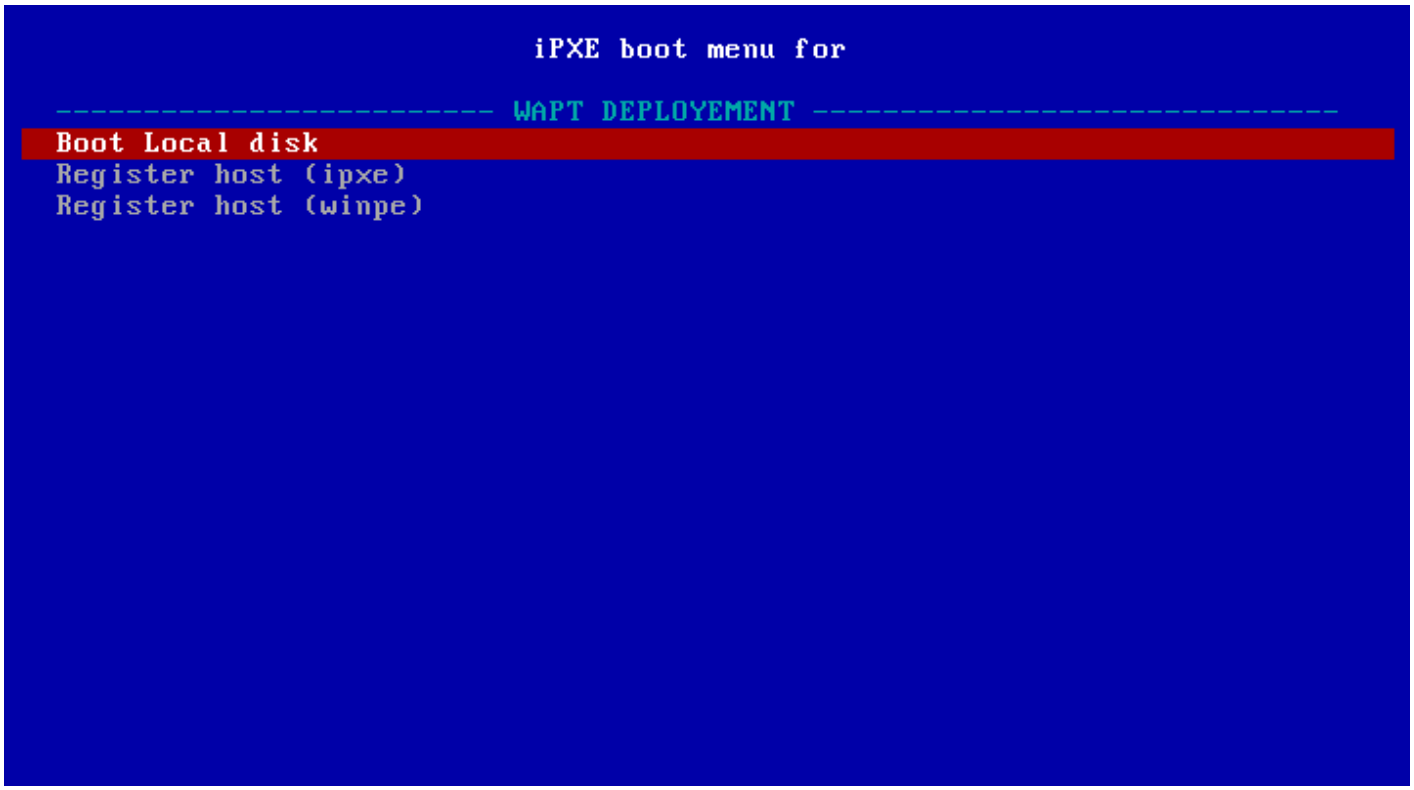


Fig. 19: iPXE boot menu window

- *Boot Local disk* for starting normally from local storage;
- *Register host (ipxe)* to register the host with the WADS Server using the *iPXE method*;
- *Register host (winpe)* to register the host with the WADS Server using the *WinPE method*.

iPXE boot

- If choosing *Register host (ipxe)*, define a hostname:

> **Warning:** The keybord is qwerty

- Refresh the WADS Console with `F5`, the host appears in the *OS Deploy* tab.

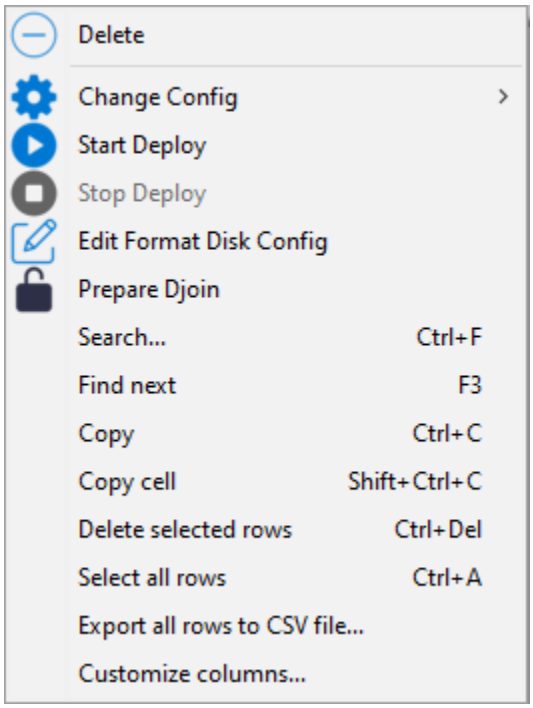
At this time, the *Waiting to Deploy* status of the host is `False`.

- Right click on the host to open the menu list.

Fig. 20: Text terminal window requesting a hostname when registering using the iPXE method

Hostname	Mac Addresses	Status	Waiting to Deploy
documentation	08:00:27:19:0D:4B		False

Fig. 21: Host waiting to be deployed



- Go to *Change Config* and select *a XML answer file*.
- Click on *Start Deploy*, the *Waiting to Deploy* status of the host switches to `True`.

Hostname	Mac Addresses	Status	Waiting to Deploy
documentation	08:00:27:19:0D:4B		True

- Reboot the host to the same boot option as before (USB or LAN), Windows will start to install.
- When the installation has completed, the *OS Deploy* tab, the status switches to `Done`.

WinPE

- If choosing *Register host (winpe)*, define a hostname:

The keybord is in the same layout as the one set during the *WinPE* step of this documentation.

- Refresh the WADS Console with `F5`, the host appears in the *OS Deploy* tab.

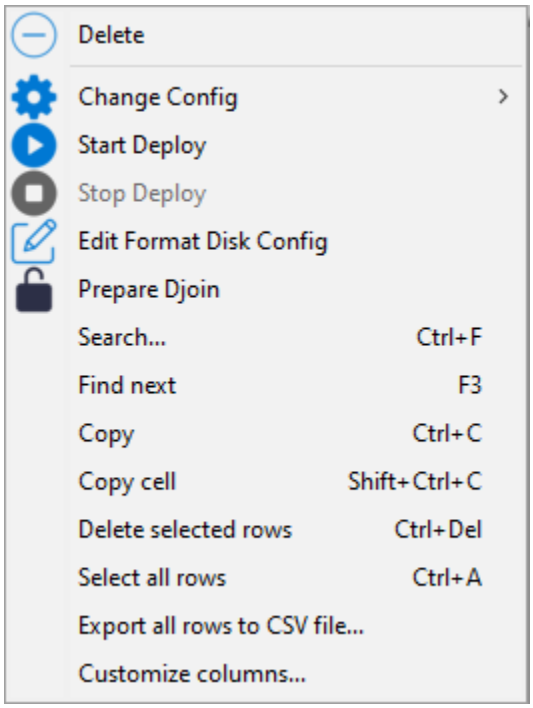
At this time, the *Waiting to Deploy* status of the host is `False`.

- Right click on the host to open the menu list.

Fig. 22: Text terminal window requesting a hostname when registering using the WinPE method

Hostname	Mac Addresses	Status	Waiting to Deploy
documentation	08:00:27:19:0D:4B		False

Fig. 23: Host waiting to be deployed



- Go to *Change Config* and select *a XML answer file*.
- Click on *Start Deploy*, the *Waiting to Deploy* status of the host switches to `True`.

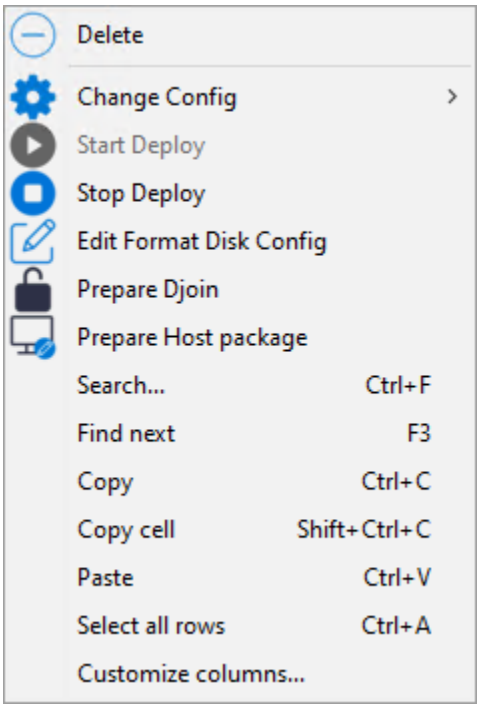
Hostname	Mac Addresses	Status	Waiting to Deploy
documentation	08:00:27:19:0D:4B		True

- Reboot the host to the same boot option as before (USB or LAN), Windows will start to install.
- When the installation has completed, the *OS Deploy* tab, the status switches to `Done`.

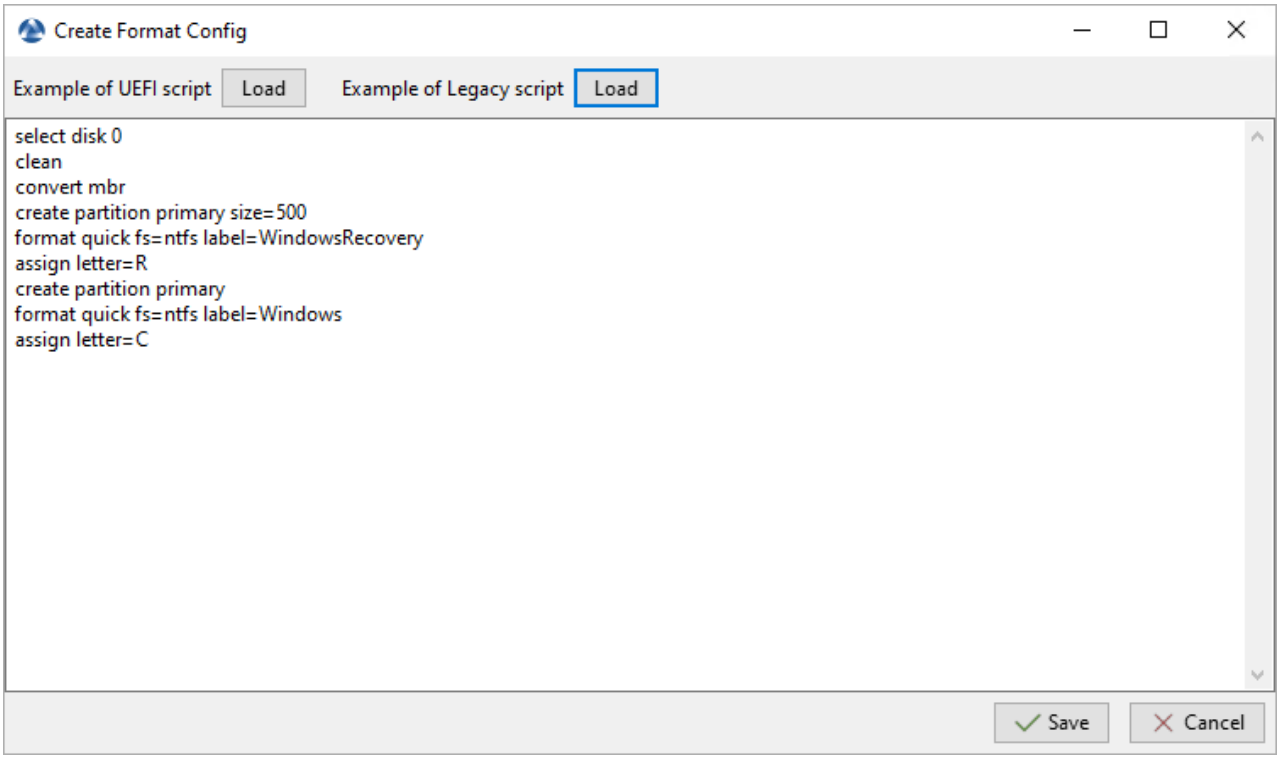
24.4.10 Format host disk

When your host is ready to be redeployed, if necessary, you can format its disk using the UEFI or the Legacy method.

To do so, right-click on host then *Edit Format Disk Config*.



Then you can choose either the UEFI or the Legacy script and customize the disk format configuration. Here is an example with the Legacy script:



CHAPTER

TWENTYFIVE

USING THE REPORTING FUNCTIONS IN WAPT

25.1 Working principle

https://youtu.be/UjBfelmJyKo

WAPT **Enterprise** offers advanced reporting capabilities.

Indeed, you are best to know what you want in your reports.

With WAPT you can write your own SQL queries in the WAPT Console, or you can download ready-to-run queries from Internet.

The database structure diagram is available here `wapt_db_data_structure.svg`.

25.2 WAPT Query Designer

The query designer allows to edit and run SQL queries on the WAPT Server PostgreSQL database.

Note: The PostgreSQL database is set to **Read-Only** mode, so queries run from the Report Designer that attempt to update, delete or insert data will fail.

You can import SQL queries from the Tranquil IT store by clicking on *Reporting* → *Import queries…* → *From Url* or on *Reporting* → *Import queries…* → *From File*.

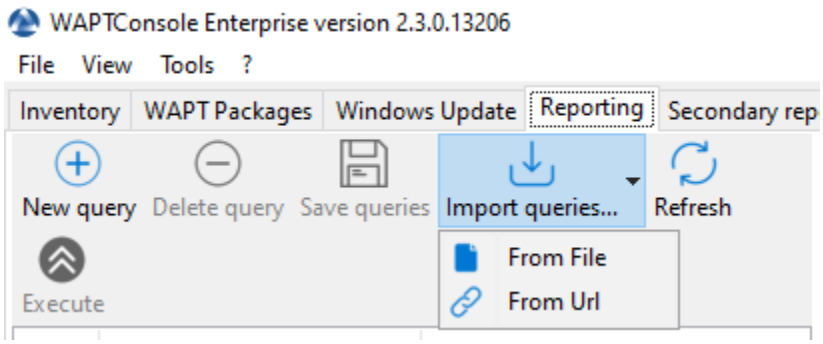


Fig. 1: Importing a SQL query report in the WAPT Console

If you choose to import a query from the Tranquil IT store, select one or several queries, then click on *Save selected queries*. The new queries will appear in the WAPT Console.

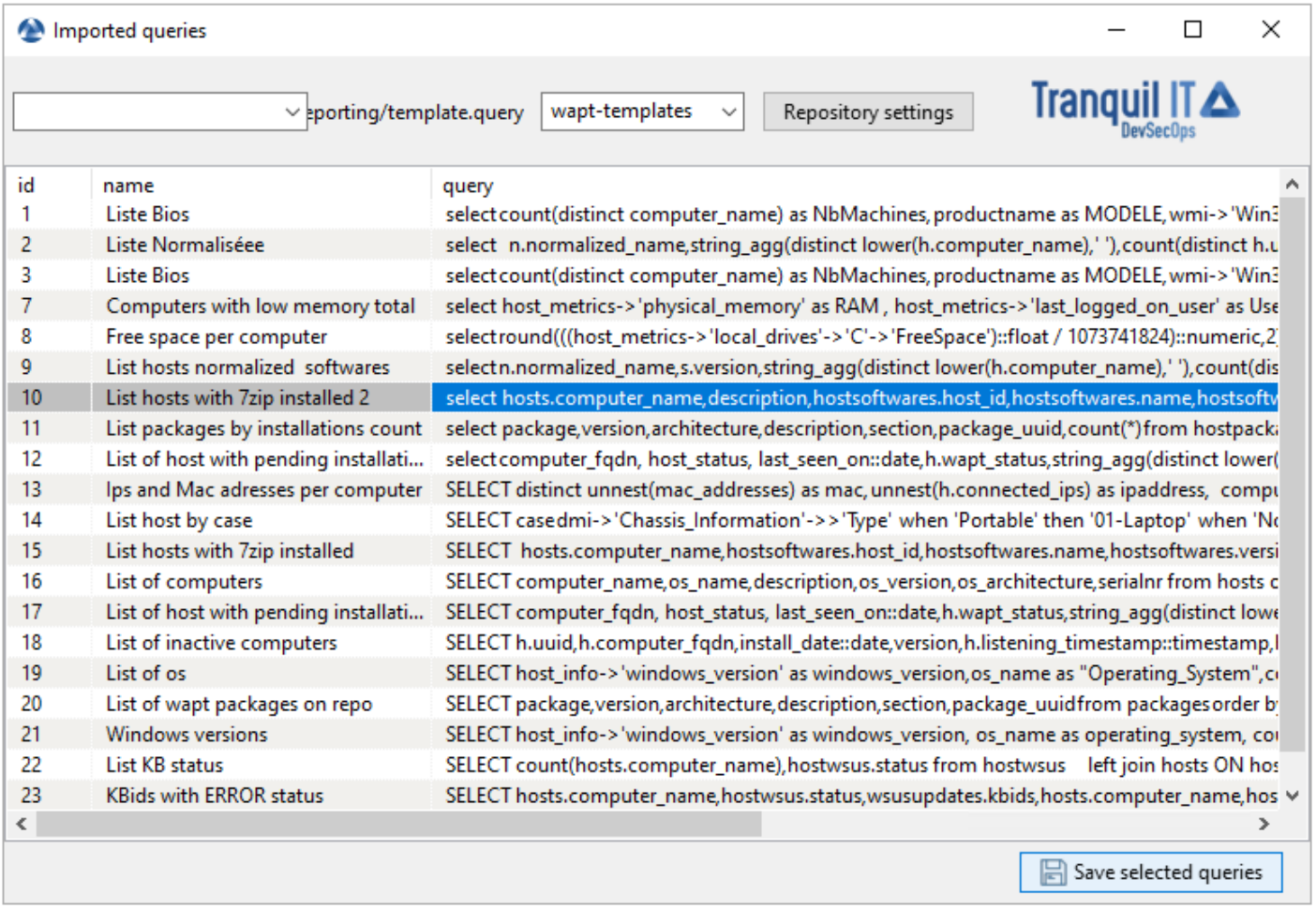


Fig. 2: Importing a SQL query report in the WAPT Console from a URL

If you import a query from a file, select your query from the file explorer. The query can be either a `.query` or a `.json` formated file.

Note: Irrelevant of the method you use to import queries, remember to click on the button *Save Queries* to save the imported queries.

To create a new SQL report, click on *Reporting* → *Design Mode* → *New query*.

Hint:

- To rename a query, select the query and press the `F2` key.
- In the top banner, you can write your SQL query.

To edit / modify / save your reports:

- The *Refresh* button is used to reload queries saved on the WAPT Server, for example, if a colleague has just edited a new query.

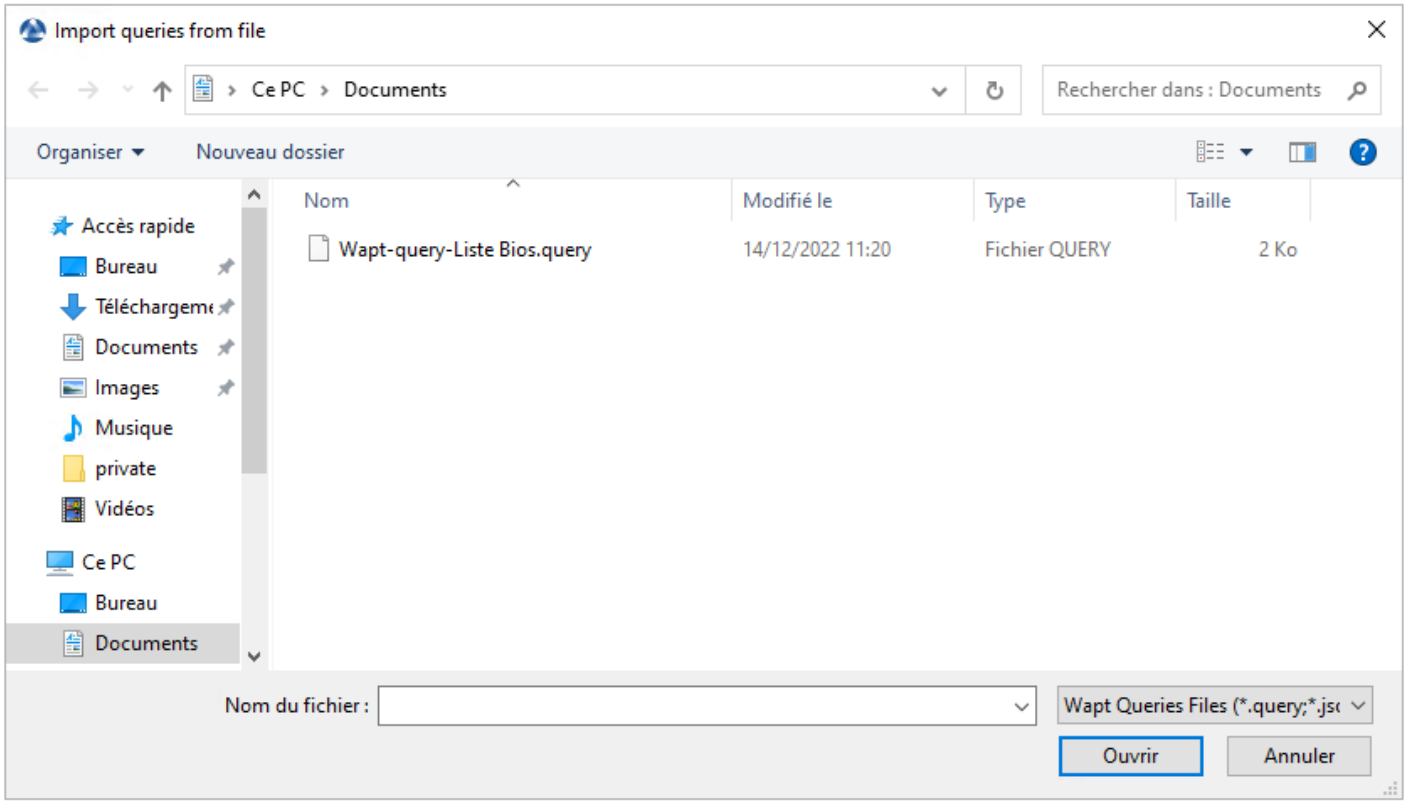


Fig. 3: Importing a SQL query report in the WAPT Console from a file

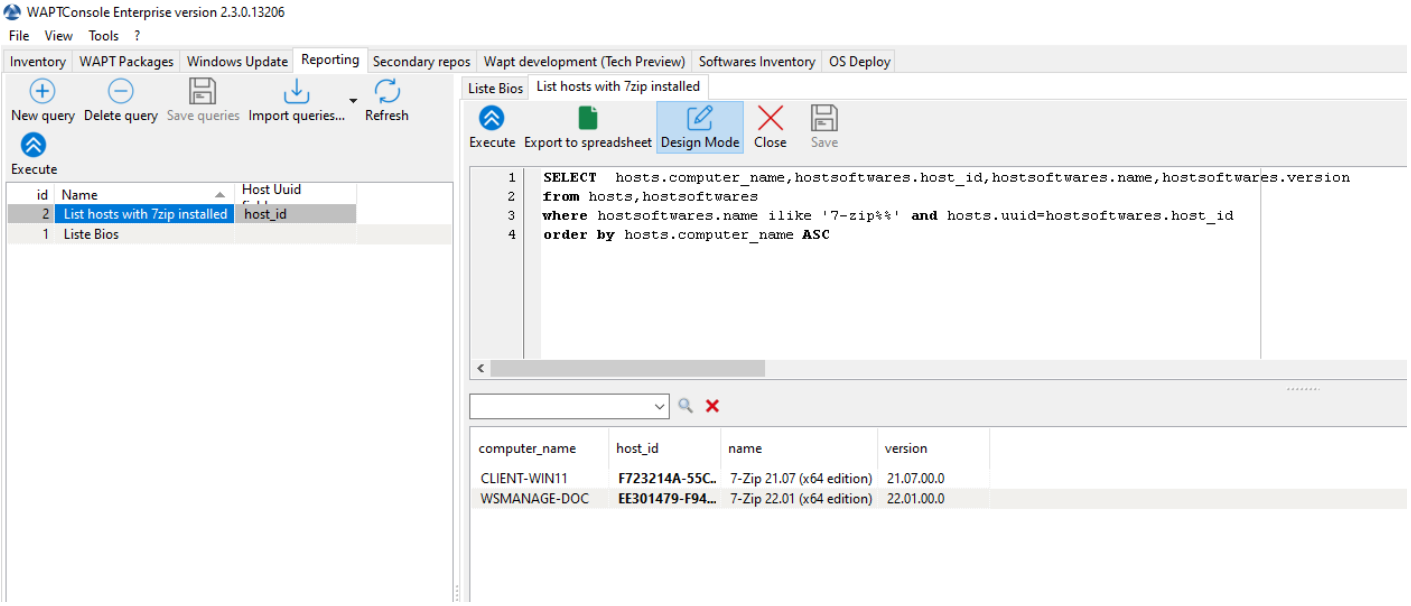


Fig. 4: Designing a SQL query report in the WAPT Console

- The *New query* button adds a new blank query to the list.
- The *Delete query* button deletes the selected query from the WAPT Server.
- The *Save queries* button saves the query on the WAPT Server.
- The *Execute* button executes the selected query.
- The *Export to spreadsheet* button exports the result of your query to a spreadsheet. The button is enabled only in edit mode.
- The *Duplicate* button duplicates an existing query to avoid writing a SQL query from scratch. The button is enabled only in edit mode.

You have several options available when you right click on a query.

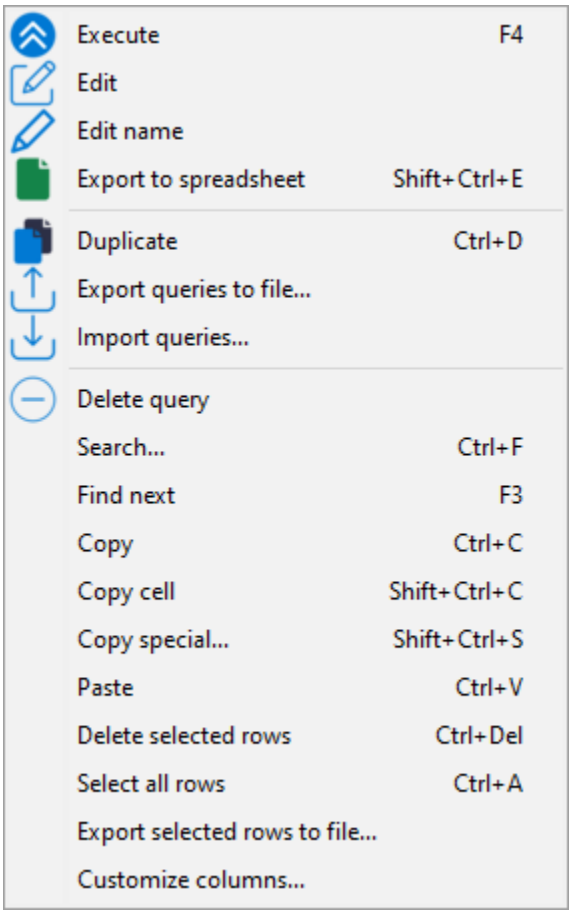


Fig. 5: Options available for a SQL query report in the WAPT Console

Table 1: List of actions available on a selection of query on WAPT Console

Name	Description
Execute	Executes the SQL query.
Edit	Edits the SQL query.
Edit Name	Edits the name of the SQL query.
Export to spreadsheet	Exports the result of the SQL query to a `csv` formatted file.
Duplicate	Duplicates the SQL query.
Export queries to file...	Exports the selected SQL queries to a file. This method allows to share or backup the SQL queries.
Import queries...	Imports queries from a file.
Delete queries	Deletes the selected queries.

Note:

- SQL queries are saved in the PostgreSQL WAPT database.
- Using `CTRL+space` allows to build queries more effectively as it will auto-complete some fields.

25.3 Using queries as a filter in the inventory tab

You can create a filter for use in the *Inventory* tab based on a SQL query. In this exemple, we use a query which lists hosts onto which the 7zip software has been installed.

Hint: This method is powerful because it allows to seach search for software titles that have not been installed using WAPT.

On the query result, right-click on *host_id*, which is a unique identifier, select this entry and click on the *Choose as Host UUID* button.

When done, save the query by pressing the *Save query* button then go to the *Main Inventory* tab.

Then, enable the *Advanced Search* panel and select the query in the drop-down field *Filter hosts on SQL query*.

You will then see a host list based on the selected query.

25.4 Query examples

25.4.1 Computers query

Counting hosts

```
select count(*) as "number_of_hosts" from hosts
```

Listing computers

```
select
computer_name,
os_name,
```

(continues on next page)

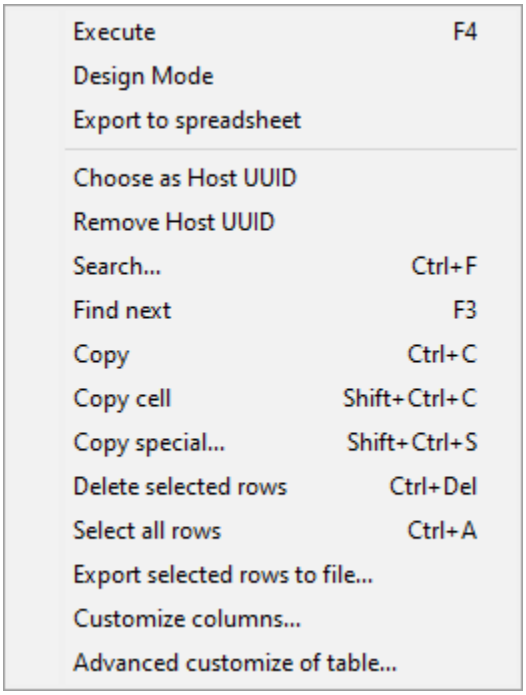


Fig. 6: Querying on host UUID to create a dynamic view in the inventory tab

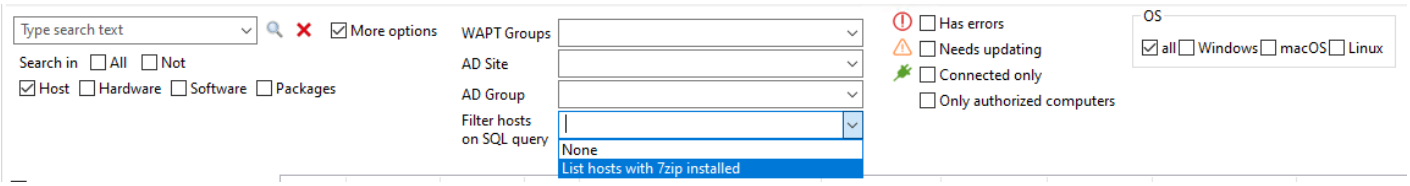


Fig. 7: Filtering the host inventory using a SQL query

(continued from previous page)

```
os_version,
os_architecture,
serialnr
from hosts
order by 4,3,1
```

Listing 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
```

Listing 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
```

Listing operating systems

```
select host_info->'windows_version' as windows_version,
os_name as "Operating_System",
count(os_name) as "number_of_hosts"
from hosts
group by 1,2
```

Listing hosts not seen in a while

```
select
h.uuid,
h.computer_fqdn,
install_date::date,
version,
h.last_seen_on::timestamp,
h.connected_users from hostsoftwares s
left join hosts h on h.uuid=s.host_id
where s.key='WAPT_is1'
and h.last_seen_on<'20190115'
```

Filtering hosts by chassis types

```
select case
dmi->'Chassis_Information'->>'Type'
```

(continues on next page)

(continued from previous page)

```
when 'Portable' then '01-Laptop'
when 'Notebook' then '01-Laptop'
when 'Laptop' then '01-Laptop'
when 'Desktop' then '02-Desktop'
when 'Tower' then '02-Desktop'
when 'Mini Tower' then '02-Desktop'
else '99-'||(dmi->'Chassis_Information'->>'Type')
end as type_chassis,
string_agg(distinct coalesce(manufacturer,'?') ||' '|| coalesce(productname,''),', '),
count(*) as "number_of_hosts" from hosts
group by 1
```

Listing of 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
```

25.4.2 WAPT query

Listing WAPT packages in the WAPT Server repository

```
select
package,
version,
architecture,
description,
section,
package_uuid,
count(*)
from packages
group by 1,2,3,4,5,6
```

Listing 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
```

(continues on next page)

(continued from previous page)

```
and host_status!='OK')
group by 1,2,3,4
```

25.4.3 Packages query

Listing 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
```

25.4.4 Software query

Listing WAPT Discovery 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%%Discovery%%' or name ilike 'WAPT %%')
```

Listing 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
```

Listing hosts with their software

```
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
```

Listing normalized software

```
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 forum with an explanation of what your query does, ideally with a screen capture or a table showing a sample of your query result.

25.5 Normalizing software names

Sometimes, the version of the software or its architecture are an integral part of the software name. When the software titles register with the WAPT Server inventory, they appear as different software titles whereas they are just one software for us humans.

To solve this problem, the name of software titles can be standardized in WAPT. Go to the *Softwares Inventory* tab.

- Click *Normalize Software Names* in the *Tools* menu.
- Select the software titles whose names to standardize, for example, all different versions of Adobe Flash Player.
- On the column *normalized*, press `F2` to assign a standardized name to the selected software titles. Finally press `Enter` to save the changes.

Note:

- To select several software titles, select them with the `shift-up/down` key combination.

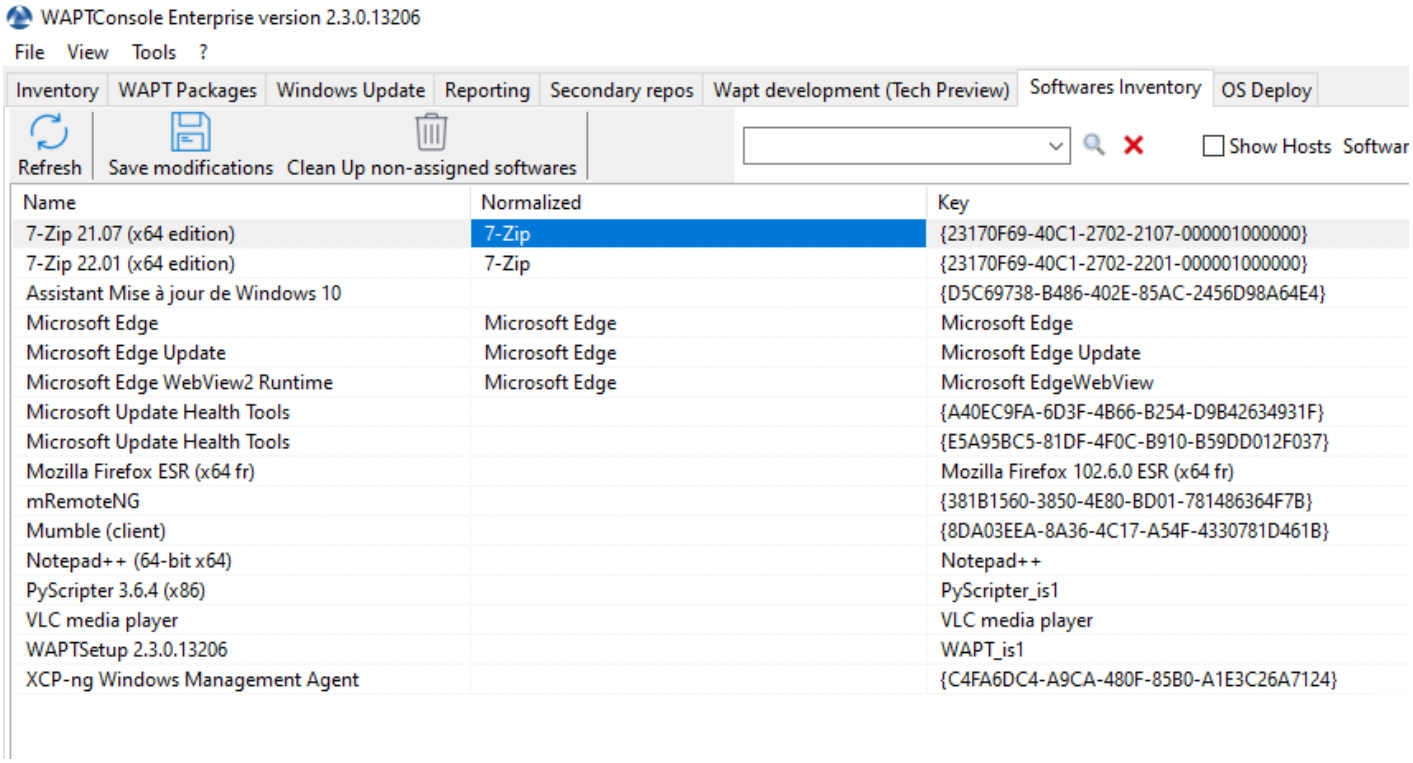


Fig. 8: Normalizing the name of software titles in the WAPT Console

- You can also indicate a software like *windows update* or *banned* (press `spacebar` in the corresponding column).

- Press on the *Save Modifications* button to upload the changes onto the WAPT Server.

You can now run queries using this standardized name.

Hint: The *Show host* checkbox allows to see the software titles that are installed on the hosts.

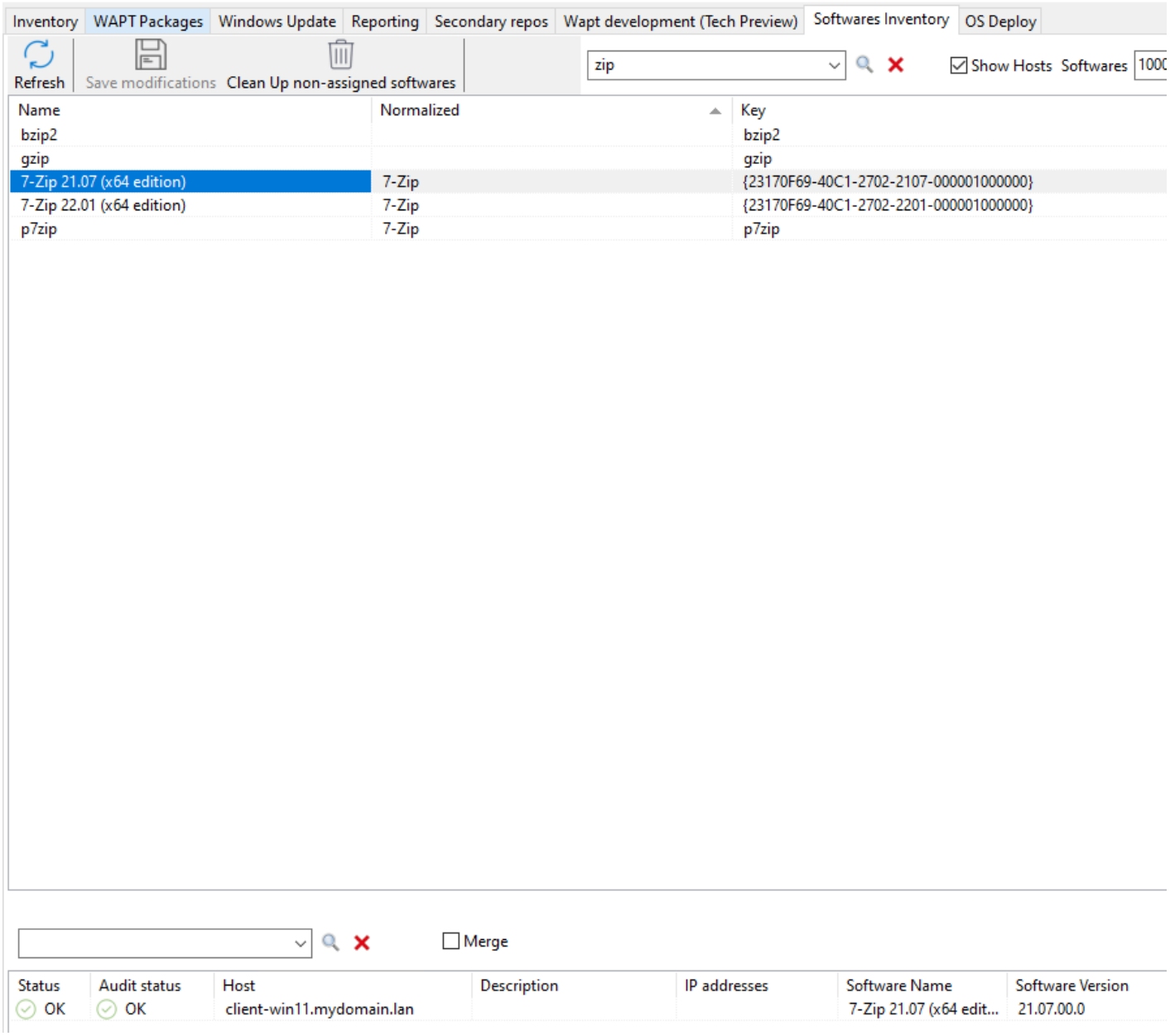


Fig. 9: Showing hosts with selected software in the Software Inventory tab

25.5.1 Using normalized software titles as a filter in the inventory tab

You can create a filter in the *Main Inventory* tab that uses the normalized software title names. To do so, normalize the software title names in *Software Inventory* then select one or several hosts. In the *Software inventory* tab of the selected hosts, drag and drop the software in the inventory list, it will create a view.

Hint: This method is powerful because it allows to seach search for software titles that have not been installed using WAPT.

Do not forget to normalize software titles beforehand.

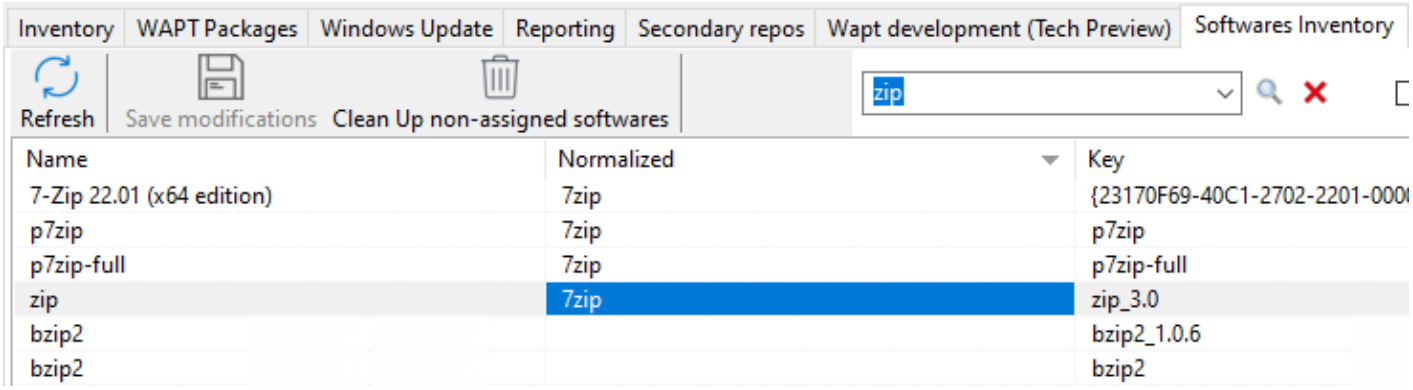


Fig. 10: Defining a normalized software title name

Fig. 11: Adding a normalized software title name to the software inventory for the selected hosts

You will see a host list sorted according to the normalized software title name.

25.6 Connecting to the WAPT database using a PostgreSQL client

You can connect a PostgreSQL client to the WAPT database if you prefer to use a PostgreSQL client.

To do so, you will have to change some configuration files on your WAPT Server.

- First, find the version of your PostgreSQL database.

```
ps -ef | grep -i sql
postgres   512     1  0 Jan05 ?        00:00:24 /usr/lib/postgresql/12/bin/postgres -D /var/lib/
↪postgresql/12/main -c config_file=/etc/postgresql/12/main/postgresql.conf
```

- Modify `pg_hba.conf` of the PostgreSQL version in use. In `/etc/postgresql/12/main/pg_hba.conf` for Debian and `/var/lib/pgsql/12/data/pg_hba.conf` for RedHat and derivatives, add the IP address of the PostgreSQL client under `# IPv4 local connections section`.

```
host    wapt            all             192.168.0.65/32             md5
```

(continues on next page)

(continued from previous page)

```
where 192.168.0.65 is your IP address that is authorized
to connect to the WAPT database.
```

- Allow PostgreSQL to listen on every interface in `/etc/postgresql/12/main/postgresql.conf` for Debian and `/var/lib/pgsql/12/data/postgresql.conf` for RedHat and derivatives, section **Connection Settings**.

```
listen_addresses = '*'
```

- Restart the service for your PostgreSQL version.

```
systemctl restart postgresql@12-main.service
```

- Connect to PostgreSQL on the WAPT Server.

```
sudo -u postgres psql template1
```

- Then give a password to the *wapt* user.

```
template1=# ALTER USER wapt WITH PASSWORD 'PASSWORD';
```

CHAPTER

TWENTYSIX

USING AUDIT DATA INTO PLUGINS FOR WAPT PACKAGE COMPLIANCE AND FOR EXTERNAL SERVICES

26.1 Displaying host audit data in the WAPT Console

You can manage audit output and display the audit result if you activate the option in the *View → Display Preferences* Tab. Check the *Show host audit data tab* to see the tab *Audit Data* on each client.

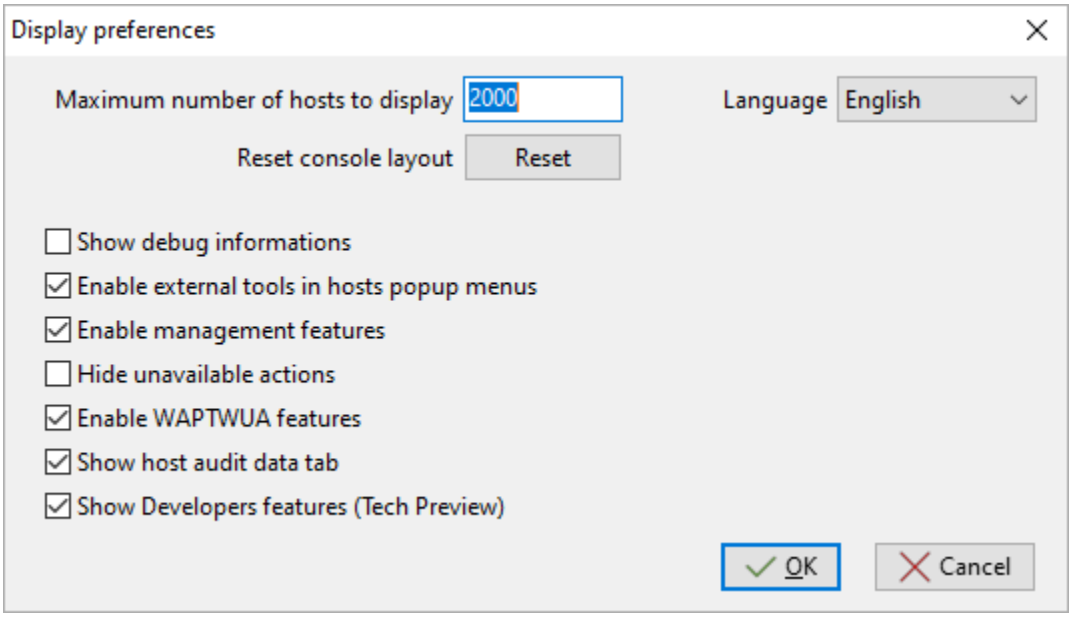


Fig. 1: Window showing the advanced preferences

To use audits in WAPT packages, visit *this page to manage audit_data*.

26.1.1 Displaying encrypted data with a certificate in the audit data tab

With audit function, it is possible to encrypt sensitive data coming from remote hosts; it will be possible to read the encrypted sensitive with a certificate installed on the WAPT Administrator's host. This way, the WAPT Server may store sensitive inventory data without the WAPT Server becoming a sensitive asset.

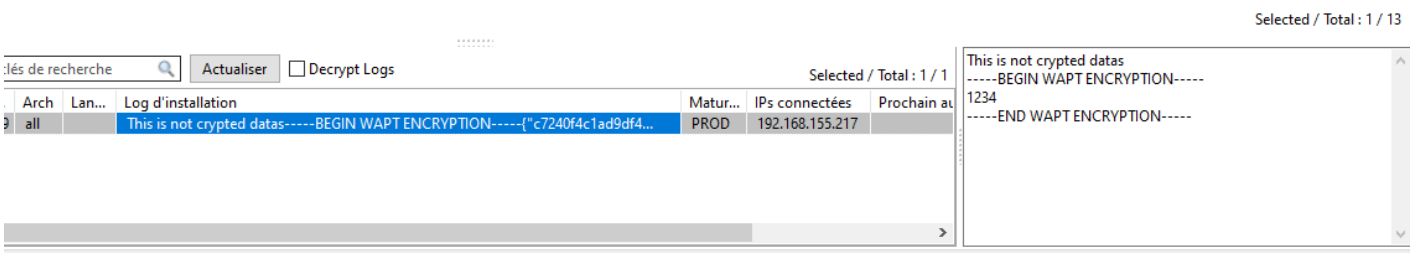
This method is particularly useful for example for securely managing LAPS (Local Administrator Password Service) random passwords in WAPT.

In `setup.py`, you can use a function to encrypt data with a certificate. If you have the private key matching the certificate that was used to encrypt the data, the data will be decrypted and it will appear in a readable form.

Here is an example of code:

```
# -*- coding: utf-8 -*-
from setuphelpers import *
from waptcrypto import print_encrypted_data

def audit():
  randompassword = '1234'
  print_encrypted_data(randompassword, glob.glob('*.crt'))
```

This code will encrypt the password *1234* with all certificates present on the host that is used to manage WAPT. From the WAPT Console, you will see in the *audit_data* tab the crypted version and you can decipher the data with your private key associated to the public certificate that was used to encrypt the data.



26.2 Synchronizing WAPT inventories to GLPI

26.2.1 Working principle

WAPT Enterprise offers synchronization between the inventories of your hosts and GLPI ITSM Software.

The method automatically synchronizes changes on your IT infrastructure with the GLPI server.

WAPT can synchronize with GLPI 10 using the native JSON API. WAPT can synchronize with GLPI version 9.x using the **FusionInventory** plugin with XML format.

> **Attention:** GPLI on WAPT does not work with Kerberos authentification for GLPI.
>
> If you use Kerberos for GLPI, exclude `glpi/plugins/fusioninventory/` from the **`Nginx`** authentification.

26.2.2 Installing the required dependencies for GLPI 9.x

In order to receive inventories on the GLPI server, the **FusionInventory** plugin will need to be installed on the GLPI server. This is not required for GLPI 10 which has its own native JSON API.

Note: You can follow this guide to install FusionInventory.

After installing **FusionInventory** on the GLPI server, an **endpoint** needs to be configured on the WAPT Server to send the inventories to the GLPI server:

```
http:/glpi.mydomain.lan/glpi/plugins/fusioninventory/
```

26.2.3 Configuring WAPTAgent and sync package

Install and configure the WAPT Agent on the computer that will run the synchronization. The WAPTAgent is installed by default on the WAPTServer, it just need to be configured.

To configure the WAPTAgent, please refer to the corresponding documentation.

Then you need to install the GLPI sync package:

- for GLPI 9.x, you need to install the package tis-glpi-plugin-export-to-glpi9
- for GLPI 10.x, you need to install the package tis-glpi-plugin-export-to-glpi10

You need to configure an audit schedule on the agent

```
[global]
...
waptaudit_task_period=120m
...
```

With the chosen package, it will create two ini file in your $WAPT_INSTALL_DIR/private (linux : `/opt/wapt/private`, windows : `C:Program Files (x86)waptprivate`). Connect to the host and modify `glpi.ini` and `wapt_api.ini` files.

- For GLPI9:

```
[glpi]
username = glpi
password = xxxxxxx
url = https://glpi.xx.xxxxx.xx/plugins/fusioninventory/
```

- For GLPI10:

```
[glpi]
username = glpi
password = xxxxxxx
url = https://glpi.xx.xxxxx.xx/front/inventory.php
```

For GLPI10, please also ensure inventory is enabled.

For both GLPI9 and GLPI10:

```
[wapt]
username = waptregister
password = waptregister2023!
url = https://srvwapt.ad.tranquil.it
```

To test the current configuration, you can trigger an audit

```
wapt-get audit tis-glpi-plugin-export-to-glpi9
# or
wapt-get audit tis-glpi-plugin-export-to-glpi10
```

26.2.4 Current items sent by WAPT to the GLPI server

Table 1: Description of items

Value	Sent	Not sent
Computer name	✅	
User name	✅	
Description	✅	
OS name	✅	
OS version	✅	
Language	✅	
CPU	✅	
Memory	✅	
Battery	✅	
Chassis type	✅	
Physical or virtual	✅	
Network card configuration	✅	
Printer list and properties	✅	
Installed software[1]	✅	
Network drives	✅	
Environment variables[2]	✅	
Display screens references	✅	
Mouse and keyboard references		❌
Controllers card references (except graphic card)		❌
Antivirus version		❌
Firewall state		❌
Local group list		❌
Memory bank list and state		❌
USB ports list and connected devices		❌
Printer status		❌
Card readers		❌
System wide Appx list		❌

[1] Not including system wide Appx install
[2] Currently both system and system-wide user environment variables are included.

26.2.5 Possible errors in reported inventory on the GLPI server

Inventories uploaded by the WAPT Server to the GLPI server may be incomplete or may have errors when compared to inventories uploaded directly by the FusionInventory agent deployed on hosts. One reason is that WAPT aims to report only the most important values.

If you feel that important items are missing or are reported in a wrong way, please report the issue to the Tranquil IT dev team.

To report the issue, you will need to send 2 `.xml` files.

1. First, install the FusionInventory agent on the computer on which you are observing a missing or wrongly reported inventory item.
2. Run the FusionInventory agent and extract the report into a `.xml` file.

Windows

```
"C:\Program Files\FusionInventory-Agent\fusioninventory-inventory" > %TEMP%\inventory.xml
```

Linux

```
fusioninventory-inventory > /tmp/inventory.xml
```

MAC

```
fusioninventory-inventory > /tmp/inventory.xml
```

1. Set the debug directory in the *waptserver.ini*.

   ```
   glpi_inventory_debug_directory = /tmp/glpi
   ```

4. Restart the WAPT Server
5. Retrieve the `/tmp/glpi/UUID.xml` file from the WAPT Server, the UUID being the identifier of the host.
6. Send the 2 files to the Tranquil IT dev team.

26.3 Synchronizing WAPT inventories to Cyberwatch for security breaches

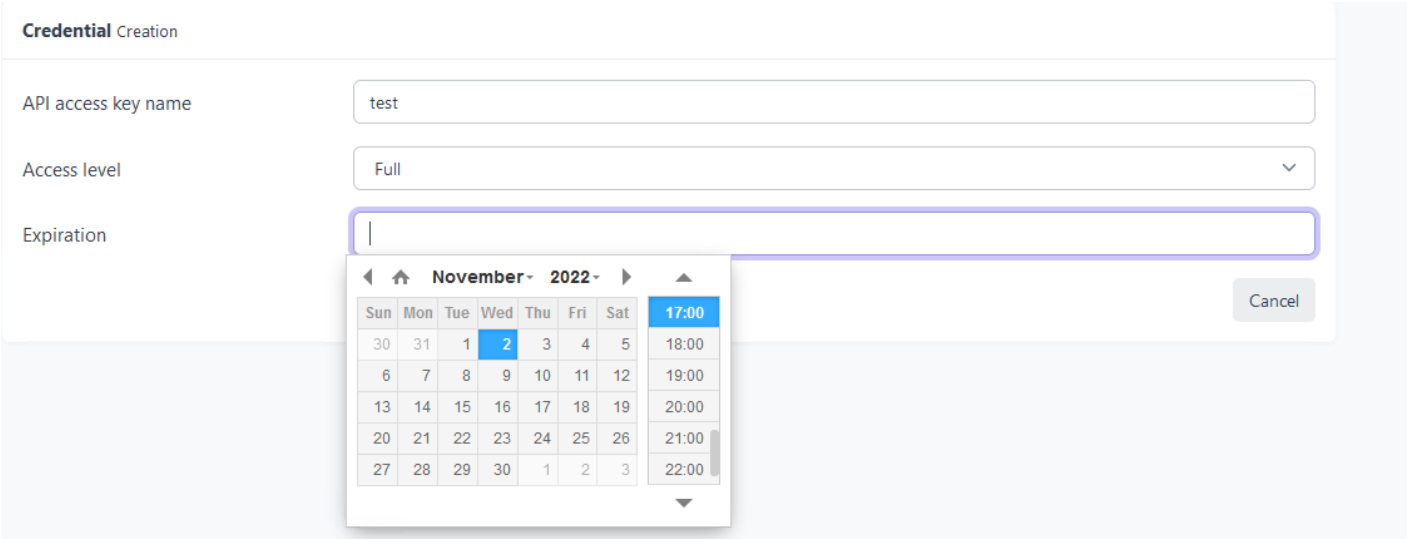
26.3.1 Working principle

WAPT Enterprise offers synchronization between the inventories of your hosts and Cyberwatch ISVM (Information Security Vulnerability Management) Software.

The method automatically synchronizes information about updates or installed softwares to Cyberwatch tool in order to scan and alert you about detected vulnerabilities.

26.3.2 Configuring Cyberwatch server side

- Connect to your Cyberwatch server and go to your profile.
- In the API section, click on **See my API Keys**.
- Click on *Add* and name your API access key for WAPT.



- Set the **access level** to Full and give an expiration date. If you don't give one, the key will *never* expire.

This key with its **API access key ID** will allow you to use the Cyberwatch API for our WAPT package.

26.3.3 Configuring WAPTAgent and sync packages

Install and configure the WAPT Agent on the computer that will run the synchronization. The WAPTAgent is installed by default on the WAPTServer, it just need to be configured.

To configure the WAPTAgent, please refer to the corresponding documentation.

Yu can have two packages :

- if you have the Cyberwatch agent, you can import from Cyberwatch installing the package tis-cyberwatch-plugin-import-from-cyberwatch, it will give you information directly on your WAPT Console.
- for agentless devices, you still can export to your Cyberwatch server information of you WAPT hosts installing the package tis-cyberwatch-plugin-export-to-cyberwatch-airgap, it will give you information to your Cyberwatch Console without Cyberwatch agent installed.

You need to configure an audit schedule on the agent

```
[global]
...
waptaudit_task_period=120m
...
```

With the package, whichever you chose (you can oblviously choose both), it will create two ini files in your $WAPT_INSTALL_DIR/private (linux : `/opt/wapt/private`, windows : `C:Program Files (x86)waptprivate`). Connect to the host and modify `cyberwatch_api.ini` and `wapt_api.ini` files.

```
[cyberwatch]
api_key =
secret_key =
url = https://cyberwatch.mydomain.lan
```

```
[wapt]
username = waptregister
password = waptregister2023!
url = https://srvwapt.ad.tranquil.it
```

To test the current configuration, you can trigger an audit

```
wapt-get audit tis-cyberwatch-plugin-import-from-cyberwatch
# and/or
wapt-get audit tis-cyberwatch-plugin-export-to-cyberwatch-airgap
```

CHAPTER

TWENTYSEVEN

ENHANCING THE SECURITY OF YOUR WAPT SETUP - CONSOLE SIDE

27.1 Generating the Certificate Authority (CA)

When installing WAPT, you are asked to *create* a `.pem` / `.crt` pair by checking the boxes *Tag as code signing* and *Tag as CA Certificate*.

This `.pem` / `.crt` pair will allow to sign WAPT packages and new certificates.

27.1.1 Generating a new certificate with the Certificate Authority

Build a new `.pem` / `.crt` pair.

Note: The new certificate will not be a self-signed certificate;

This new certificate will be signed by the CA (the key generated at the time of the first installation of WAPT);

You **MUST** then fill in the *Authority Signing Key* and the *Authority Signing Certificate*.

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 security guidelines of the *Organization* and the user customizations desired by her.

Generating a new .pem / .crt pair will also allow to formally identify the individual who has signed a package by looking up the CN attribute of the WAPT package certificate.

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*.

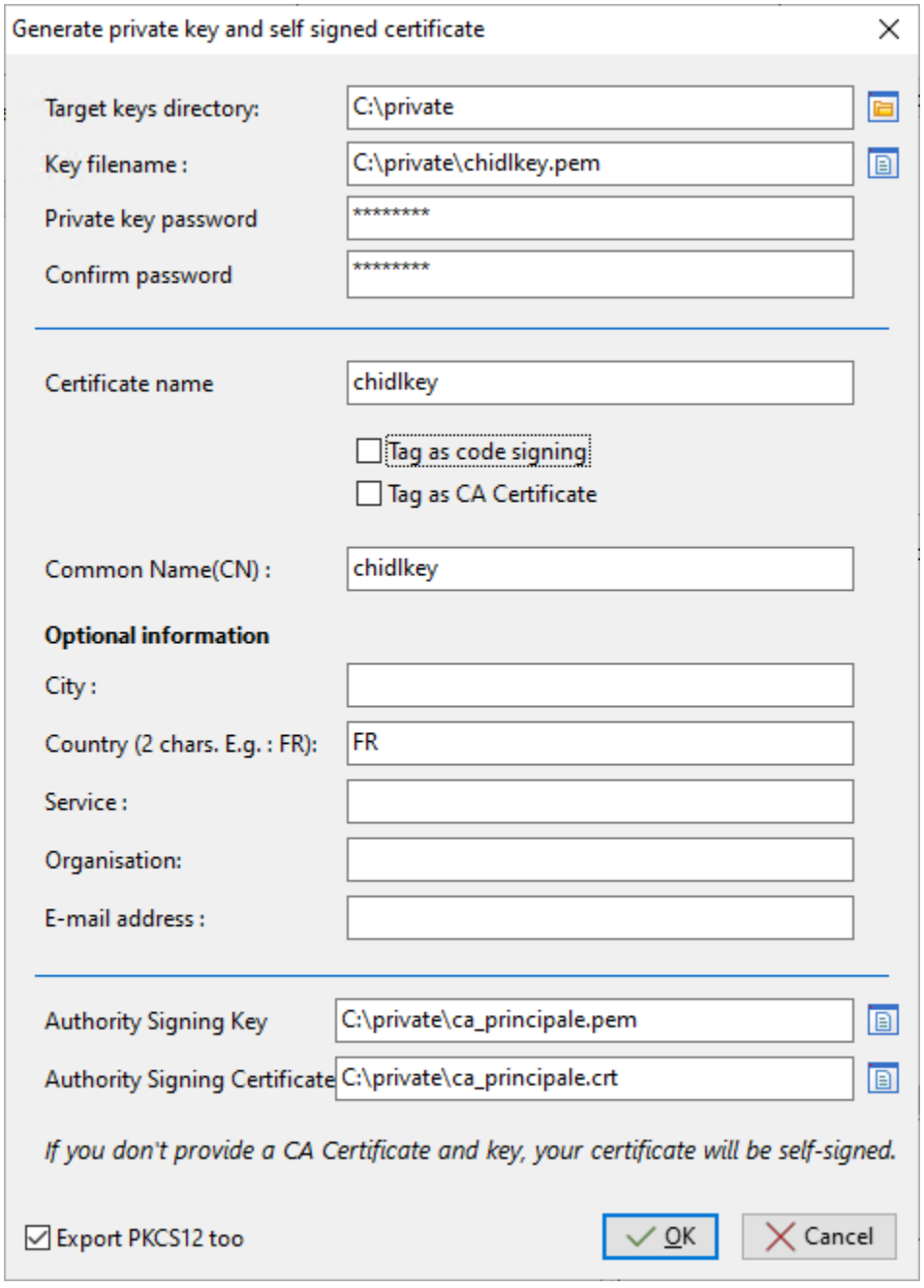


Fig. 1: Generating a certificate without the *Code Signing* attribute

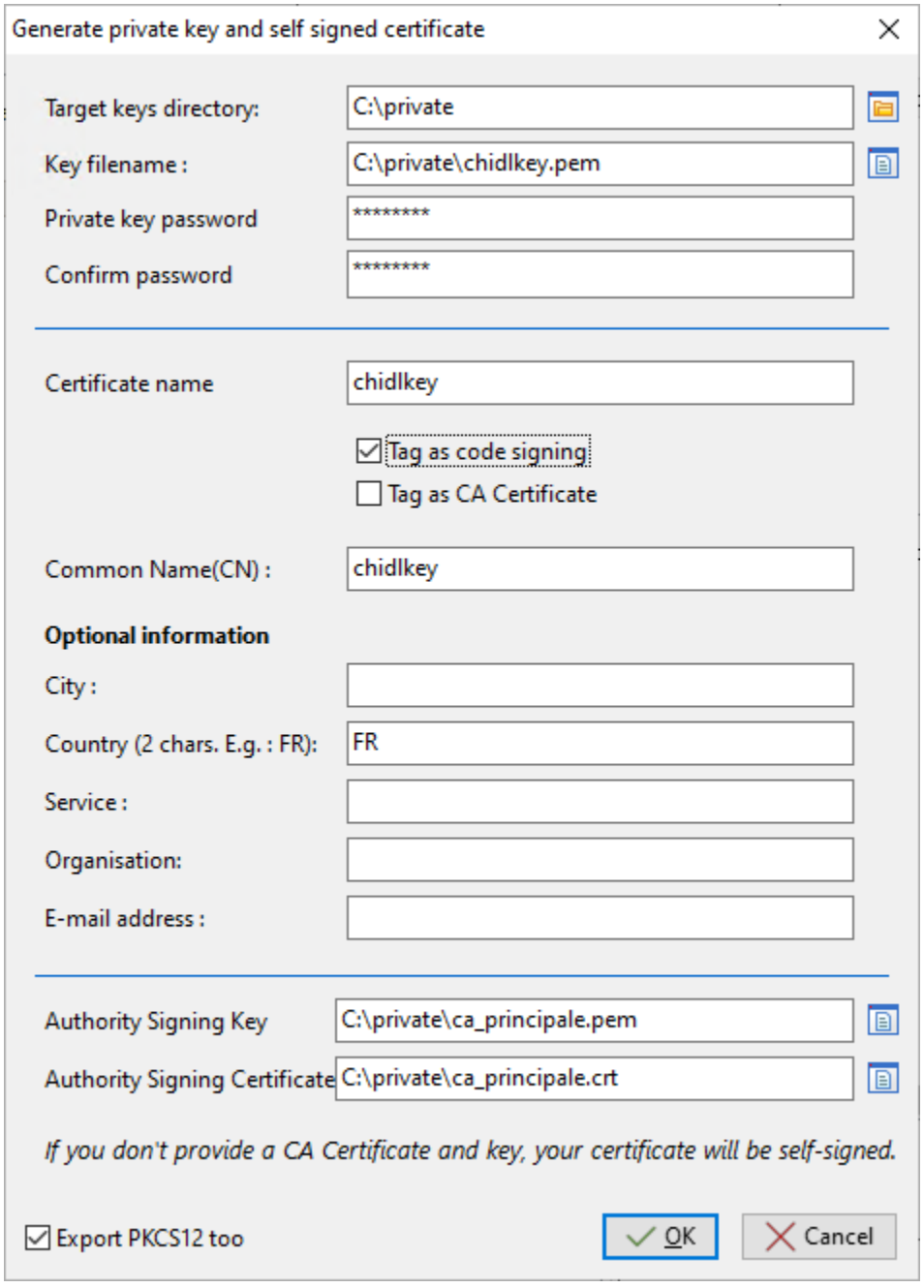


Fig. 2: Generating a certificate with the *Code Signing* attribute

Attention: It is not necessary to deploy child certificates with the WAPT Agent.

Child certificates are used with the WAPT Console to allow or restrict actions.

27.1.2 Deploying certificates of local IT admins on clients

Hint: Some Organizations will choose to let local IT administrators perform actions on WAPT equipped devices by issuing them personal 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.

It is possible to manage simply and in a finer way using *Access Control Lists* with the Enterprise version of WAPT.

You will need to copy the certificates of allowed local IT admins on WAPT clients in `C:\program files(x86)\wapt\ssl`.

Hint: Do not forget to restart the WAPT service on clients for them to use their new certificate. Open a command line **`cmd.exe`**.

```
net stop waptservice && net start waptservice
```

If you want to deploy the certificates using WAPT, use *WAPT package templates*

27.2 Displaying the Certificates trusted by the hosts in the WAPT Console

In this tab, you can see the certificates that the host accepts to trust.

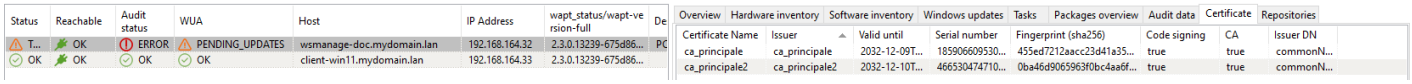


Fig. 3: Window showing the certificates trusted by the selected host

27.3 Configuring Access Control Lists

Hint: The *SuperAdmin* user of WAPT is authenticated by a password stored in `waptserver.ini` as a value of the `wapt_password` attribute. Others WAPT users may be local users `htpasswd_path`) or AD account users (`ldap_auth_server` / `ldap_auth_base_dn`).

ACLs define actions enabled for all types of users in the WAPT context.

Note: Default ACLs user level are defined by `default_ldap_users_acls` in `waptserver.ini`.

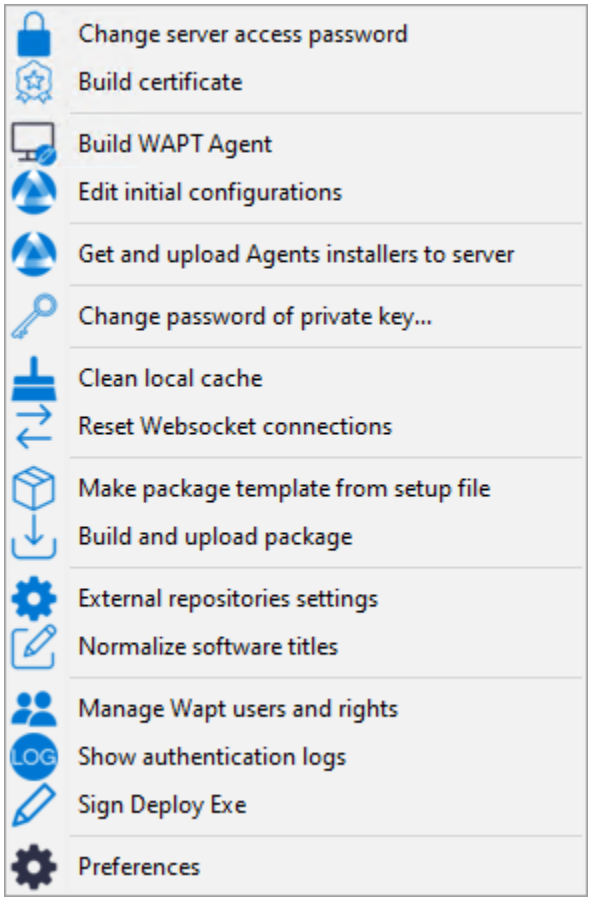
The default ACL for a new user is `view`.

Attention: Security is define by the certificate deployed on clients, not by ACLs.

ACLs simply limit what actions the WAPT Server is allowed to relay from the WAPT Console to the WAPT Agents.

As of |date|, the WAPT Agents do not check ACL rights.

To configure ACLs in WAPT, go to *Tools → Manage WAPT users and rights*.



Note: On first launch after the WAPT Server installation, only the *SuperAdmin* account is present in the list of users.

If the *SuperAdmin* account does not exist or does not have the *admin* right, then the account is recreated by restarting the WAPT Server service.

The *SuperAdmin* account is authenticated using the value of `wapt_password` in the `waptserver.ini` configuration file.

Two types of account are manageable by ACL, *local* and *Active Directory*.

27.3.1 Local user account

Local users are defined by a *.htpasswd* file.

WAPT Server configuration

For using local user accounts, you need create a file named `waptusers.htpasswd` in the same *folder* on the WAPT Server containing the `waptserver.ini` file.

Linux:

```
touch /opt/wapt/conf/waptusers.htpasswd
chown wapt /opt/wapt/conf/waptusers.htpasswd
```

Windows

```
cd. > C:\wapt\conf\waptusers.htpasswd
```

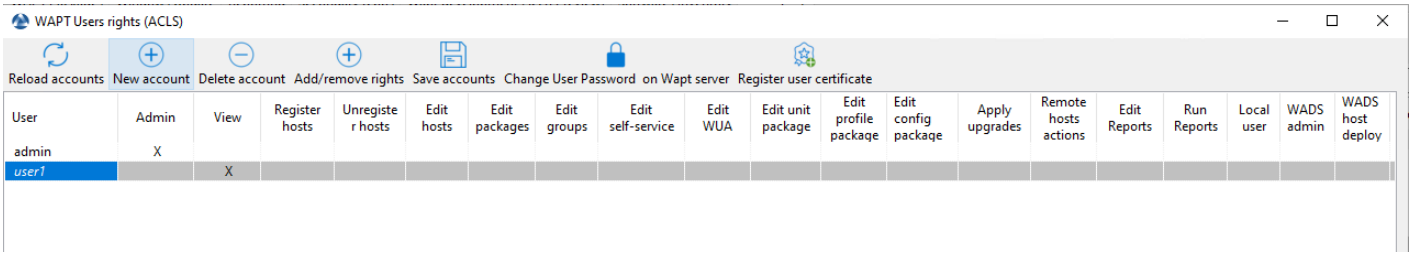
- On `waptserver.ini` add `htpasswd_path` settings.

```
htpasswd_path = password file location
```

Hint: Restart **the WAPT Server** service

Creating the user account

- In *WAPT Users rights* window, click on *New account*.



It is possible to rename accounts by pressing `F2` on the *User* column.

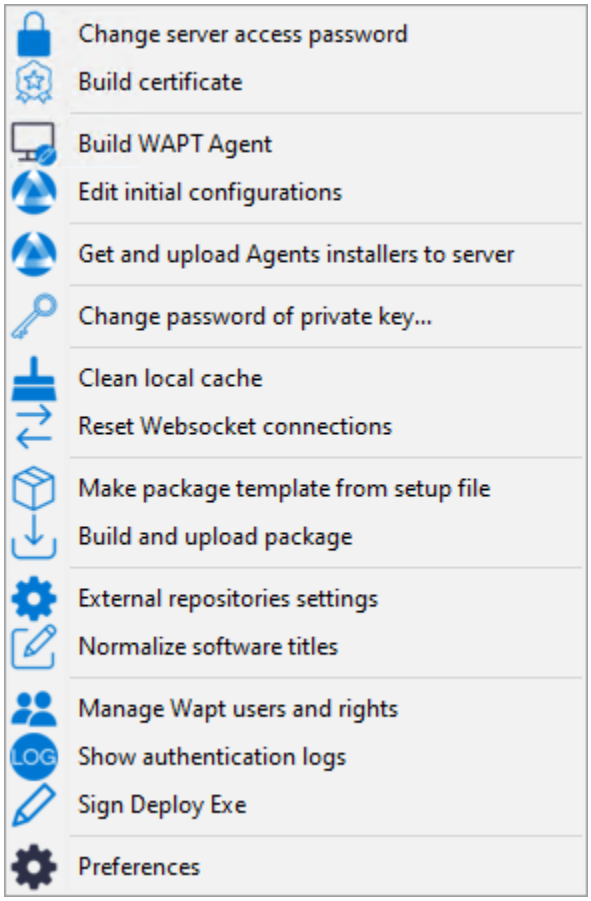
- Save by clicking on *Save account*.
- For setting a password, see below.
- For setting rights, see the section on *managing ACL rights*.

If the local user has a password in `waptusers.htpasswd`, then the username appears in **bold** and *Local User* is checked, else change the password for this user.

Changing the user password

To change the password for the selected account:

- Do a *right click on the account → Change User Password on Wapt Server*.



- Enter the new password.

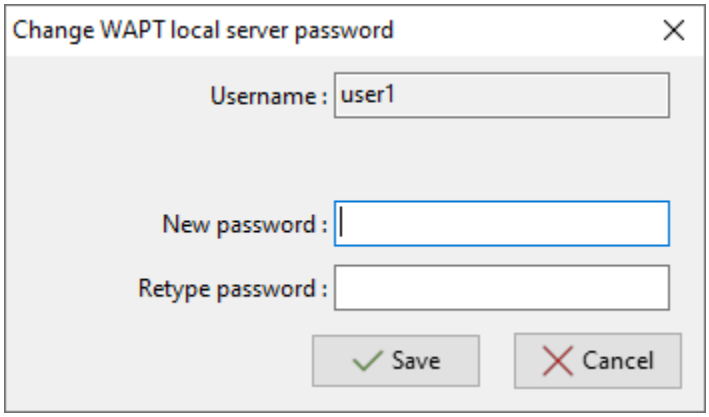


Fig. 4: Dialog box for changing the user password in the htaccess file

The local user appears in *bold* and the *Local User* is checked.

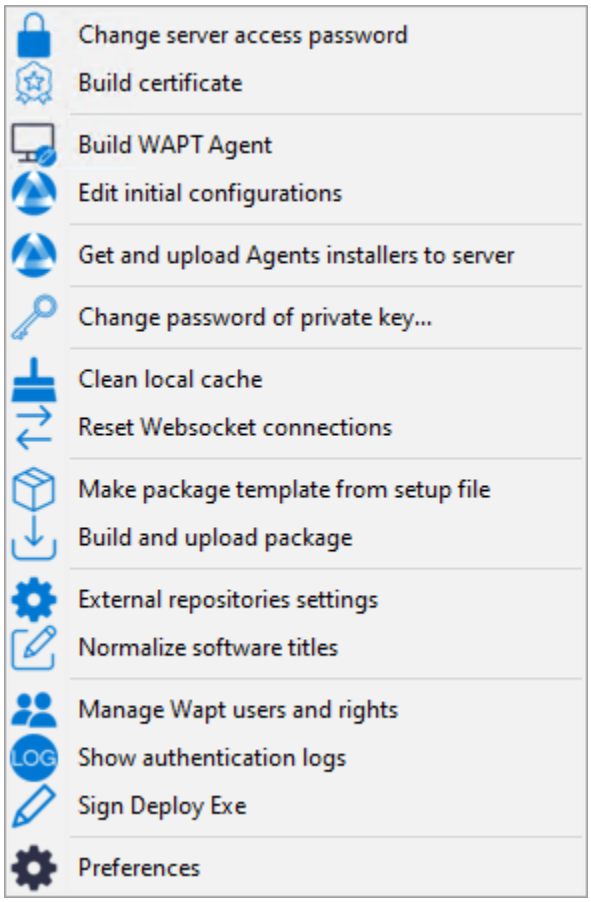
27.3.2 WAPT users set as Active Directory users

To manage WAPT users with Active Directory, you need to activate *Active Directory authentication*.

After a first successful login, the AD account will appear automatically in the list of WAPT users.

27.3.3 Blocking local user accounts

To unregister local users, do *right click on the account → Invalidate User Password on WAPT Server*.



The user account will be blocked from managing anything in WAPT.

27.3.4 List of rights

Many *rights and restrictions* can be set for each user in the WAPT Console.

Table 1: List of user rights

Right	Description
Admin	Grants the same rights as *SuperAdmin*, all rights are granted except *local user*.
View	Allows only view information on the WAPT Console.
Register hosts	Allows to use the Admin credentials to *register manually a host* with the WAPT Server.
Unregister hosts	Allows to *remove a host* from the WAPT Console.
Edit hosts	Allows to *edit the host profile* on the WAPT Console.
Edit packages	Allows to *modify base packages* on the WAPT Console.
Edit groups	Allows to *modify group packages* on the WAPT Console.
Edit self-service	Allows to *modify self-service rules* on the WAPT Console.
WUA	Allows to *modify WUA / WSUS rules* on the WAPT Console.
Edit unit package	Allows to *modify unit packages* on the WAPT Console.
Edit profiles package	Allows to *modify profiles packages* on the WAPT Console.
Apply upgrades	Allows to remotely apply upgrades on her perimeter of hosts, if host is on **PENDING** status.
Remote hosts actions	Allows to make use of the Windows Computer Management tool with the WAPT Console.
Edit Reports	Allows to *create new or modify reporting queries*.
Run Reports	Allows to *run existing SQL reports*.
Local user	Defines a Local User

27.3.5 Managing rights

By default, the **SuperAdmin** is the *CA certificate* user.

For other user, it is possible to associate a certificate that has been generated from the WAPT PKI or from another CA.

These certificates may or may not be children of the WAPT Certificate Authority.

Attention: If certificates are not issued from the Certificate Authority:

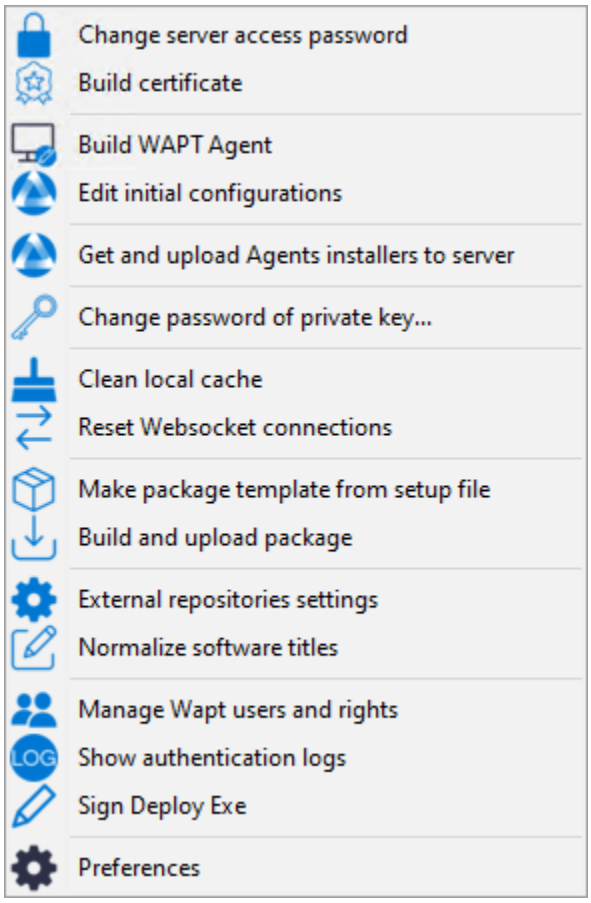
- Updated WAPT packages are available only to computers where certificates are deployed.
- ACLs are valid only on the perimeter of the hosts where the certificates are deployed.

Associating a certificate to a user

Hint: By default no certificate is set for any user (including *SuperAdmin*).

The account in the WAPT Console appears in *italic* if no certificate is associated to the user.

To associate a certificate to an user, do *Right-Click on user* → *Register user certificate*.



Then, choose the certificate to associate to the user.

Adding / Removing rights

To add or remove rights, select the cell with *left click* and check-it by pressing the `spacebar`.

Hint: It is possible to do a multiple selection by using keyboard shortcuts `Crtl+left-click` and pressing the `spacebar`.

Restricting the perimeter of rights permitted to user

It is possible to associate a perimeter to a right given to a user.

View

Table 2: Definition of the allowed perimeter

Perimeter	Description
Deny all	Denies any view right (not checked).
Allow on any perimeter	Allows view right for all WAPT Agents.
Allow specific perimeters	Allows view right on the selected perimeter defined as a list of certificates.
Allow where user certificate is deployed	Allows view only on the perimeter where the certificate of the WAPT Administrator is deployed.

Edit group packages

Hint: All group packages work on the same principle as described below.

Table 3: Definition of the allowed perimeter

Perimeter	Description
Deny all packages	Denies any edit right to any package (not checked).
Allow any packages	Allows edit right to all WAPT packages.
Allow specific packages name	Allows edit right for the WAPT packages selected in the list.

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*.

CHAPTER

TWENTYEIGHT

MANAGING THE WAPT AGENT

28.1 Deploying the WAPT Agent on Windows

Note: To install WAPT on a Windows client, the minimal requirements are:

- 512Mo Ram;
- 1 CPU;
- 300Mo Drive space (without package cache).

Attention: If you install the WAPT Agent on **Windows Server 2012r2**, it needs these features need to be activated before installing the WAPT Agent:

- `KB2919442`.
- `KB2919355`.
- `vcredist2015`

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

- The first method is manual and the procedure **MUST** be applied on each host.
- The second one is automated and relies on a GPO.

The **`waptagent.exe`** installer is available at WAPT serveur web home page. The direct download link is for example: https://srvwapt.mydomain.lan/wapt/waptagent.exe.

Warning: 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 host's certificate store.

28.1.1 Manually

Manually installing the WAPT Agent requires *Local Administrator* rights on the computer. Manually installing the WAPT Agent using a Domain Admin account **WILL NOT WORK**.

Manual deployment method is efficient in these cases:

- Testing WAPT.
- Using WAPT in an organization with a small number of computers.
- If you do not have a means of mass deployment.
- Download the WAPT Agent from your WAPT Server then launch the installer.

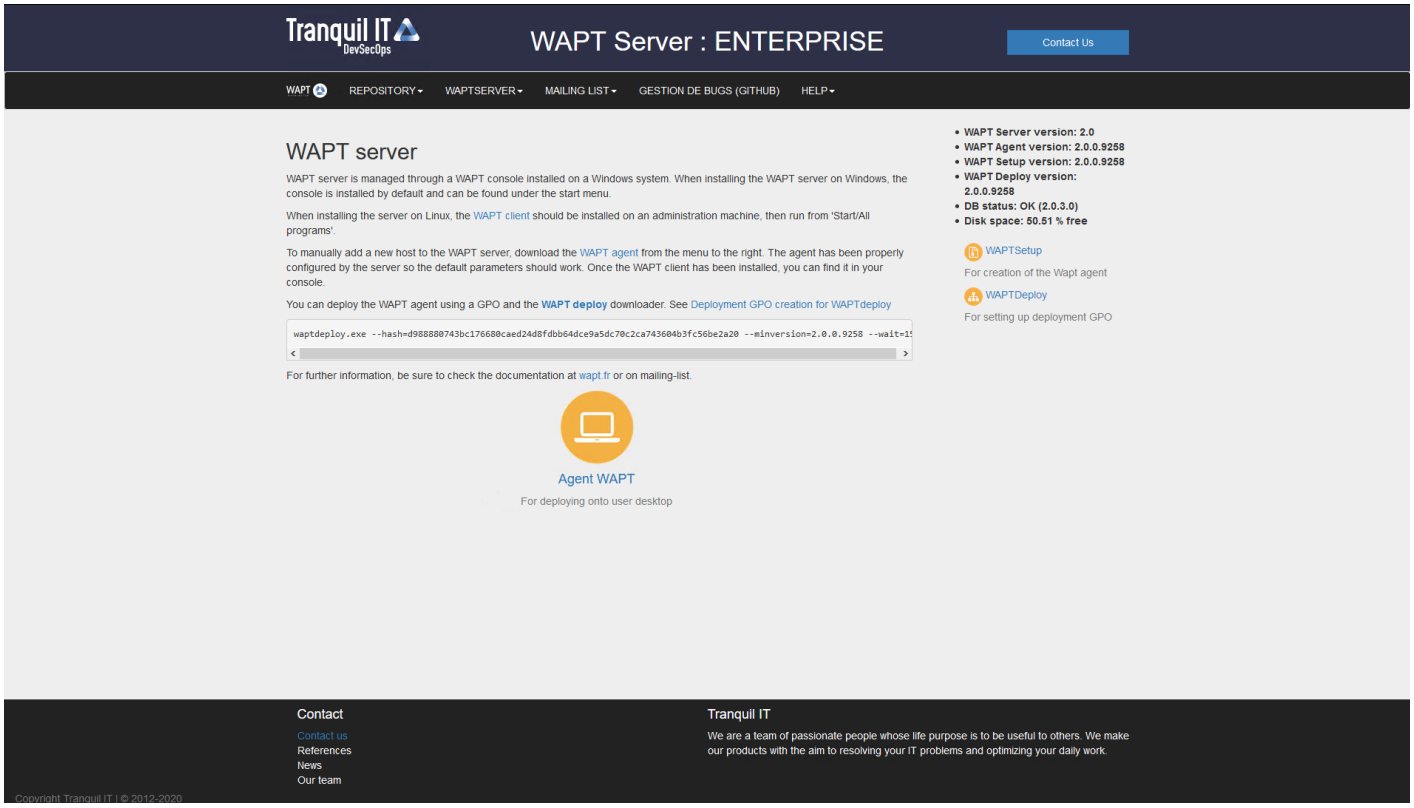
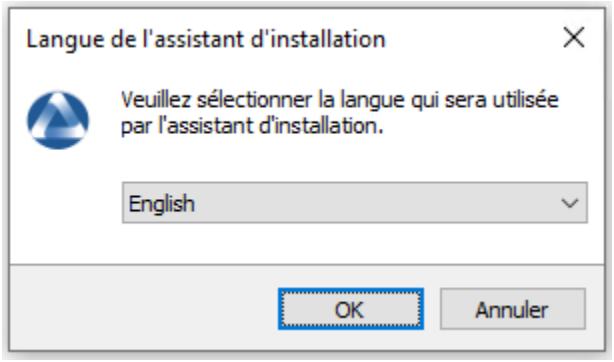
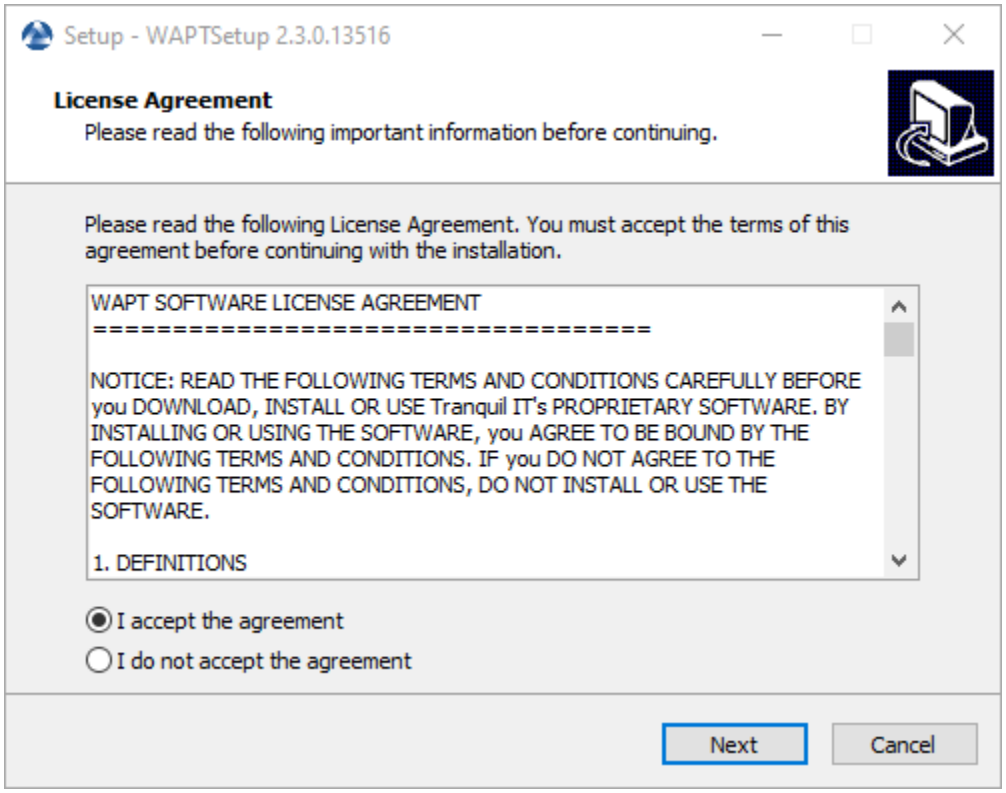


Fig. 1: The WAPT Server interface in a web browser

- Choose the language for the WAPT installer.



- Click on *OK* to go on to the next step.



- Accept the licence terms and click on *Next* to go to next step.
- Choose additional configuration tasks (leave the default if not sure).

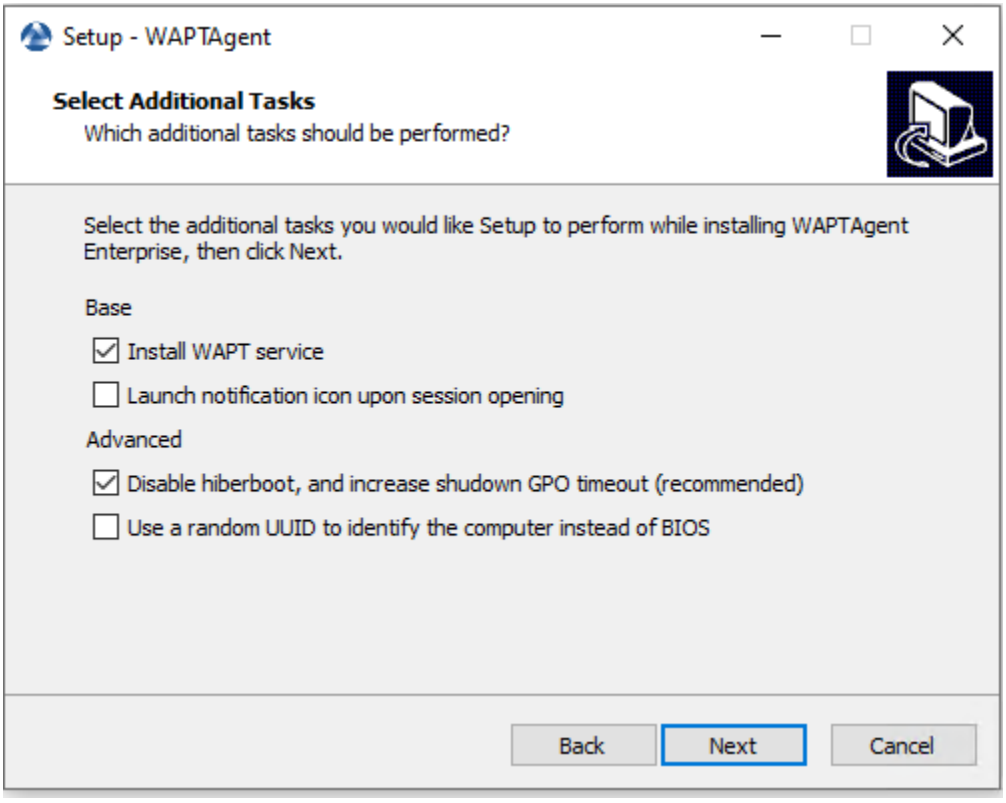


Fig. 2: Choosing the installer options for deploying the WAPT Agent

Table 1: Avialable options

Settings	Description	Default value
Install WAPT service	Adds the WAPT service on the computer.	Checked
Launch notification icon upon session opening	Launches the WAPT Agent in the System tray on startup.	Not checked
Disable hiberboot, and increase shutdown GPO timeout (recommended)	Disables Windows fast startup for stability, increases the timout for the WAPT Exit utility.	Checked
Use a random UUID to identify the computer instead of BIOS	Solves possible *BIOS UUID bugs*.	Not checked

- Choose the WAPT repository and the WAPT Server and click on *Next* to go to next step.

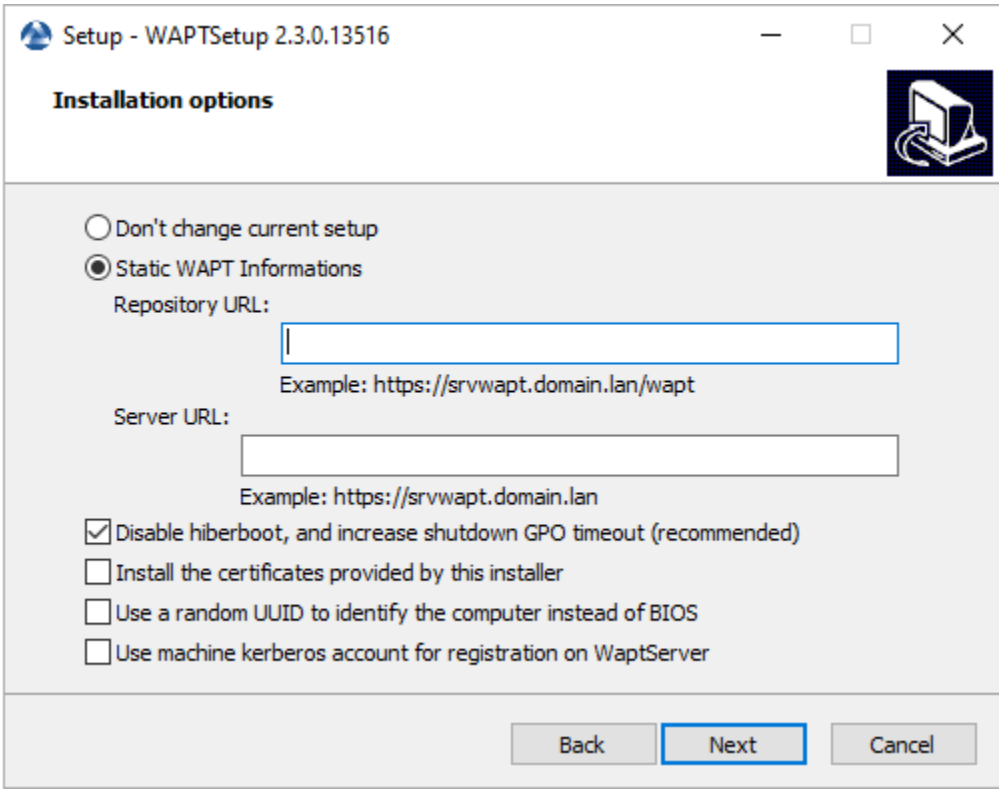
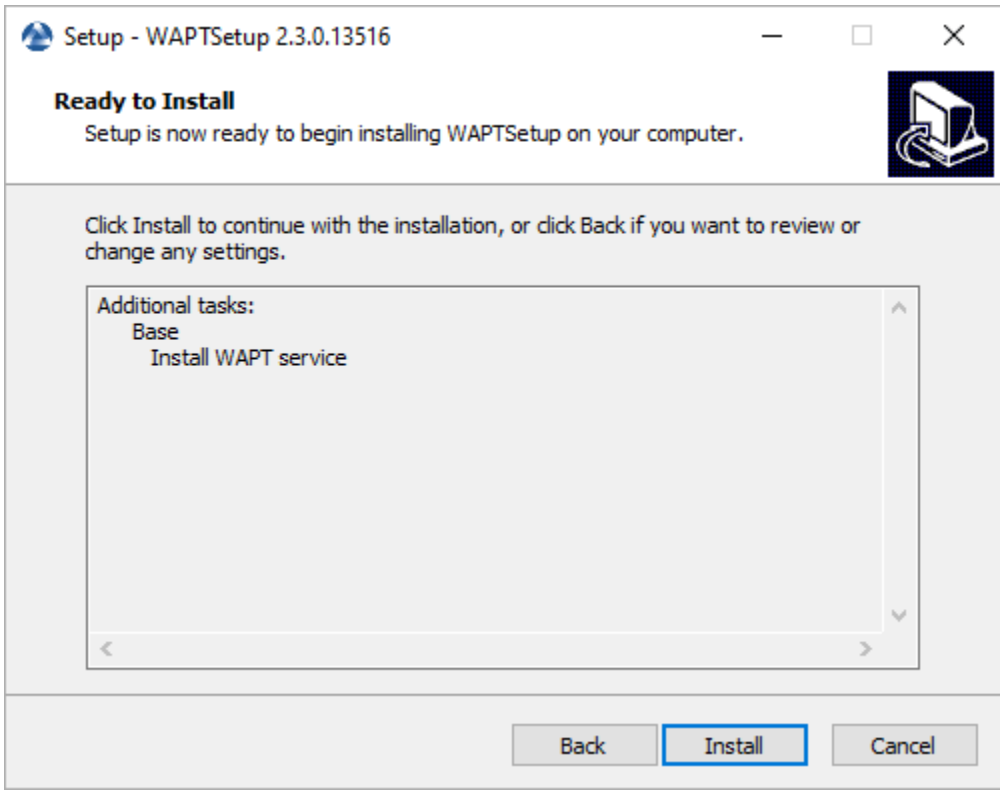
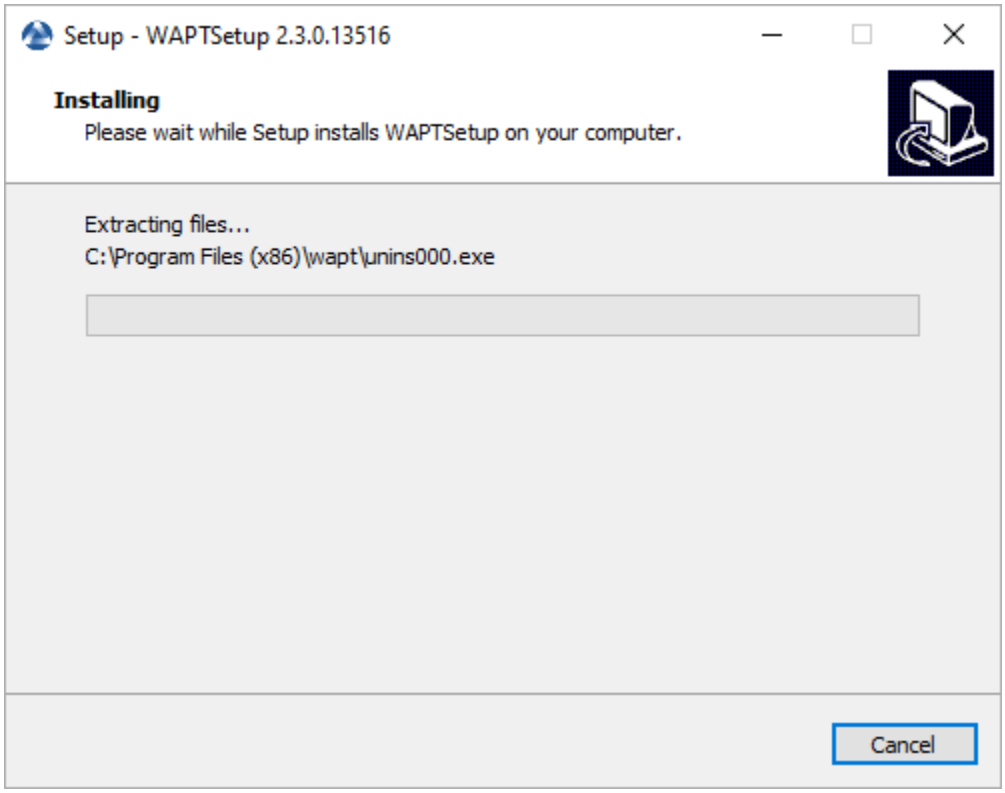


Fig. 3: Choosing the WAPT repository and the WAPT Server

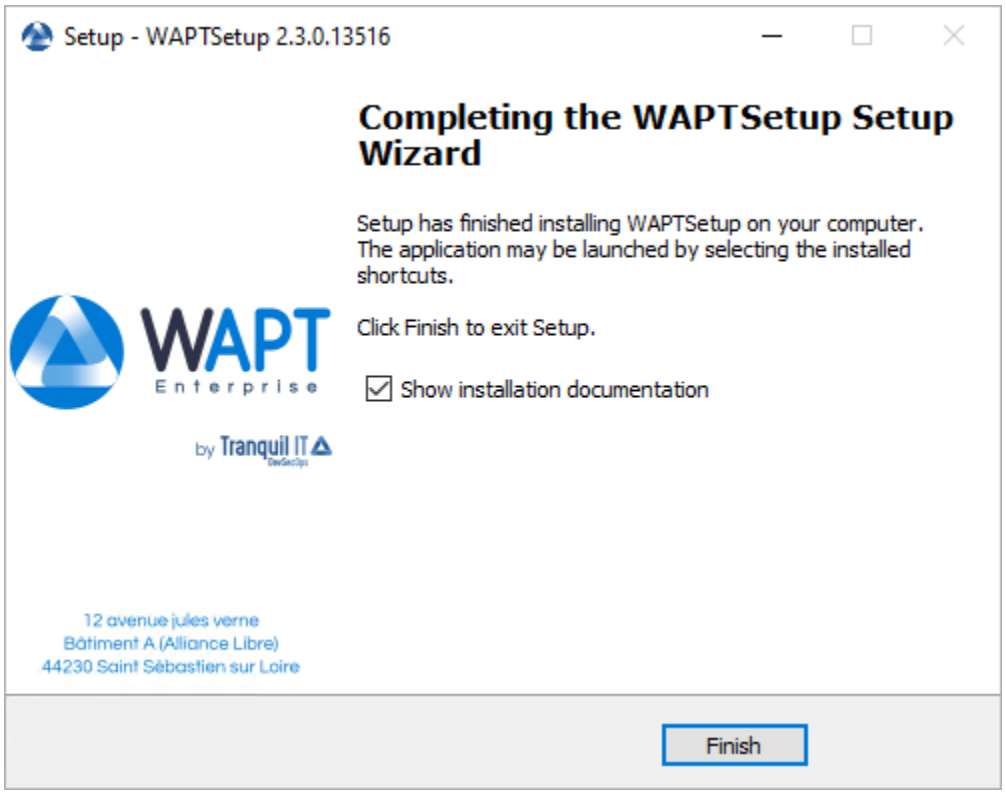
- Install the WAPT Agent by clicking on *Install*.



- Wait for the installation of the WAPT Agent to finish, then click on *Finish* to exit.



The installation of the WAPT Agent is finished. The registration of the host with the WAPT Server is done automatically.



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

28.1.2 Automatically

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*.

With the WAPT Deployment utility

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

```
waptagent.exe /VERYSILENT
```

- Additional arguments are available for the WAPT Deployment utility.

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.
:code:/verify_cert = ``True 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 in C:\Program Files (x86)\wapt\waptsetup\waptsetup.iss.

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

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

The WAPT Deployment utility 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).

Warning: The WAPT Deployment utility **MUST** be started as *Local Administrator*, that is why a GPO is a good method to deploy the WAPT Agent.

Download waptdeploy.exe from your WAPT Server homepage.

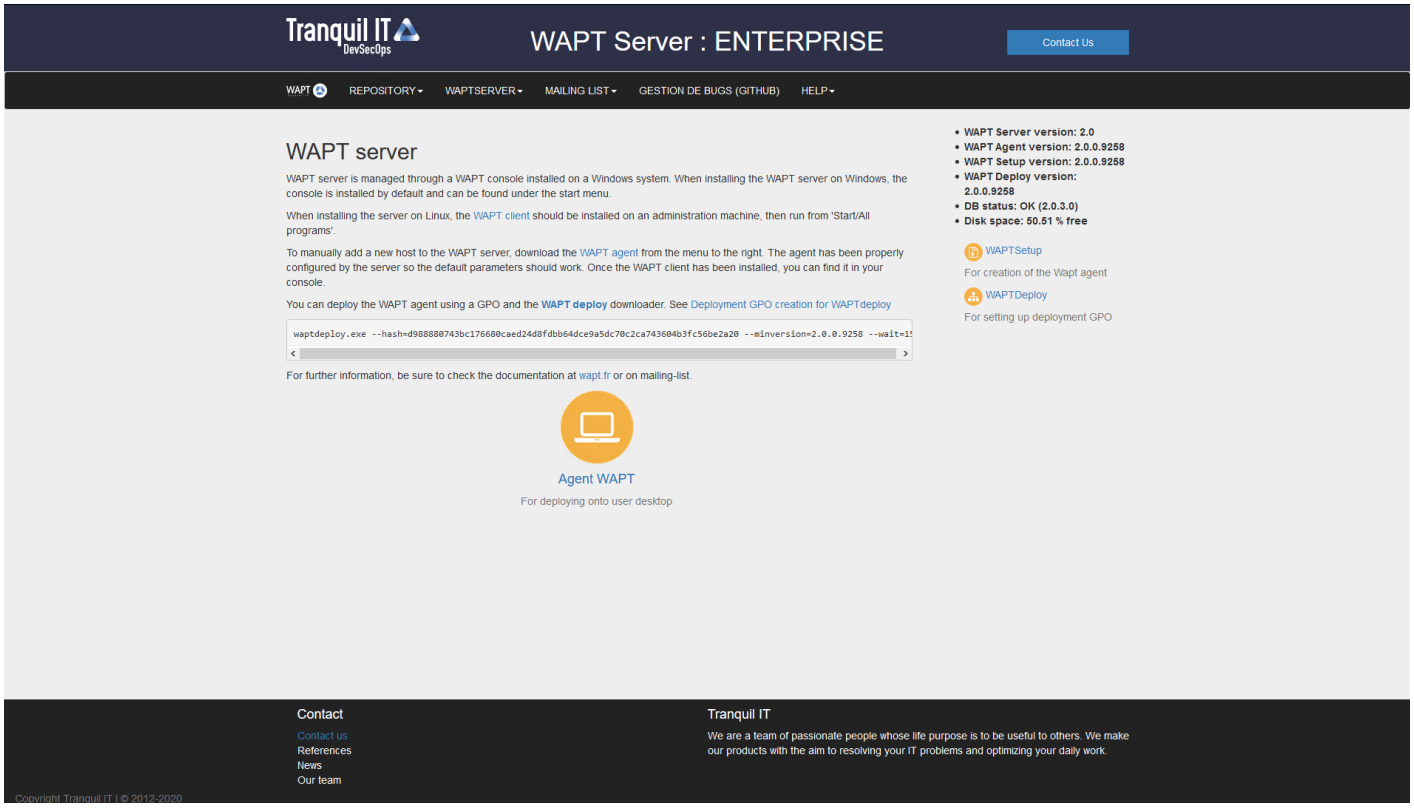


Fig. 4: The WAPT Server interface in a web browser

With a GPO

- Create a new group strategy on the Active Directory server (Microsoft Active Directory or Samba-AD).
- Add a new strategy with *Computer configuration* → *Policies* → *Windows Settings* → *Scripts* → *Startup* → *Properties* → *Add*.

Fig. 5: Creating a group strategy to deploy the WAPT Agent

- Click on *Browse* to select the `waptdeploy.exe`.

Fig. 6: Finding the WAPT Deployment utility file on your computer

- Copy `waptdeploy.exe` in the destination folder.

Fig. 7: Selecting the the WAPT Deployment utility script

- Click on *Open* to import the `waptdeploy.exe`.
- Click on *Open* to confirm the importation of the the WAPT Deployment utility binary.

Hint: It is necessary to provide the checksum of the `waptagent.exe` as an argument to the the WAPT Deployment utility GPO. This will prevent the remote host from executing an erroneous / corrupted **`waptagent`** binary.

```
--hash=checksum WaptAgent --minversion=2.4.0 --wait=15 --waptsetupurl=http://srvwapt.mydomain.lan/
↪wapt/waptagent.exe
```

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

- Copy the required parameters into the GPO.
- Click on *OK* to go on to the next step.
- Click on *OK* to go on to the next step.
- Apply resulting GPO strategy to the Organization's Computers OU.

Note: We recommend adding `waptdeploy.exe` to the startup and shutdown scripts on the GPO.

Hint: More arguments are available for the WAPT Deployment utility

---

Fig. 8: Selecting the the WAPT Deployment utility script

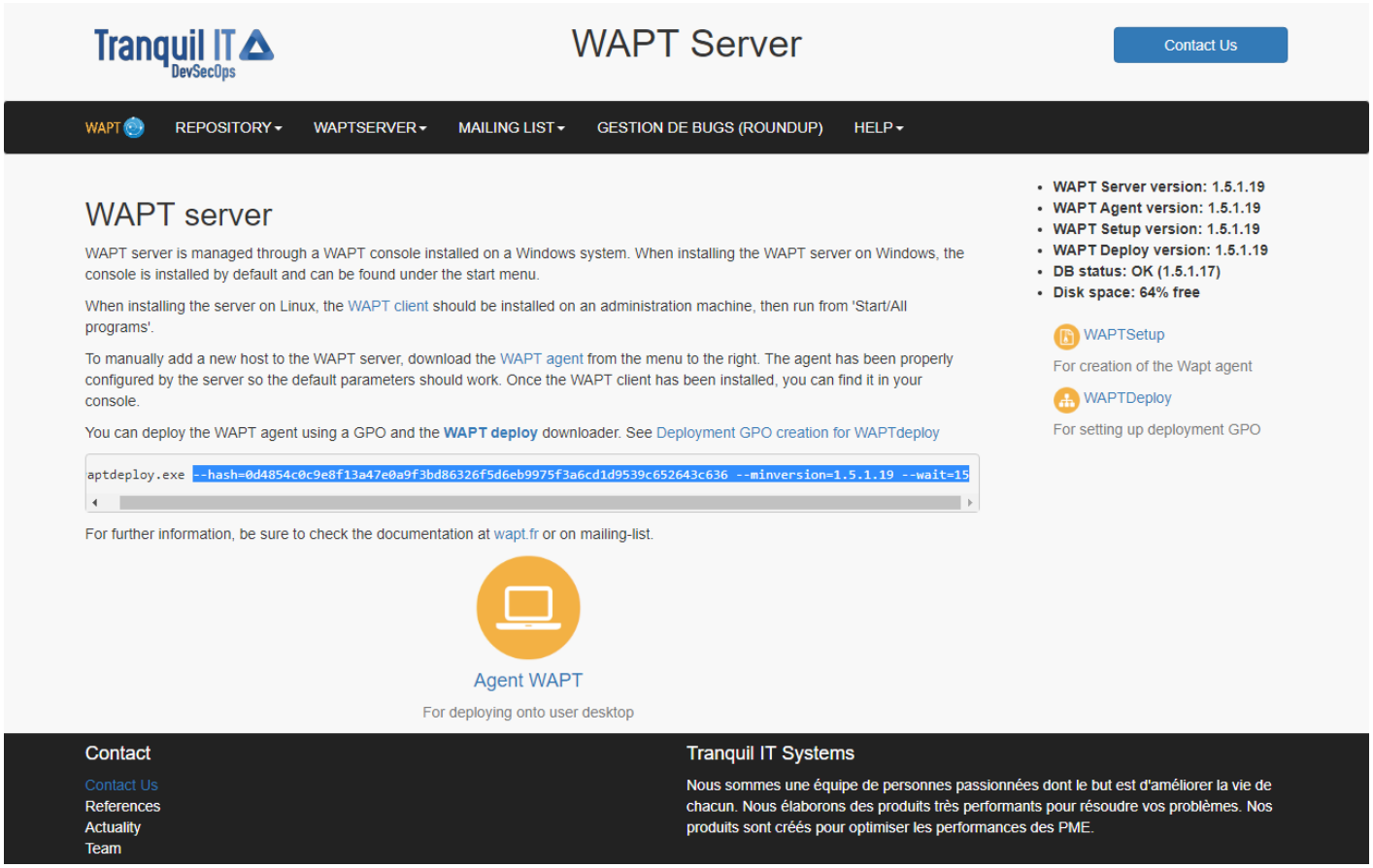


Fig. 9: Web console of the WAPT Server

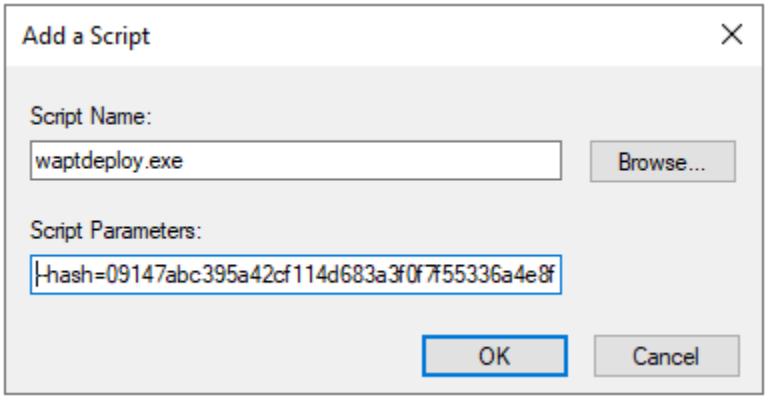


Fig. 10: Adding the the WAPT Deployment utility script to the startup GPO

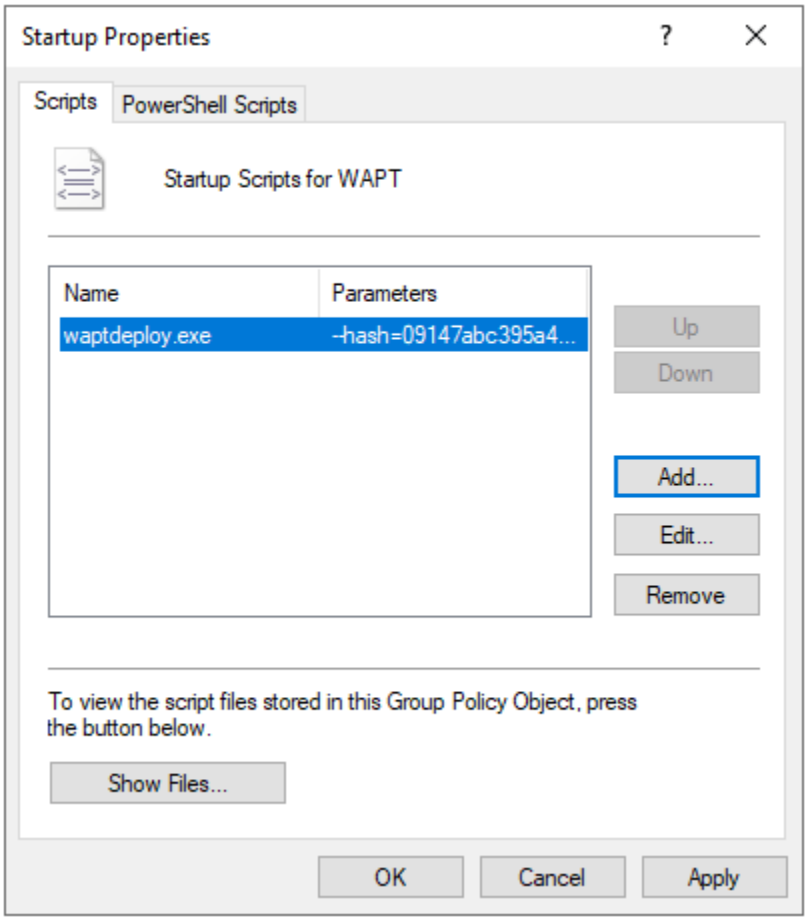


Fig. 11: The WAPT Deployment utility GPO to be deployed on next startup

Table 3: Description of available options for the WAPT Deployment utility

Options	Description
`--force`	Forces the installation of **`waptagent.exe`** even if alread installed.
`--hash` = <sha256hash>	Check that the downloaded **`waptagent.exe`** setup sha256 hash matches the hash.
`--help`	Displays the options
`--minversion` = <version>	Install **`waptagent.exe`** if installed version is less than minversion.
`--tasks` = autorunTray,installService,installredist2008,autoUpgradePolicy	If given, it passes the arguments to the /TASKS options of the **`waptagent`** installer (default is `autorunTray, installService, installredist2008, autoUpgradePolicy`).
`--repo_url` = <repo_url>	Location of the repository to get **`waptagent.exe`** (default <repo_url>/wapt)
`--setupargs` = <setupargs>	Adds arguments to the command line of **`waptagent.exe`**.
`--wait` = <minutes>	Defines the delay for running and pending tasks to complete if **`waptservice`** is running before installing.
`--waptsetupurl` = <waptsetupurl>	Explicit location to download setup executable. It can be a local path (default <repo_url>`/waptagent.exe`).

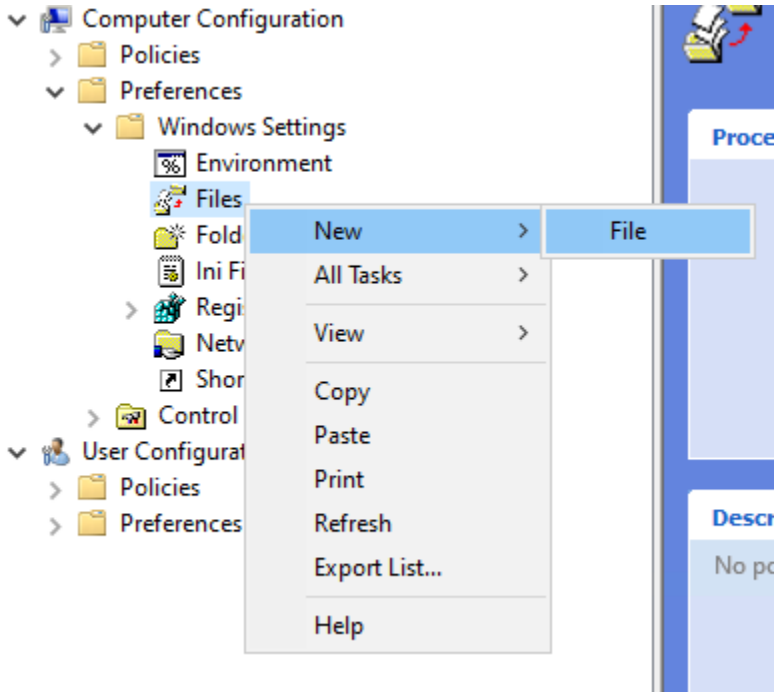
With a scheduled task

You may also choose to launch the WAPT Deployment utility 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 locally `waptdeploy.exe` and `waptagent.exe` and create a scheduled task for installing.

- Copy `waptdeploy.exe` and `waptagent.exe` in the netlogon share of your Active Directory Server (`\mydomain.lan\netlogon\waptagent.exe`).
- Create a new group strategy on the Active Directory server (Microsoft Active Directory or Samba-AD).
- Add a new strategy with *Computer configuration* → *Preferences* → *Windows Settings* → *Files*.
- Create a new file and copy the WAPT Deployment utility.



- Set parameters.

Table 4: Description of options for copy

Options	Value
Action dropdown menu list	Replace
Source file(s) field	`\mydomain.lan\netlogon\waptdeploy.exe`
Destination File field	`C:\Temp\waptdeploy.exe`
Suppress errors on individual file actions checkbox	not checked
Read-only checkbox	not checked
Hidden checkbox	not checked
Archive checkbox	checked

- Create a new GPO and copy the **`waptagent.exe`** file.

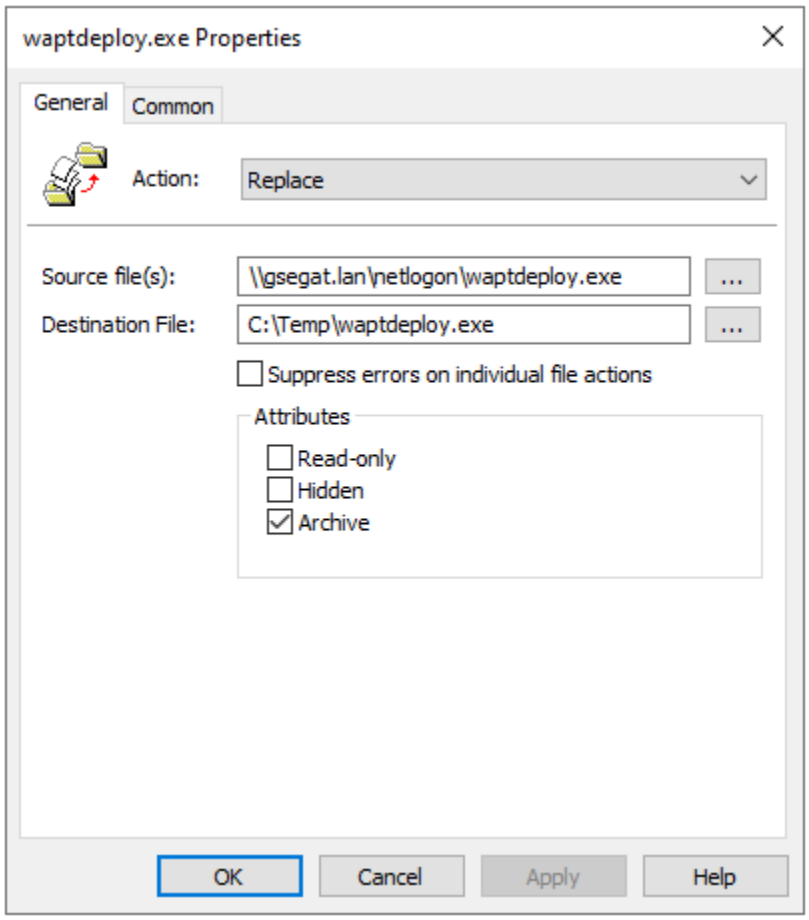
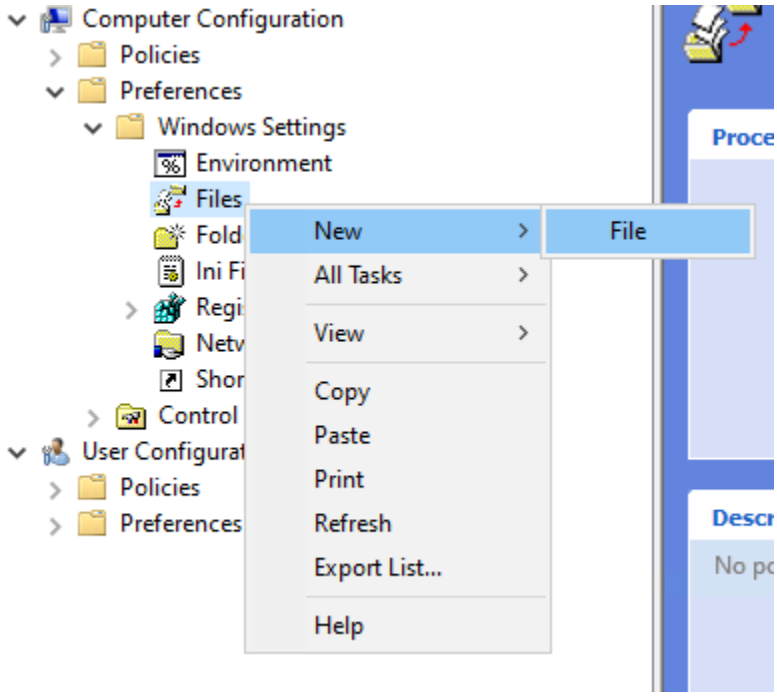


Fig. 12: WAPT Agent installation progress



- Set parameters.

Table 5: Description of options for copy

Options	Value
Action dropdown menu list	Replace
Source file(s) field	`\mydomain.lan\netlogon\waptagent.exe`
Destination File field	`C:\Temp\waptagent.exe`
Suppress errors on individual file actions checkbox	not checked
Read-only checkbox	not checked
Hidden checkbox	not checked
Archive checkbox	checked

- Then go to the Scheduled Task menu with *Computer configuration* → *Preferences* → *Control Panel Settings* → *Scheduled Tasks*.
- Create a new Scheduled Task with *Right-click* → *New* → *Scheduled Task (At least Windows 7)*.
- Set *Action* to `Replace`.
- For *When running the task, use the following user account* paste *S-1-5-18* (system account). You can visit for more information.
- Check *Run whether user is logged on or not*.
- Check *Run with highest privileges*, then go on to the *Triggers* tab.
- Create a new trigger.
- Check *Daily*, select *today's date*.
- Check *Repeat Task every* and select *1 hour* and *for a duration of* select *1 day*.
- Check *Stop task if it runs longer than* and select *2 hours*.
- Check that *Enabled* is checked, and then go to the *Actions* tab.

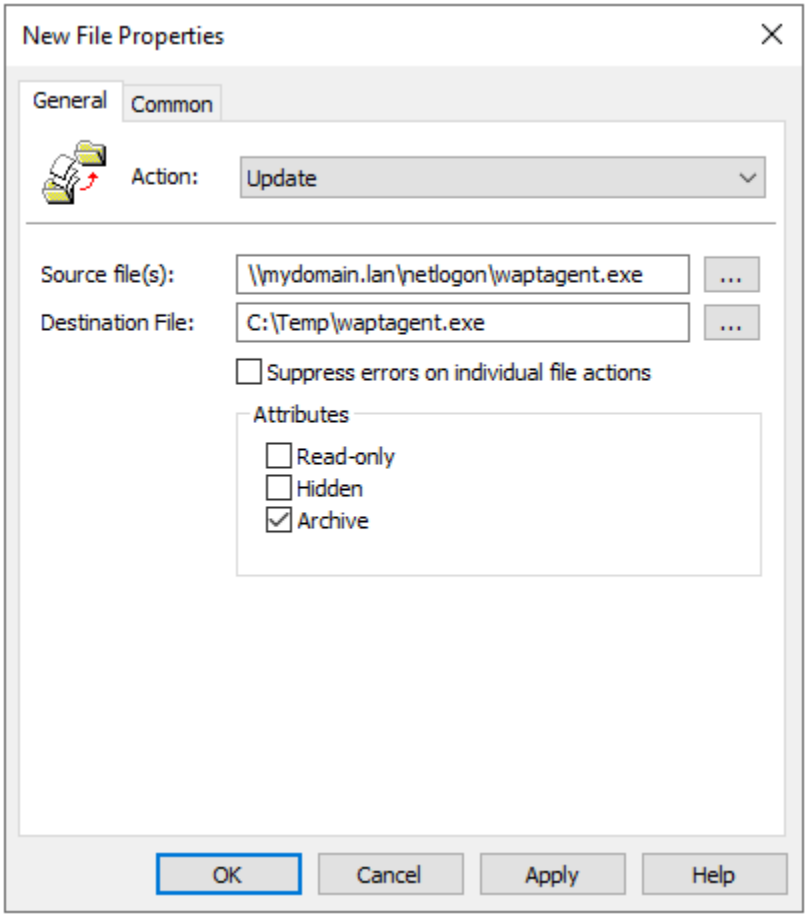


Fig. 13: Preparing the WAPT update GPO

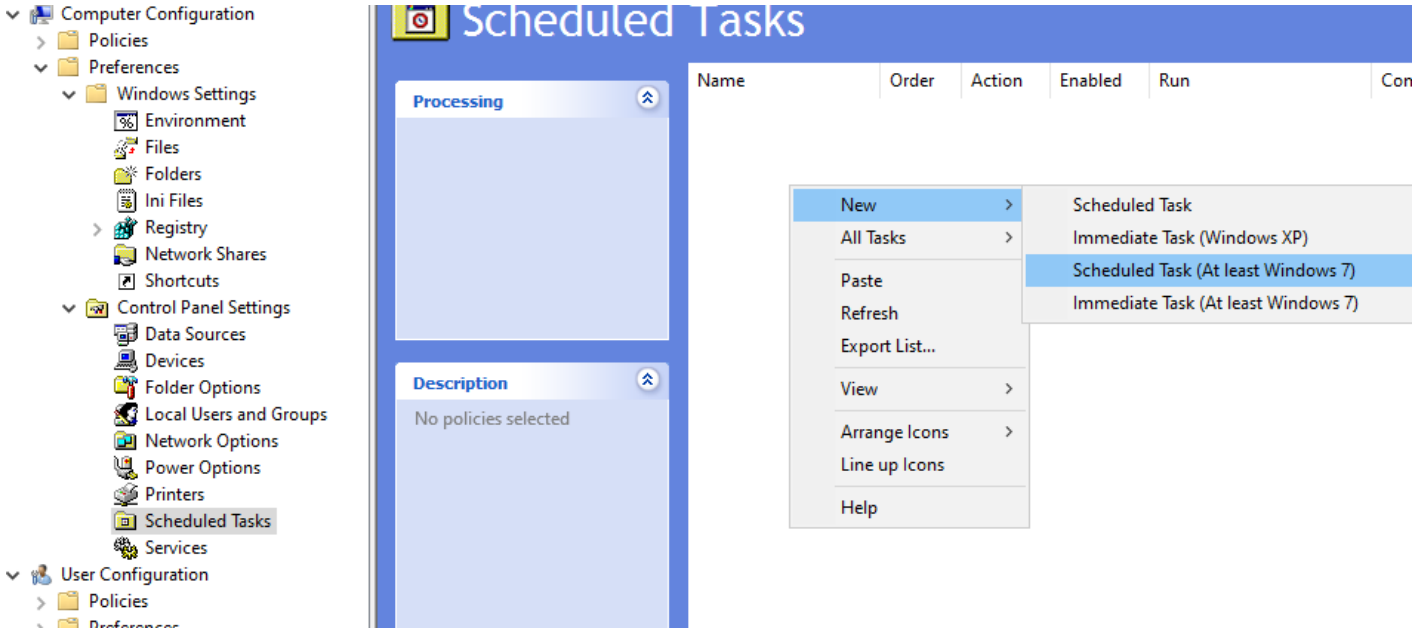


Fig. 14: Create the scheduled task for the WAPT Deployment utility Properties window in RSAT

- Create a new action *Start a program* for `waptdeploy.exe`.

Table 6: Description of options to copy

Options	Value
Action	Start a program
Program / script	C:\Temp\waptagent.exe
Add arguments (optional)	See the next point
Start in (optional)	empty

Hint: It is necessary to provide the checksum of the `waptagent.exe` as argument to the WAPT Deployment utility. This will prevent the remote host from executing an erroneous / corrupted **`waptagent`** binary.

```
--hash=checksum WaptAgent --minversion=2.4.0 --wait=15 --waptsetupurl=http://srvwapt.mydomain.lan/
↪wapt/waptagent.exe
```

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

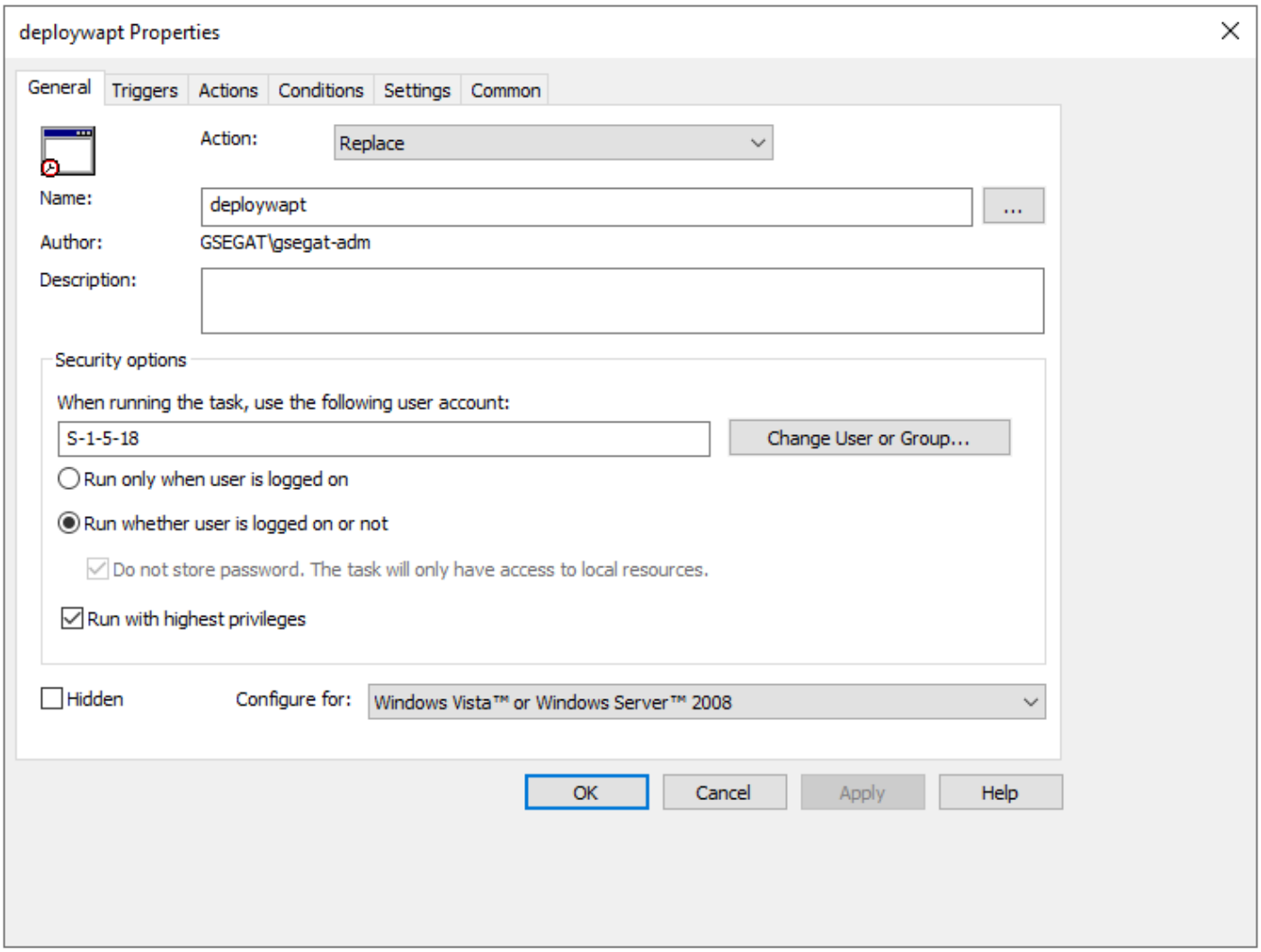


Fig. 15: General tab in the Properties window in RSAT

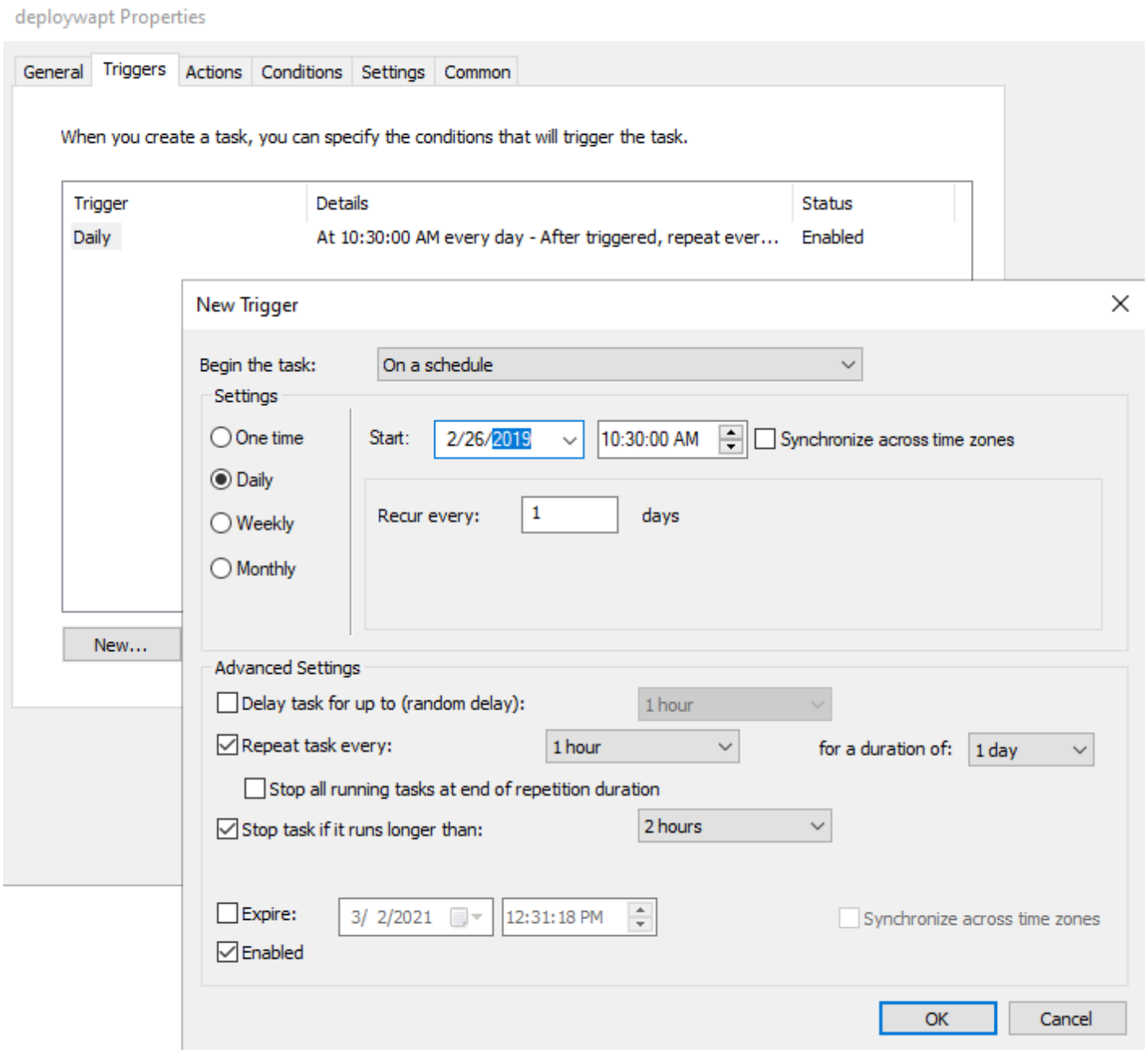


Fig. 16: Trigger tab in the Properties window in RSAT

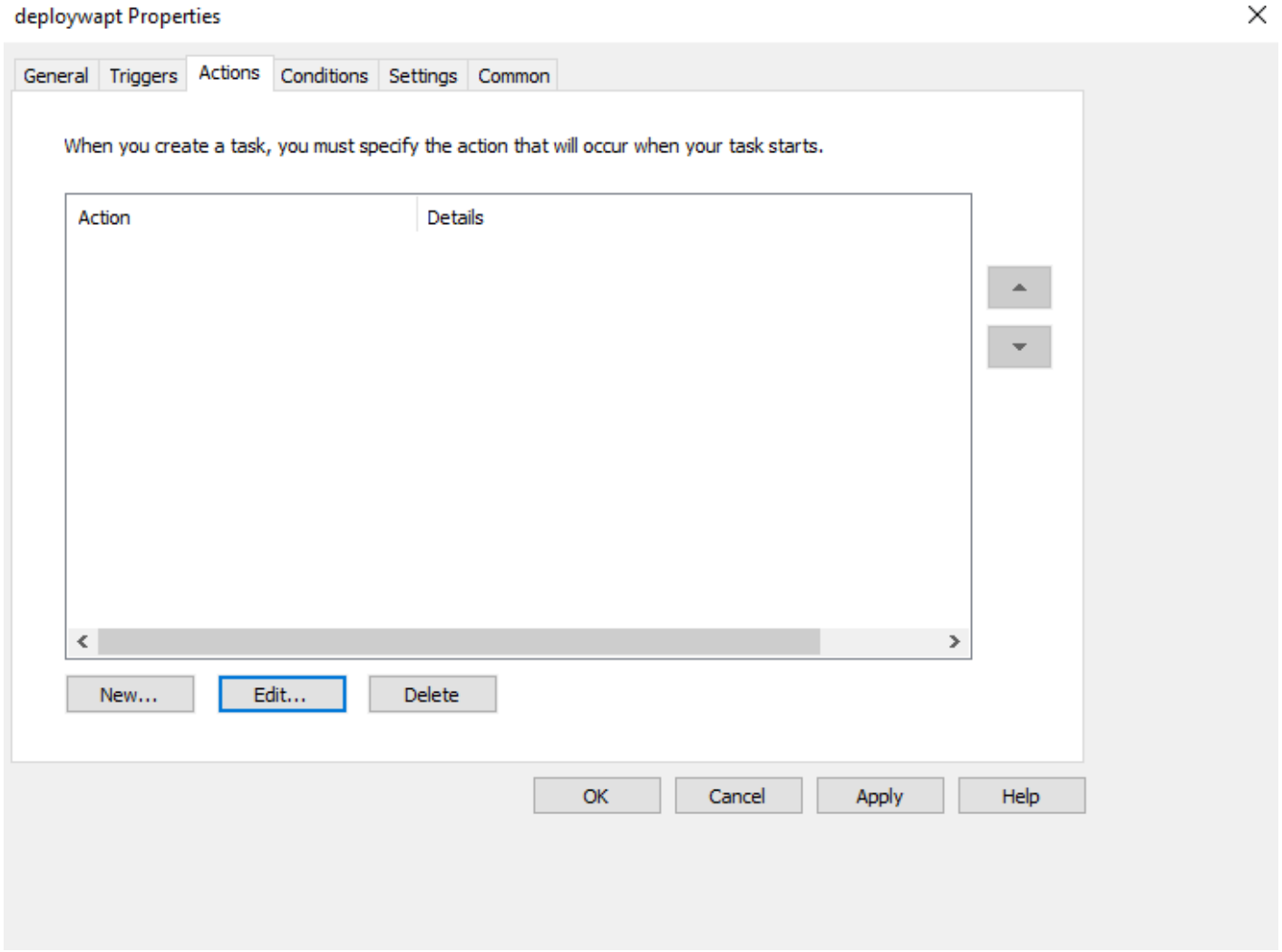


Fig. 17: Actions tab in the Properties window in RSAT

Fig. 18: Actions tab in the Properties window in RSAT

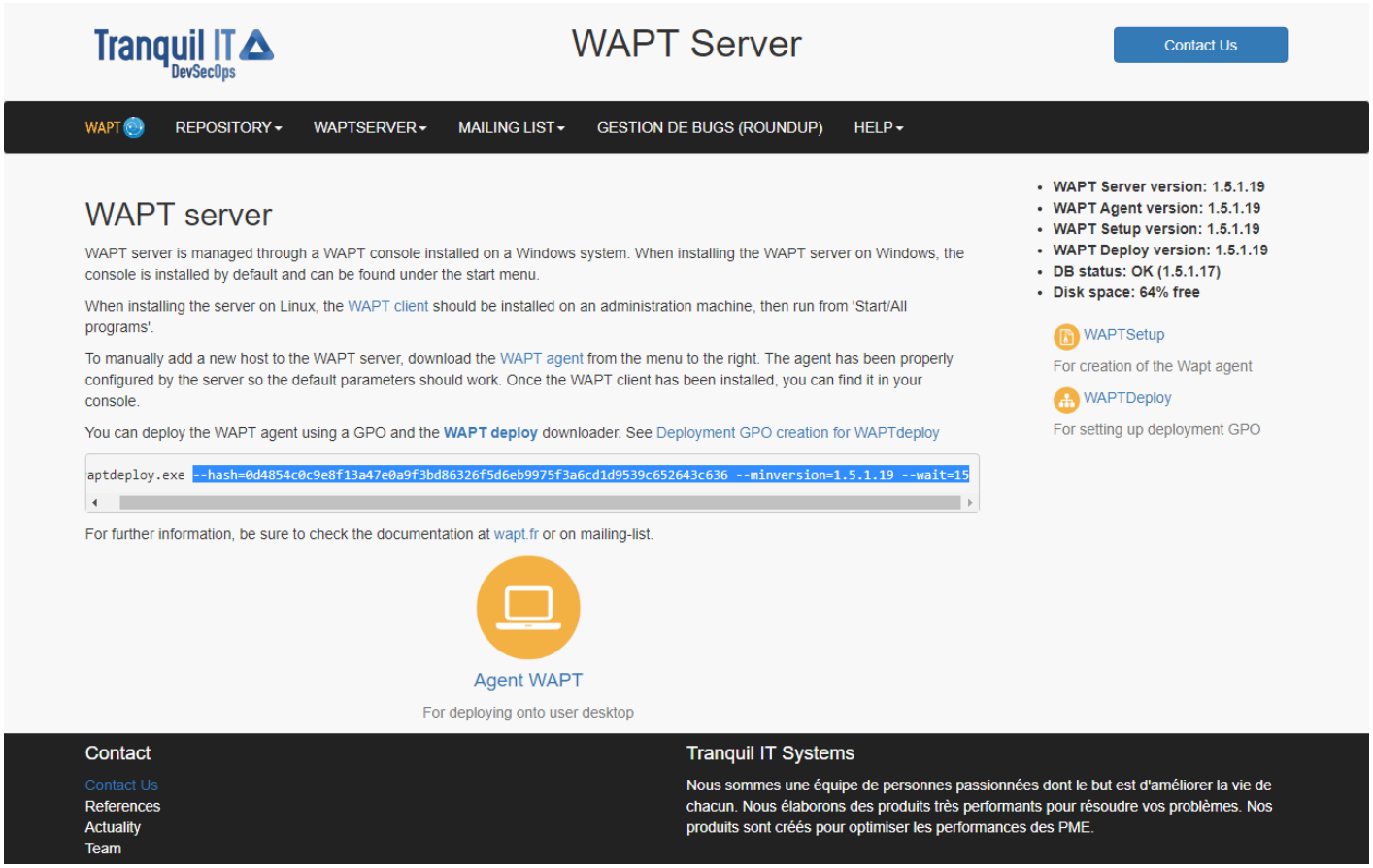


Fig. 19: Web console of the WAPT Server

- Copy the required parameters and change `waptsetupurl` to `C:\Temp\waptagent.exe`.

```
--hash=checksum WaptAgent --minversion=2.4 --wait=15 --waptsetupurl=C:\Temp\waptagent.exe
```

Table 7: Description of available options for the WAPT Deployment utility

Options	Description
`--force`	Installs waptagent.exe even if not needed
`--hash` = <sha256hash>	Checks that the downloaded waptagent.exe setup sha256 hash matches the hash.
`--help`	Displays the options.
`--minversion` = 2.4.0	Installs waptagent.exe if installed version is less than minversion.
`--tasks` = autorunTray,installService,installredist2008,autoUpgradePolicy	If given, passes this arguments to the /TASKS options of the waptagent installer. Default = installService, installredist2008, autoUpgradePolicy
`--repo_url` = https://srvwapt.mydomain.lan/wapt	Defines the location of the repository to get the `waptagent.exe`.
`--setupargs` = <options>	Adds arguments to the command line of waptagent.exe.
`--wait` = <minutes>	Defines the maximum allowed time for running and pending tasks to complete if the WAPT service is running before installing.
`--waptsetupurl` = https://srvwapt.mydomain.lan/wapt/waptagent.exe	Defines an explicit location to download setup executable. This can be a local path (default=:file:*<repo_url>/waptagent.exe*).

- Go on to the *Settings* tab.
- In the *Settings* tab, only check *Run task as soon as possible after a scheduled start is missed*.

Hint: To verify that the GPO is working, you can run the **`gpupdate /force`** command and verify that the scheduled task is present on the computer by launching **`Task Scheduler`** as a Local Administrator.

28.2 Deploying the WAPT Agent on Linux and macOS

Note: To install WAPT on a Linux client, the minimal requirements are:

- 512Mo Ram;
- 1 CPU;
- 300Mo Drive space (without package cache).

The procedure depends on your operating system:

Debian / Ubuntu based distributions

Hint: The WAPT Agent for Debian has been tested on Debian 9, 10, 11 and 12.

The WAPT Agent for Ubuntu has only been tested on Ubuntu Bionic and Ubuntu Focal.

- Update the underlying distribution and check that apt https transport is installed

```
sudo apt update && apt upgrade -y
sudo apt install apt-transport-https lsb-release gnupg -y
```

- Retrieve the key *.gpg*, add it to the Tranquil IT repository and install the WAPT Agent.

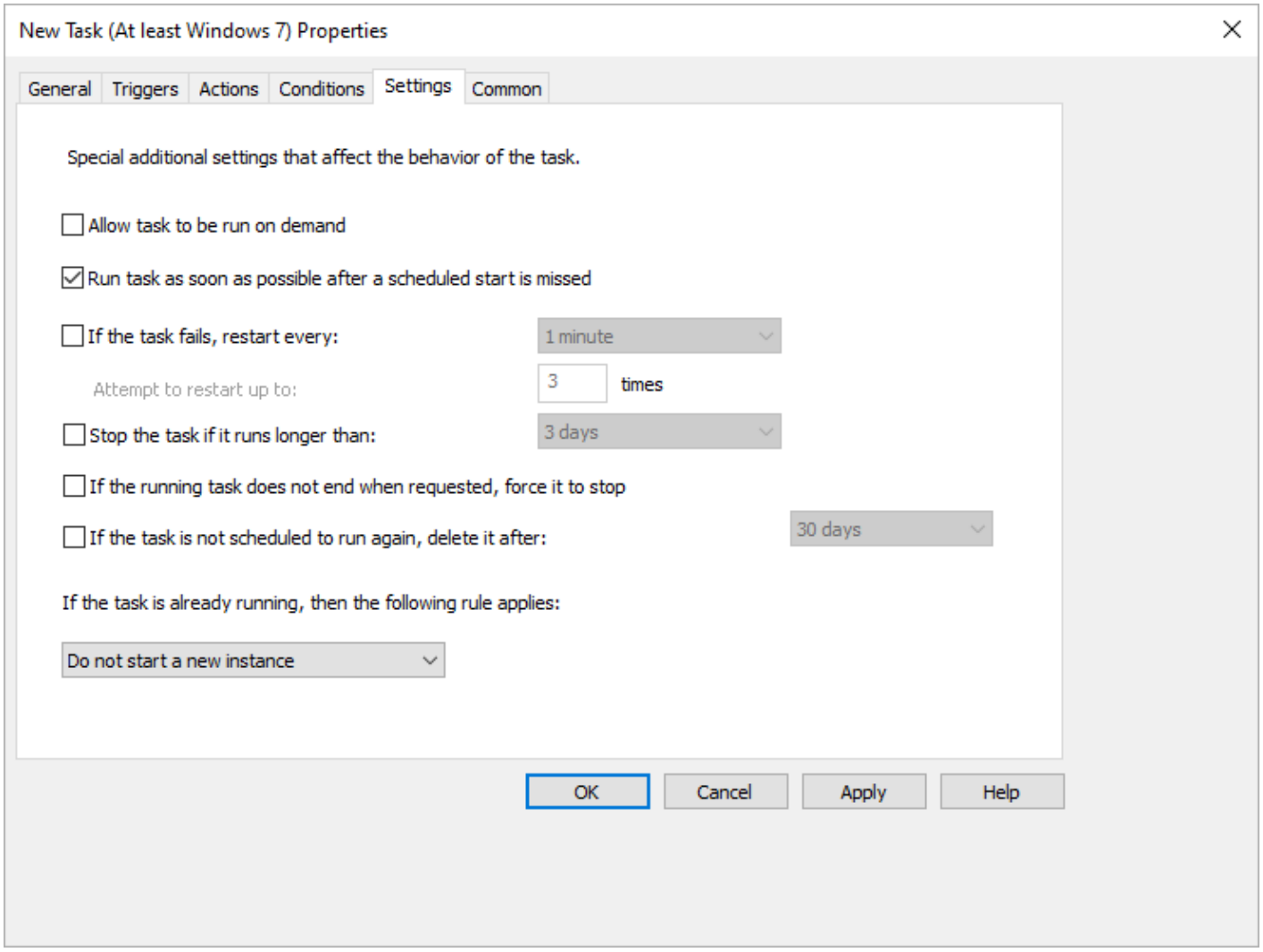


Fig. 20: Settings tab in the Properties window in RSAT

```
sudo wget -qO- https://wapt.tranquil.it/$(lsb_release -is)/tiswapt-pub.gpg | tee /usr/share/
↪keyrings/tiswapt-pub.gpg > /dev/null
sudo echo "deb [signed-by=/usr/share/keyrings/tiswapt-pub.gpg] https://wapt.tranquil.it/$(lsb_
↪release -is)/wapt-2.4/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/wapt.list

export DEBIAN_FRONTEND=noninteractive
sudo apt update
sudo apt install tis-waptagent -y
unset DEBIAN_FRONTEND
```

RedHat based distributions

Hint: The WAPT Agent for Redhat based system has been tested on Redhat 7/8/9 and derivatives on x86_64 platforms.

- Update the underlying distribution.

```
yum update
```

- Retrieve the key `.gpg` and configure the WAPT repository.

```
RH_VERSION=$(cat /etc/system-release-cpe | awk -F: '{ print $5}')
wget -q -O /tmp/tranquil_it.gpg "https://wapt.tranquil.it/redhat${RH_VERSION}/RPM-GPG-KEY-TISWAPT-$
↪{RH_VERSION}"; rpm --import /tmp/tranquil_it.gpg

cat > /etc/yum.repos.d/wapt.repo <<EOF
[wapt]
name = WAPT Server Repo
baseurl = https://wapt.tranquil.it/redhat${RH_VERSION}/wapt-2.4/
enabled = True
gpgcheck = True
EOF
```

- install the WAPT Agent using yum:

```
yum install tis-waptagent
```

macOS

Warning: The WAPT agent for macOS is currently only available in the WAPT Enterprise version.

Hint: The WAPT Agent has only been tested on **Intel architecture** and **Apple Silicon M1** processors:

- Mojave (10.14);
- Catalina (10.15);
- Big Sur (11.x);
- Monterey (12.x).
- Ventura (13.x).

- Download and install the WAPT Agent (note: the hash string may change, to get the latest, point your browser on the url https://wapt.tranquil.it/wapt/releases/wapt-2.4/). Choose the version depending on your processor architecture (intel or m1):

```
# for mac m1
curl -o tis-waptagent-2.4.0.14143-9847ee8b-macos-all-arm64.pkg https://wapt.tranquil.it/wapt/
↪releases/wapt-2.4/tis-waptagent-2.4.0.14143-9847ee8b-macos-all-arm64.pkg
# for mac intel
curl -o tis-waptagent-2.4.0.14143-9847ee8b-macos-all-x86_64.pkg https://wapt.tranquil.it/releases/
↪wapt-2.4/tis-waptagent-2.4.0.14143-9847ee8b-macos-all-x86_64.pkg

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

28.2.1 Installing the WAPT Agent configuration file

Before installing the WAPT Agent configuration file, you have to create a *initial config for you agent* in your WAPT Console.

Warning: The WAPT Agent configuration wizard is only available on WAPT Entreprise Edition. To configure Linux WAPT Agent, please refer to the *manual WAPT Agent configuration method*.

When done, copy the command with the *Copy installation command*.

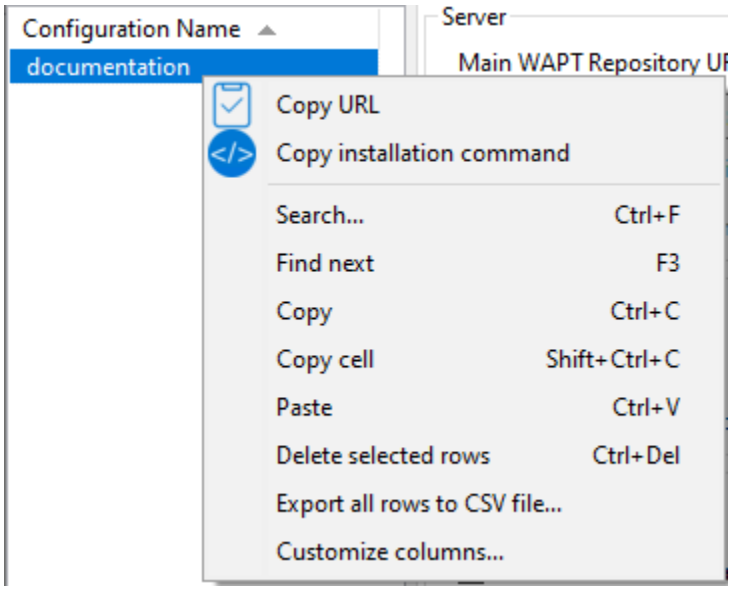


Fig. 21: Menu list showing the *Copy installation command*

Then use this copied command prompt on the Linux / macOS agent.

```
wapt-get add-config-from-url https://srvwapt.mydomain.lan/wapt/conf.d/default_
↪f0288df2131b8dce667b8c34b9999959bdc2d253b3934fcb3be2eabad8a50021.json␣
↪f0288cf2131b9dce667b8c34b9999959bdc2d253b3934fcb3be2eabad8a50020
```

Finally, execute the following command to register the Linux / macOS host with the WAPT Server:

```
sudo wapt-get register
```

When you have modified the configuration of the WAPT Agent, you should restart the WAPT Agent using the following command:

```
sudo wapt-get restart-waptservice
```

Feature matrix

There are some features that are not currently available on Linux and macOS:

- installing updates on shutdown (WAPT Exit);
- any Windows specific feature.

Particularities with domain functionality

On Linux:

- Testing was carried out with sssd with an Active Directory domain and kerberos authentication.
- To integrate a host in the Active Directory domain, you can choose to follow this documentation.
- 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 a member of.

Attention: We have noticed that the kerberos LDAP query does not work if the reverse DNS record is not configured correctly for the domain controllers. These records **MUST** therefore be created if they do not exist.

28.3 Manual method to configure the WAPT Agent running on Linux / macOS

Attention: Please, see the new method *to deploy configuration file* instead if you are using WAPT Entreprise Edition.

28.3.1 Creating the WAPT Agent configuration file

Use the WAPT Server FQDN address for the `repo_url` and the `wapt_server` arguments.

```
sudo cat > /opt/wapt/wapt-get.ini <<EOF
[global]
repo_url = https://srvwapt.mydomain.lan/wapt
wapt_server = https://srvwapt.mydomain.lan
use_hostpackages = True
use_kerberos = False
verify_cert = False
EOF
```

28.3.2 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 host in `C:\Program Files (x86)\wapt\ssl\`.

Copy your certificate(s) in `/opt/wapt/ssl` using **WinSCP** or **rsync** if you are deploying on Linux or macOS.

28.3.3 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 or macOS Agent.

The certificate should be located on your Windows host in `C:\Program Files (x86)\wapt\ssl\server\`.

- Copy your certificate(s) in `/opt/wapt/ssl/server/` using **WinSCP** or **rsync** if you are deploying on Linux or macOS.
- Then, modify in the `/opt/wapt/wapt-get.ini` configuration file the path to your certificate.
- And give the absolute path of your certificate.

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

Hint: Change the *.crt* file with your certificate name.

CHAPTER

TWENTYNINE

UPDATING THE WAPT AGENT

29.1 Updating on Windows

For each WAPT Server's *upgrade*, you will have to upgrade the WAPT Agents.

To do so, you have to *generate the WAPT Agent* and deploy it.

29.1.1 Manually

You can do that manually *by following this documentation on installing the WAPT Agent*.

Hint: It is the only upgrade solution available for now for macOS and Linux.

29.1.2 Via waptupgrade

While you *generate* the WAPT Agent, package named `waptupgrade` is created.

This package is a standard WAPT package designed to upgrade the WAPT Agents on remote hosts.

Hint: For now, `waptupgrade` only works for Windows. Waptupgrade does not upgrade the WAPT Agent if the WAPT Server version and the WAPT Agent version are the same.

Upgrading the WAPT Agents using the `waptupgrade` package is a two step process:

- First the package copies the `waptsetup.exe` file on the client computer and creates a scheduled task that will run **`waptsetup.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 WAPT Server shows the package installation as *OK*, with the correct version installed, but the inventory will still show the old version as the WAPT Agent is not yet updated.
- After two minutes, the scheduled task starts and runs **`waptsetup.exe`** with a predefined configuration created in the WAPT Console. This new method keeps the **`waptsetup.exe`** signed by Tranquil IT, but the WAPT Agent configuration will come from the WAPT Server. **`waptsetup.exe`** shutdowns the local WAPT service, upgrades WAPT locally, and then restarts the WAPT service. The scheduled task is then automatically removed and the WAPT Agent starts sending back its inventory to the WAPT Server. From then on, the inventory on the WAPT Server will show the new version of the WAPT Agent.

It is recommanded to install `waptupgrade` on all hosts for the WAPT Agents to update automatically.

29.2 Updating on Linux and MacOS

For each WAPT Server's *upgrade*, you will have to upgrade the WAPT Agents.

To do so, you have to *generate the WAPT Agent* and deploy it.

29.2.1 Manually

You can install manually the Linux / macOS Agent *by following this documentation on installing the WAPT Agent*.

Hint: It is the only upgrade solution available for now for macOS and Linux

CHAPTER

THIRTY

UNINSTALLING THE WAPT AGENT FROM CLIENTS

30.1 Windows

If you need to uninstall the WAPT Agent from computers, the uninstaller is automatically created in the WAPT install location. By default the uninstaller is locatedin `C:\Program Files (x86)\wapt\unins000.exe`.

- A 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
```

Table 1: Complete list of command-line arguments for **unins000.exe**

Settings	Description
/VERYSILENT	Launches **unins000.exe** silently.
/purge_wapt_dir = True	Purges the WAPT directory (removes all folders and files).

- It is possible to use a WAPT package to achieve the same result.

```
# -*- coding: utf-8 -*-
from setuphelpers import *

def install():

  print("Creation of the task")
  task = create_onetime_task('removewapt', "unins000.exe", "/VERYSILENT /purge_wapt_dir = True")
  print(task)
```

30.1.1 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 WAPT package to push before uninstalling the WAPT Agent:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

def install():
  print('Disable WAPT WUA')
  inifile_writestring(WAPT.config_filename,'waptwua','enabled','false')

  print('DisableWindowsUpdateAccess registry to 0')
  registry_set(HKEY_LOCAL_MACHINE,r'Software\Policies\Microsoft\Windows\WindowsUpdate',
↪'DisableWindowsUpdateAccess',0,REG_DWORD)

  print('AUOptions registry to 0')
  registry_set(HKEY_LOCAL_MACHINE,r'SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto␣
↪Update','AUOptions',0,REG_DWORD)

  print('Enable wuauserv')
  run_notfatal('sc config wuauserv start= auto')
  run_notfatal('net start wuauserv')

  print('Reload WAPT configuration')
  WAPT.reload_config_if_updated()
```

30.2 Linux

- The default uninstall of a WAPT Agent can be achieved with the following command, depending on your Linux distribution:

Debian / Ubuntu

```
apt remove --purge tis-waptagent
```

Redhat and derivatives

```
yum remove tis-waptagent
```

- An additional step can be done using these commands (WIP).

Debian / Ubuntu

```
rm -f /opt/wapt/
rm /etc/apt/sources.list.d/wapt.list
```

Redhat and derivatives

```
rm -f /opt/wapt/
rm /etc/yum/yum.repos.d/wapt.list
```

30.3 MacOS

The default uninstall of a WAPT Agent can be achieved with the following command:

```
pkgutil --only-files --files it.tranquil.waptservice > file_list
sudo pkgutil --forget it.tranquil.waptservice
```

CHAPTER

THIRTYONE

USING WAPT WITH THE COMMAND LINE

The WAPT Agent provides a command line interface utility **wapt-get**.

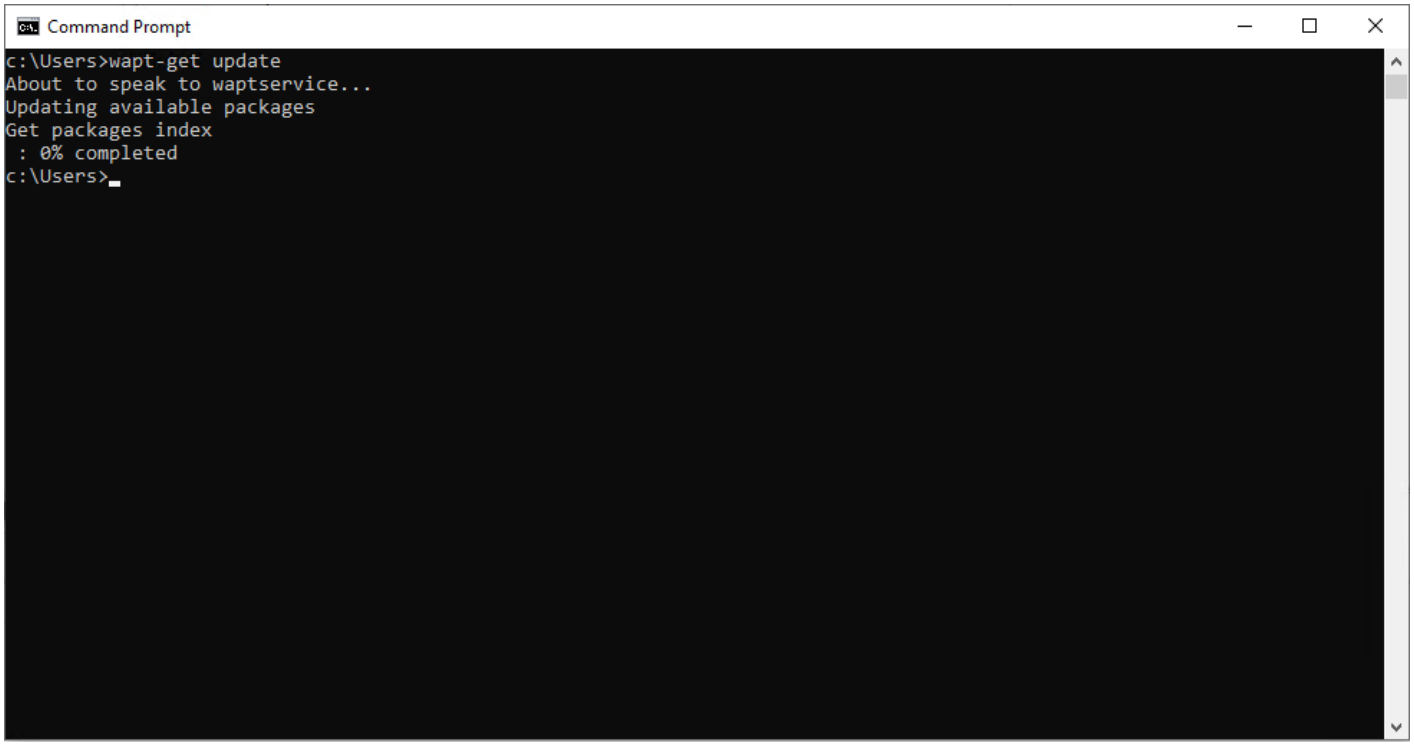


Fig. 1: The Windows Command Line utility

Note:

- By default, command-line actions in WAPT are executed with the rights of the user who launched the **cmd.exe**.
- If the **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`.

Note: Each commands that takes a package name as a parameter can also take the unique *package_uuid* of the package as a parameter (**wapt-get install**, **wapt-get forget**, etc.). Using a GUID allows to specify a unique package without ambiguity on its architecture or version. The *package_uuid* is listed in the output of **wapt-get list** and **wapt-get search**. For example:

31.1 Using the more common functions in WAPT with the command line

31.1.1 wapt-get install

The **wapt-get install** command launches the installation of a WAPT package.

To install Mozilla Firefox, the command is **wapt-get install tis-firefox**.

It is possible to install several packages at once:

```
wapt-get install package1 package2
```

If the package has not been downloaded to cache, **wapt-get install** will first download the package to cache, then the WAPT Agent will install the package.

Attention: Installing a WAPT package with **wapt-get install** does not add the package as a dependency to the host.

The package is installed on the host, but if the computer is re-imaged, the package will not be reinstalled automatically.

The command **wapt-get install tis-firefox** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
installing WAPT packages tis-firefox
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 14121562 /␣
↪54313787 (26%) (24624 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 33131357 /␣
↪54313787 (61%) (29414 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 50511741 /␣
↪54313787 (93%) (30412 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 54313787 /␣
↪54313787 (100%) (30360 KB/s)
Installing tis-firefox(=94.0.1-106)
Installing: Firefox_Setup_94.0.1.exe
Waiting for key key Mozilla Firefox 94.0.1 (x64 en-US) to appear in Windows registry
Delete C:\Program Files (x86)\wapt\cache\tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt

Results:

=== install packages ===
```

(continues on next page)

(continued from previous page)

```
  tis-firefox [x64_en_PROD]     | tis-firefox (94.0.1-106)                    | tis-firefox (50.0.2-
↪73)
```

31.1.2 wapt-get update

The **wapt-get update** command allows 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 titles have been added on the repository, they become available for download by the WAPT Agent.

Note: The **wapt-get update** command does not download packages, it only updates the local database of packages.

The command **wapt-get update** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Update package list from https://srvwapt.mydomain.lan/wapt, https://srvwapt.mydomain.lan/wapt-host
Total packages: 8
Added packages:

Removed packages:

Discarded packages count: 6
Pending operations:
  install:
  upgrade:
  additional:
  remove:
  immediate_installs:
Repositories URL :
  https://srvwapt.mydomain.lan/wapt
  https://srvwapt.mydomain.lan/wapt-host
```

31.1.3 wapt-get upgrade

The command **wapt-get upgrade** launches the installation of WAPT packages waiting to be upgraded or waiting to be installed.

The local WAPT Agent first downloads its WAPT packages to the local cache, then the WAPT Agent installs them.

Hint: It is strongly advised to launch a **wapt-get update** command before launching a `wapt-get upgrade` command.

Without previously launching a **wapt-get update**, the WAPT Agent will install nothing.

The command **wapt-get upgrade** returns:

```
Installing tis-mumble
Shutting down Mumble
installing Mumble 1.2.3

=== install packages ===
tis-mumble
```

31.1.4 wapt-get remove

The **wapt-get remove <package name>** command removes the listed WAPT packages from the host.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

To remove Mozilla Firefox, the command is **wapt-get remove <prefix>-firefox**.

> **Attention:** Removing a WAPT package with **wapt-get remove** does not remove the package dependency on the host.
>
> **The package will effectively be uninstalled from the host, but it will automatically be reinstalled on the next upgrade.**
>
> To completely remove a package from a host, do a **wapt-get 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:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Removing tis-firefox ...

Waiting for the removal of key key Mozilla Firefox 94.0.1 (x64 en-US) from Windows registry
=== Removed packages ===
  tis-firefox
```

31.1.5 wapt-get uninstall

The **wapt-get uninstall [<package name>]** command uninstalls the listed packages from the host if a `def uninstall()` function exists in the `setup.py` files of the listed packages.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

> **Attention:** Running the uninstall package function does not delete the cached package on the host.

The command **wapt-get uninstall tis-adwcleaner** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
 Uninstalling tis-adwcleaner ...
 None
 Uninstallation done
```

31.1.6 wapt-get forget

The **wapt-get forget <package name>** command removes the package from the local database so that the lifecycle of the software title or the configuration is no longer managed by WAPT.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

Attention: Forgetting the WAPT package does not uninstall the software title or the configuration associated with the WAPT package.

The command **wapt-get forget tis-adwcleaner** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini

=== Packages removed from status ===
  tis-adwcleaner
```

31.1.7 wapt-get audit

The **wapt-get audit [<package name>]** command runs the audit function for the listed packages if a `def audit()` function exists in the `setup.py` files of the listed packages.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

Also, the **wapt-get audit ALL** command runs the audit function for all packages installed on the host.

The command **wapt-get audit tis-firefox** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Auditing tis-firefox ...
Auditing tis-firefox
OK: Uninstall Key Mozilla Firefox 94.0.1 (x64 en-US) in Windows Registry.
tis-firefox -> OK
```

31.1.8 wapt-get show

The **wapt-get show <package name>** command displays informations stored in the `Packages` index file.

If several versions of a WAPT package are available on the WAPT repository, each version of the package will be displayed.

The command **wapt-get show tis-7zip** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Display package control data for tis-7zip

package           : tis-7zip
version           : 19.00-25
architecture      : x64
section           : base
priority          : optional
```

(continues on next page)

(continued from previous page)

```
name              : 7-Zip
categories        : Utilities
maintainer        : WAPT Team,Tranquil IT,Jimmy PELÉ
description       : 7-Zip is a free and open-source file archiver with a high compression ratio
depends           :
conflicts         :
maturity          : PROD
locale            : all
target_os         : windows
min_wapt_version  : 1.7
sources           : https://www.7-zip.org/download.html
installed_size    :
impacted_process  : 7zFM,7z,7zG
description_fr    : 7-Zip est un logiciel gratuit et open source pour archiver des fichiers avec un␣
↪taux de compression élevé
description_pl    :
description_de    : 7-Zip ist ein Datenkompressionsprogramm mit einer hohen Kompressionsrate
description_es    : 7-Zip es un archivador de ficheros con una alta relación de compresión
description_pt    : O 7-Zip é um compactador de arquivos com alta taxa de compressão
description_it    :
description_nl    :
description_ru    : 7-Zip
audit_schedule    :
editor            : Igor Pavlov
keywords          : 7zip,7,zip,7-zip,file,archiver,high,compression,ratio
licence           : LGPL
homepage          : https://www.7-zip.org/
package_uuid      : dc66ccd1-d987-482e-b792-04e89a3803f7
valid_from        :
valid_until       :
forced_install_on :
changelog         : https://www.7-zip.org/history.txt
min_os_version    : 5.0
max_os_version    :
icon_sha256sum    : eddc038d3625902b6ddeaabd13dd91529e8d457ffbd0c554f96d343ae243a67a
signer            : documentation
signer_fingerprint: 3f2c0a02231a36eafa1f67905f5c083e4b66cb59942f69cbd231d778a1a25b3d
signature         : QzhPeZFrRbjcGzfqRpoWsDP9Plaz6BBVlL3adq/MRM19D61+Aez/
↪JiA8skriCgwSErJXbxOPfxusVqqIpEtyoqh/RlRcnmgCQqk2Fig4gmxpz0rHKokukPQlRk+HdC/
↪uByxSjfp9oXuB3PVG2PZAFifjVBtjEX2QmV+OY6NdMI9dtkxCsn1Xotn2qhu2bwbJWQ0s51rD9emWuQR7l/
↪8WXl+HoquuRho4aCeAOYd6Nta9ktVSR2FM6OO5ZeUOg4fsnMg+hwp2MlDOmBHX37aJm3hLYkGP2xWjpL9YDDxI7ruRXSHyT7YmbILrS0h1m
signature_date    : 2021-11-19T16:15:42.019196
signed_attributes : package,version,architecture,section,priority,name,categories,maintainer,
↪description,depends,conflicts,maturity,locale,target_os,min_wapt_version,sources,installed_size,
↪impacted_process,description_fr,description_pl,description_de,description_es,description_pt,
↪description_it,description_nl,description_ru,audit_schedule,editor,keywords,licence,homepage,
↪package_uuid,valid_from,valid_until,forced_install_on,changelog,min_os_version,max_os_version,
↪icon_sha256sum,signer,signer_fingerprint,signature_date,signed_attributes
filename          : tis-7zip_19.00-25_x64_windows_0f4137ed1502b5045d6083aa258b5c42_5.0_PROD_
↪a10c57d7848cf7b145d6cd64bf4d5389.wapt
```

(continues on next page)

(continued from previous page)

```
size              : 1704227
md5sum            : a10c57d7848cf7b145d6cd64bf4d5389

OK Package control signature checked properly by certificate documentation (fingerprint:␣
↪3f2c0a02231a36eafa1f67905f5c083e4b66cb59942f69cbd231d778a1a25b3d )
```

Note:

```
WARNING: control data signature can not be validated with certificates [<SSLCertificate cn=
↪'documentation' fingerprint=3f2c0a issuer='documentation' validity=2021-11-19 - 2031-11-17 Code-
↪Signing=True CA=True>]
```

If this message appears, it is because the certificate is not trusted.

If you want to check the package, download it to cache and run the **wapt-get show** on the local package.

For example:

```
wapt-get download tis-7zip
wapt-get show "C:\Program Files (x86)\wapt\cache\tis-7zip_19.00-25_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_5.0_PROD_a10c57d7848cf7b145d6cd64bf4d5389.wapt"
```

31.1.9 wapt-get show-params

The **wapt-get show-params <package name>** command returns a list of *parameters* that would be passed on to the **wapt-get install <package name> --params=PARAMS** command.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

The command `wapt-get show-params tis-7zip` returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
tis-7zip : {True, 'documentation': True}
```

31.1.10 wapt-get show-log

The **wapt-get show-log <package name>** command return the latest audit logs stored in the local sqlite database of the WAPT Agent.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

The command **wapt-get show tis-7zip** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Package: tis-7zip (21.06-34) PROD
--------------------
Status: OK

Installation log:
```

(continues on next page)

(continued from previous page)

```
-------------------
Installing: 7z2106-x64.msi
Waiting for key key {23170F69-40C1-2702-2106-000001000000} to appear in Windows registry

Installation Parameters:
-------------------
{}

Last audit:
-------------------
Status: OK
Date: 2022-01-06T10:32:38.698272

Output:
Auditing tis-7zip
OK: Uninstall Key {23170F69-40C1-2702-2106-000001000000} in Windows Registry.

Next audit on: 2022-01-06T10:32:38.698272
```

31.1.11 wapt-get search

The **wapt-get search** command allows to search for one or more package in the repositories.

> **Warning:** This command returns only WAPT packages available for the host executing the command, according to the locale, operating system, architecture or maturity of the host.
>
> If a WAPT package with another locale, operating system, architecture or maturity exists in the repository, it will not be listed.

The search command takes one keyword argument.

The command **wapt-get search "Firefox"** returns (for example):

```
status     package            version    target_os       architecture maturity locale description
↪                                                      repo
---------- ------------------ ---------- --------------- ---- ----- --- --------------------------
↪------------------------------------------------------ ----------
-          tis-firefox        94.0.2-106 windows         x64  PROD  fr  Mozilla Firefox est un
↪navigateur web gratuit et open source                    wapt
I          tis-config-firefox 68.3-6     windows         all  PROD      Configuration for Mozilla
↪Firefox - The package will not have any effect if an*  wapt
I          tis-firefox-esr    91.3.0-105 windows         x64  PROD  fr  Mozilla Firefox Extended
↪Support Release (ESR) est une version officielle de*    wapt
```

Value	status	package	version	target_os	architecture	maturity	locale	description	repo
Description	Packages installation status	Name of package	Version of package	Target OS (if defined)	Architecture of CPU (if defined)	Maturity of package (if defined)	Locale of package (if defined)	Description of package	Folder of package on the WAPT Server

Note: The value of *status* defines the installation status as follows:

- - for not installed.
- I for installed.

31.1.12 wapt-get download

The **wapt-get download <package name>** command downloads the WAPT package to the local cache.

The command **wapt-get download tis-7zip** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Downloading packages tis-7zip(=19.00-25)
https://srvwapt.mydomain.lan/wapt/tis-7zip_19.00-25_x64_windows_0f4137ed1502b5045d6083aa258b5c42_5.
↪0_PROD_a10c57d7848cf7b145d6cd64bf4d5389.wapt : 1704227 / 1704227 (100%) (11804 KB/s)

Downloaded packages:
  C:\Program Files (x86)\wapt\cache\tis-7zip_19.00-25_x64_windows_0f4137ed1502b5045d6083aa258b5c42_
↪5.0_PROD_a10c57d7848cf7b145d6cd64bf4d5389.wapt
```

31.1.13 wapt-get download-upgrade

The **wapt-get download-upgrade** command downloads packages to be upgraded to the local WAPT cache.

The command **wapt-get download-upgrade** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 18466658 /␣
↪54313787 (34%) (32089 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 36390179 /␣
↪54313787 (67%) (32693 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 52684289 /␣
↪54313787 (97%) (31564 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 54313787 /␣
↪54313787 (100%) (30747 KB/s)
```

(continues on next page)

(continued from previous page)

```
=== downloaded packages ===
  C:\Program Files (x86)\wapt\cache\B8D346E7-DDDB-0013-5A8A-425CF3B6199E.wapt
  C:\Program Files (x86)\wapt\cache\tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt
```

31.1.14 wapt-get list

The **wapt-get list** command lists WAPT packages that are installed on the computer.

The command **wapt-get list** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
package                              version    install_status install_date     description
↪                                                              package_uuid
---------------------------------- ---------- ---------- ---------------- ------------------------
↪------------------------------------------------------ ----------------------------------
tis-7zip                           21.06-34   OK         2021-12-10T14:57 7-Zip is a free and open-
↪source file archiver with a high compression ratio      717a30cc-0d44-42d1-9538-0f2f298d8603
tis-firefox                        94.0.1-106 OK         2021-12-10T14:58 Mozilla Firefox is a
↪free and open-source web browser                            5a91f54a-3e27-44cf-a2b6-6b84012aa3a2
```

package	version	install status	install_date	description	package_uuid
Package Name	Package Version	Installation status	Date and time of installation	Package description	Unique UUID of the package

31.1.15 wapt-get list-upgrade

The **wapt-get list-upgrade** command lists WAPT packages that need to be upgraded on the host.

The command **wapt-get list-upgrade** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini

=== upgrade packages ===
tis-notepadplusplus(=8.2-10)
```

31.1.16 wapt-get -S tasks

The **wapt-get -S tasks** command checks whether some tasks are running or are pending in queue.

The command **wapt-get -S tasks** returns:

```
About to speak to waptservice...
Running task 14: Uninstall of tis-vlc (task #14), status:
```

31.2 Using special Command Lines with WAPT

31.2.1 wapt-get restart-waptservice

The **wapt-get restart-waptservice** restarts the **waptservice** on Windows, Linux and macOS.

31.2.2 wapt-get add-config-from-url

The **wapt-get add-config-from-url <filelink> <sha256hashfile>** command fetches a *json* dynamic configuration file from the specified url and places the file into the conf.d directory under the wapt installation folder.

The <sha256hashfile> parameter is optional.

```
C:\Users\administrator>wapt-get add-config-from-url https://srvwapt.mydomain.lan/wapt/conf.d/
↪default_config_55863a6b54a47255097b6403731b36de716fc7ee9ec824bffad36d5fdc49b6b5.json
New config installed as C:\Program Files (x86)\wapt\conf.d\default.json
```

31.2.3 wapt-get add-config-from-file

The **wapt-get add-config-from-file <filepath>** command adds a *json* dynamic configuration file into the directory conf.d under the wapt installation folder.

The path to the *json* dynamic configuration file is defined by <filepath>.

31.2.4 wapt-get add-config-from-base64

The **wapt-get add-config-from-file <base64 file>** command adds a *json* dynamic configuration file into the directory conf.d under the wapt installation folder.

The path to the *json* dynamic configuration file is defined by <base64 file>.

31.2.5 wapt-get remove-config

The **wapt-get remove-config <config-name>** command removes the specified *json* dynamic configuration files from the conf.d folder under the wapt installation folder.

31.2.6 wapt-get list-config

The **wapt-get list-config** command lists installed *json* dynamic configuration files that are present in the conf.d folder under the wapt installation folder.

```
C:\Users\administrator>wapt-get list-config
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini

config files are located in C:\Program Files (x86)\wapt\conf.d

* config_base
```

31.2.7 wapt-get list-available-config

The **wapt-get list-available-config** command lists all available *json* dynamic configuration files that are present on the WAPT Server and will show the command to install them.

Listing the available configurations requires that the user be authenticated.

```
C:\Users\administrator>wapt-get list-available-config
Server: https://srvwapt.mydomain.lan
Server UUID: 32464dd6-c261-11e8-87be-cee799b43a00
Server CABundle: 0

Waptserver https://srvwapt.mydomain.lan Admin User () :admin
Waptserver Password: **************

default_config : wapt-get add-config-from-url https://srvwapt.mydomain.lan/wapt/conf.d/default_
↪config_55863a6b54a47255097b6403731b36de716fc7ee9ec824bffad36d5fdc49b6b5.json
     Server: https://srvwapt.mydomain.lan
     Repo: https://srvwapt.mydomain.lan/wapt

default : wapt-get add-config-from-url https://srvwapt.mydomain.lan/wapt/conf.d/default_
↪91ab2cd1901b5e36214224229c3461e49e65f7b065ea6b0eb16bd83c7fcdda57.json
     Server: https://srvwapt.mydomain.lan
     Repo: https://srvwapt.mydomain.lan/wapt

mac_config : wapt-get add-config-from-url https://srvwapt.mydomain.lan/wapt/conf.d/mac_config_
↪2720657c276cbc0ee14734e68fbd0fadc4dea3171625406e10cd9828631e5c72.json
     Server: https://srvwapt.mydomain.lan
     Repo: https://srvwapt.mydomain.lan/wapt
```

31.2.8 wapt-get clean

The **wapt-get clean** command removes packages from the cache folder.

The command is launched after each **wapt-get upgrade** to save disk space.

The command **wapt-get clean** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Removed files:
C:\Program Files (x86)\wapt\cache\tis-mumble_1.2.3-1_all.wapt
C:\Program Files (x86)\\wapt\cache\tis-vlc_1.2.3-2_all.wapt
```

31.2.9 wapt-get upgradedb

The **wapt-get upgradedb** command upgrades the local WAPT database schema if necessary.

The command **wapt-get upgradedb** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
WARNING upgrade db aborted: current structure version 20210420 is newer or equal to requested␣
↪structure version 20210420
No database upgrade required, current 20210420, required 20210420
```

31.2.10 wapt-get add-upgrade-shutdown - wapt-get remove-upgrade-shutdown

These 2 commands modify the file `C:\Windows\System32\GroupPolicy\Machine\Scripts\scripts.ini` on Windows devices.

- The **wapt-get add-upgrade-shutdown** command adds a **waptexit** local security policy object, enabling the execution of **waptexit** at system shutdown.

The command **wapt-get add-upgrade-shutdown** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
0
```

The `scripts.ini` contains:

```
[Shutdown]
0CmdLine = C:\Program Files (x86)\wapt\waptexit.exe
0Parameters =
```

- The **wapt-get remove-upgrade-shutdown** command removes the **waptexit** local security policy object, disabling the execution of **waptexit** during system shutdown.

The command **wapt-get add-upgrade-shutdown** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
0
```

The `scripts.ini` contains:

```
[Shutdown]
```

31.2.11 wapt-get register

The **wapt-get register <description>** command reports the computer hardware and software inventory to the WAPT Server.

The <description> parameter is optional.

Hint: A description may be passed as an argument to the **wapt-get register** command, the description will be displayed in the WAPT Console in the column *description*.

You may benefit from WAPT to improve your IT management by affecting an inventory tag as a description for your hosts for example.

Note: If the host is already registered, re-registering the host using a description updates the registered information.

The command **wapt-get register "John Doe PC"** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Registering host against server: https://srvwapt.mydomain.lan
Host correctly registered against server https://srvwapt.mydomain.lan.
```

31.2.12 wapt-get unregister

The `wapt-get unregister` command removes the hardware and software inventory of the host from the WAPT Server.

The command `wapt-get unregister` returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Unregistering host from server: https://srvwapt.mydomain.lan
Please get login for api/v3/hosts_delete:admin
Password:
Host correctly unregistered against server https://srvwapt.mydomain.lan.
```

31.2.13 wapt-get inventory

The **wapt-get inventory** command displays the inventory information of the host in *json* format.

The command **wapt-get inventory** returns (in part):

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
{
  "host_info": {
    "description": "John Doe PC",
    "system_manufacturer": "Xen",
    "system_productname": "HVM domU",
    "computer_name": "Documentation",
    "computer_fqdn": "Documentation.srvwapt.mydomain.lan",
    "dnsdomain": "mydomain.lan",
    "workgroup_name": "Documentation",
    "domain_name": null,
    "domain_controller": null,
    "domain_controller_address": null,
    "domain_info_source": "history",
    "networking": [
    {
      "iface": "{085AB96368A-05A3B96-43EC-B773-0C0BB96794D9}",
      "mac": "a2:1d:6e:fc:8d:e6",
      "addr": [
      {
        "addr": "192.168.0.1",
        "netmask": "255.255.255.0",
```

(continues on next page)

(continued from previous page)


```
        "broadcast": "192.168.0.255",
        "connected": true
      },
      {
        "addr": "fe80::2437:567f:79c8:f964",
        "netmask": "ffff:ffff:ffff:ffff::/64",
        "broadcast": "fe80::ffff:ffff:ffff:ffff%3",
        "connected": true
      }
      ]
    }
    ],
    "gateways": [
    "192.168.0.254"
    ],
    "dns_servers": [
    "192.168.0.11"
    ],
    "connected_ips": [
    "192.168.0.1",
    "fe80::2437:567f:79c8:f964"
    ],
    "mac": [
    "a2:fc:1d:6e:8d:e6"
    ],
...
```


31.2.14 wapt-get update-status

The command **wapt-get update-status** sends the current status of the host to the WAPT Server.

Note: If a hardware component has changed on the computer, **wapt-get update-status** would not report that information back to the WAPT inventory Server.

To do so, the command to use is **wapt-get inventory**.

The command **wapt-get update-status** returns:


```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Updated host status correctly sent to the WAPT Server https://srvwapt.mydomain.lan. {'success':␣
↪True, 'msg': 'update_host', 'result': {'uuid': 'B8D346E7-DDDB-0013-5A8A-425CF3B6199E', 'computer_
↪fqdn': 'documentation.mydomain.lan', 'status_hashes': {'dmi':
↪'124b8bcef5b690afea7cf8001351a22132885123', 'wmi': 'ae5dbb5627b7b3a5a31d5914a9dbf48b85b133da',
↪'host_info': 'e737a82da15fbe9cae88ba9b4a9662a73657d959', 'audit_data': None, 'wapt_status':
↪'bcb76ad07cf1b6f814082ec5a58c4fee0364a640', 'audit_status':
↪'c34adb535c711b59d4408f00f77b7392687d7e56', 'host_metrics':
↪'9fc68bd98c82e0e9bece0ce3afaeeb63a3ed1db1', 'waptwua_status':
↪'4f9dcf0af339ce28d7354283fd4e6bdaf17b85c8', 'waptwua_updates':
```


(continues on next page)

(continued from previous page)


```
↪'c5cf38908fc549f499ade5b17ce221ff0ced377f', 'wuauserv_status':
↪'7c30215c3c34566e5b0c69c9e1dbfe3e6117b837', 'host_capabilities':
↪'c31286122a213f3bb313531541582bb2ba1d0a81', 'installed_packages':
↪'3279f3bf4d5ed5086b198fa94a6a6f422f519ab3', 'last_update_status':
↪'347c5a8c01e182f1e03e5c9d0fe07dd87ab79153', 'installed_softwares':
↪'d582a6f7325af35eae17cb7ecdca59ef0d137dda', 'authorized_certificates':
↪'2974f9535f813fc454b735193c31828b132a6ba0', 'waptwua_updates_localstatus':
↪'c5cf38908fc549f499ade5b17ce221ff0ced377f'}, 'server_uuid': '82295c4d-4944-11ec-bac6-a25b5d7da3d5
↪'}, 'request_time': 0.046843767166137695}
```


31.2.15 wapt-get setlocalpassword

The **wapt-get setlocalpassword** command allows to define a local password for installing WAPT packages.

The command **wapt-get setlocalpassword** returns:

```
Local password:
Confirm password:
Local auth password set successfully
```

31.2.16 wapt-get reset-uuid

The **wapt-get reset-uuid** command retrieves the host's *UUID* from BIOS and sends it to the WAPT Server.

The command **wapt-get reset-uuid** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
New UUID: B0F23D44-86CB-CEFE-A8D6-FB8E3343FE7F
```

31.2.17 wapt-get generate-uuid

The **wapt-get generate-uuid** command generates a random *UUID* for the host and sends it back to the WAPT Server.

The **wapt-get generate-uuid** is useful if there are *bios UUID bugs* with some hosts in the fleet.

The command **wapt-get generate-uuid** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
New UUID: RND-0279A1F4-BBBE-43AE-A591-F82652E0104B
```

Note: All randomly generated UUID start with *RND-*.

31.2.18 wapt-get get-server-certificate

The **wapt-get get-server-certificate** command downloads the SSL certificate from the WAPT Server so that the WAPT Agent may establish a secured HTTPS connection with the WAPT Server.

The downloaded certificate is stored in <wapt>\ssl\server.

The command **wapt-get get-server-certificate** returns:

```
Server certificate written to C:\Program Files (x86)\wapt\ssl\server\srvwapt.mydomain.lan.crt
```

31.2.19 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 WAPT Server.

The **wapt-get enable-check-certificate** command is used for *activating the verification of the SSL / TLS certificate*.

The command **wapt-get enable-check-certificate** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Server certificate : C:\Program Files (x86)\wapt\ssl\server\template-auto.test.lan.crt
Certificate CN: template-auto.test.lan
Pining certificate C:\Program Files (x86)\wapt\ssl\server\template-auto.test.lan.crt
```

31.2.20 wapt-get check-upgrades

The **wapt-get check-upgrades** command shows the most recent update / upgrade status for the host.

The command **wapt-get check-upgrades** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
{
  "running_tasks": [],
  "errors": [],
  "date": "2022-01-06T14:09:26.554391",
  "upgrades": [
    "tis-notepadplusplus(=8.2-10)"
  ],
  "pending": {
    "install": [],
    "upgrade": [
    "tis-notepadplusplus(=8.2-10)"
    ],
    "additional": [],
    "remove": [],
    "immediate_installs": []
  }
}
```

31.2.21 wapt-get add-licence

The **wapt-get add-licence** command add a WAPT licence onto the WAPT Server.

The command **wapt-get add-licence** returns:

```
Using config file C:\Program Files (x86)\wapt\wapt-get.ini
Server: https://srvwapt.mydomain.lan
Server UUID: 82295c4d-4944-11ec-bac6-a25b5d7da3d5
Server CABundle: 0

{"licence_nr":"6f011e23-cb70-40a4-b340-0d18ae1e2f02","product":"WAPT Enterprise","features":["full
↪"],"licenced_to":"documentation","domain":"","contact_email":"documentation@tranquil.it","count":
↪"10","valid_from":"2021-06-14T00:00:0","valid_until":"2022-01-12T00:00:0","renewal_url":null,
↪"signed_attributes":["licence_nr","product","features","licenced_to","domain","contact_email",
↪"count","valid_from","valid_until","renewal_url","signed_attributes","signer","signature_date",
↪"signer_certificate","server_uuid"],"signer":"","signature_date":"2022-01-13T16:38:56","signer_
↪certificate":"-----BEGIN CERTIFICATE-----\nMIIEIjCCAwqgAwIBAgIUIOMdx8FmRdmCNTHxOfKecSp/
↪cAAwDQYJKoZIhvcNAQEL\nBQAwgZcxCzAJBgNVBAYTAkZSMSIwIAYDVQQHDBlTYWludCBTZWJhc3RpZW4gc3Vy\
↪nIExvaXJlMRwwGgYDVQQKDBNUcmFucXVpbCBJVCBTeXN0ZW1zMSAwHgYDVQQDDBdy\
↪nZWxpY2VuY2luZy50cmFucXVpbC5pdDEkMCIGCSqGSIb3DQEJARYVdGVjaG5pcXVl\
↪nQHRyYW5xdWlsLml0MB4XDTIxMDYwODE0MTQ0MVoXDTMxMDYwNjE0MTQ0MVowgZcx\
↪nCzAJBgNVBAYpk6dZrIrw9Kb5hee+1EgqEbudCBTZWJhc3RpZW4gc3VyIExvaXJl\
↪nMRwwGgYDVQQKDBNUcmFucXVpbCBJVCBTeXN0ZW1zMSAwHgYDVQQDDBdyZWxpY2Vu\
↪nY2luZy50cmFucXVpbC5pdDEkMCIGCSqGSIb3DQEJARYVdGVjaG5pcXVlQHRyYW5x\
↪ndWlsLml0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzT43W8OhWXAe\nhDB+IWwQm9IGGdR0VY/klKcSheo/
↪8jGlNziyH6BANhjFKYNX9UtQ+ghzv6BGfSTH\nyua1aXEQM89sSKFOoJztoD1L9FZtuWQb/vfLWkisP8fRPvH4B/
↪tYG+5nchGa6+6r\nqGSGSpWcnj6CovgQR01ATUuHN3NV1N7q48hBT/ZT9R5U3sEi+hNK4eRIeZ220Pzm\
↪nDoNGkVKlEiczgXuM77ezYp8UWvpk6dZrIrw9Kb5hee+1EgqEbgVmdARoaOPGTK8h\
↪n8VW+milWsl4TEY19kxXWvva+M6wX00ipJ2LxEiu5+dlOok9E8i405UTNE7oSVYsF\n90/
↪6S3C4twIDAQABo2QwYjAPBgNVHRMBAf8EBTADAQH/MCAGA1UdJQEB/wQWMBQG\
↪nCCsGAQUFBwMDBggrBgEFBQcDAjAdBgNVHQ4EFgQUpRT6Co2uoWZMCwP7FKiF73+j\nfAEwDgYDVR0PAQH/
↪BAQDAgHWMA0GCSqGSIb3DQEBCwUAA4IBAQAdXX5IkpuH/Gek\nPPHC4KvE/
↪6GsU0kqLI1w5ML5pbF1zyCCL0nm4f8w2JJIJ2Ycdb4QVD27kJqgZcH1\nniYQ3RCIh6aasS8qpCOf90KkpvKMJiyk/
↪ra7I6NSgPut4ErkoxUWocgF6SNFEjwB\naqUZY//Hkoqk2dXqdujLVGJfBpX95ZJ9PmFNLfsyUsvu1WcFMb0En0EUO74Mq4M3\
↪nKo2S86G9pEDKooaN5Vq19biReOwQYpX1YlSLtrxFx8AM87auQgaD8EWSdA1q2ycN\
↪n8ZnmXGxAhDv8hmE2Fv0x0t3hzYXxxcv1ZjYWRHlMUl/buWQQ35u9MFkjh7YZlTlM\nb9wjtN+W\n-----END CERTIFICATE-
↪----\n","signature":
↪"J7DZ+mja7zGghYFCDKh8WIxzzdhKPeoNswWjnKZziT+ddpoRdg45kZz4E8PxMIUzhTI8WIxzzdhKPeoNswrICpQ8t5kepzovZpoONwjgOQ
↪de18bEgSSlgjXgE/wr2ZfclsRsRRfsRbGSterRKQcthNDrFlf8RjH5cpDnDvMJ+qJtTsqxA13/WT2NS2uNWZI93si/
↪9mowWY8MdT/PZjosciCqijbq4oa+/FrPsALhUOtcGE9JylwknszUD5Ayfh+9sNLLxsG6eT0JlnNgf4nx9mXAu4GBg==",
↪"server_uuid":"82295c4d-4944-11ec-bac6-a25b5d7da3d5"}
Login to server api/v3/licences
Waptserver https://srvwapt.mydomain.lan Admin User () :admin
Waptserver Password: ************
Licence properly activated on server
```

31.2.22 wapt-get check-licences

The **wapt-get check-licences** command shows the licences registered on the WAPT Server.

The command **wapt-get check-licences** returns:

```
Using config file C:\Program Files (x86)\wapt\wapt-get.ini
Server: https://srvwapt.mydomain.lan
Server UUID: 36bf01bc-c8f5-11eb-bf04-36127be97253
Server CABundle: 0

Total licences count: 10
Licenced to: documentation

Valid Nr:b7b6e537-3cb7-4d9a-3cb7-2448020e2e51 Count:10 From:2022-01-13T00:00:0 Expire:2023-01-
↪12T23:59:0 Server:36bf01bc-c8f5-11eb-bf04-36127be97253 Licencee:documentation
```

31.2.23 wapt-get dnsdebug

The **wapt-get dnsdebug** command shows network configuration data for the host, notably the local DNS data related to WAPT.

The command **wapt-get dnsdebug** returns:

```
DNS Server : dns.mydomain.lan
DNS Domain : mydomain.lan
Main repo url: https://srvwapt.mydomain.lan/wapt
wapt SRV: []
waptserver SRV: []
CNAME: []
```

31.3 Using the Command Line for user session setup

31.3.1 wapt-get session-setup

The **wapt-get session-setup** command launches user level customizations of installed WAPT packages.

The **wapt-get session-setup** command runs the *def session_setup()* function defined in the `setup.py` file of the WAPT package if the function exists.

Note: The argument *ALL* will launch **wapt-get 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
```

(continues on next page)

(continued from previous page)

```
Configuring mdl-firefox-esr ... No session-setup. Done
Configuring tis-paint.net ... No session-setup. Done
```

31.4 Using the Command Line to create WAPT packages

31.4.1 wapt-get list-registry

The **wapt-get list-registry <keyword>** command lookups a keyword in software installed on the computer.

The command can take one case insensitive argument to search for the specified keyword.

The informations returned is:

Information	Definition	Available on Windows	Available on Linux	Available on macOS
`UninstallKey`	Searches the uninstall key identifier in the registry hive.	✅	❌	❌
`Software`	Searches the name of the software in the registry hive.	✅	✅	✅
`Version`	Searches the version of the software in the registry hive.	✅	✅	✅
`Uninstallstring`	Searches the uninstall string of the software in the registry hive.	✅	❌	❌

Note:

- On Windows, WAPT searches in two registry locations:
  - `ComputerHKEY_LOCAL_MACHINESOFTWAREMicrosoftWindowsCurrentVersionUninstall`.
  - `ComputerHKEY_LOCAL_MACHINESOFTWAREWOW6432NodeMicrosoftWindowsCurrentVersionUninstall`.
- On Linux, WAPT searches using `Applications`.
- On macOS, WAPT searches in `/var/lib/dpkg/info/`.

The output of **wapt-get list-registry** is a table listing *uninstall keys* for each software corresponding to the search term.

The command **wapt-get list-registry firefox** returns (on Windows):

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
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"
```

31.4.2 wapt-get sources

The **wapt-get sources <package name>** command downloads sources from a source code management repository like Git or SVN.

The command **wapt-get sources tis-firefox** returns nothing;

31.4.3 wapt-get make-template

> **Warning:** This method is deprecated, instead use the *WAPT Console to create a package template*.

The **wapt-get make-template <installer-path> [<packagename> [<source directoryname>]]** command allows to create a package template from a *msi* or an *exe* installer.

The command **wapt-get make-template C:\Users\User\Downloads\tightvnc.msi tis-tightvnc** returns:

```
Using config file: C:\Users\Documentation\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-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
```

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.

31.4.4 wapt-get make-host-template

> **Warning:** This method is mainly for scripting, usually the host packages are automatically created with the WAPT Console.

The **wapt-get make-host-template <hostname> [[<package>,<package>,...] [directory]]** command creates an empty WAPT host package from a template.

The command **wapt-get make-host-template host01.mydomain.lan** returns:

```
Using config file: C:\Users\Documentation\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\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
```

31.4.5 wapt-get make-group-template

Warning: This method is to be used only if you can not use the WAPT Console to create a package.

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
```

31.4.6 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 build-package c:waptdevtis-dropbox** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
Building packages 1 packages
Personal certificate is documentation
Please type the password to decrypt the private key C:\Users\documentation\private\documentation.pem
Password:
Private key is <SSLPrivateKey 'C:\\Users\\documentation\\private\\documentation.pem'>
Personal certificate is SSLCertificate cn=documentation
Private key is <SSLPrivateKey 'C:\\Users\\documentation\\private\\documentation.pem'>
Building c:\waptdev\tis-dropbox
Signing c:\waptdev\tis-dropbox with key <SSLPrivateKey 'C:\\Users\\documentation\\private\\
↪documentation.pem'> and certificate documentation (C:\Users\documentation\private\documentation.
↪crt)
Package c:\waptdev\tis-dropbox signed : signature : BN7j6lwloY...Iu9QVulA=
...done building. Package filename c:\waptdev\tis-dropbox_104.4.175-7_windows_
↪0f4137ed1502b5045d6083aa258b5c42_5.1_PROD.wapt
1 packages successfully built
0 packages failed

You can upload to repository with
  C:\Program Files (x86)\wapt\wapt-get.exe upload-package "c:\waptdev\tis-dropbox_104.4.175-7_
↪windows_0f4137ed1502b5045d6083aa258b5c42_5.1_PROD.wapt"
```

Warning: The directory name does not define the name of the package, nor does it define its prefix, these values are defined by the `control` file.

31.4.7 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: **wapt-get 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:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
Signing packages c:\waptdev\tis-dropbox
Personal certificate is SSLCertificate cn=documentation
Please type the password to decrypt the private key C:\Users\documentation\private\documentation.pem
Password:
Private key is <SSLPrivateKey 'C:\\Users\\documentation\\private\\documentation.pem'>
Signing c:\waptdev\tis-dropbox
  OK: Package c:\waptdev\tis-dropbox signed : signature : b'nJYfYswDWi'...b'v790D7uA='
```

31.4.8 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 **wapt-get build-upload**, the WAPT packaging version number is incremented before the package is uploaded, so to avoid having to modify manually the `control` file.

The command **wapt-get -i build-upload C:\waptdev\tis-tightvnc-wapt** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
Building packages 1 packages
Personal certificate is documentation
Please type the password to decrypt the private key C:\Users\documentation\private\documentation.pem
Password:
Private key is <SSLPrivateKey 'C:\\Users\\documentation\\private\\documentation.pem'>
Personal certificate is SSLCertificate cn=documentation
Private key is <SSLPrivateKey 'C:\\Users\\documentation\\private\\documentation.pem'>
Building c:\waptdev\tis-dropbox
Signing c:\waptdev\tis-dropbox with key <SSLPrivateKey 'C:\\Users\\documentation\\private\\
↪documentation.pem'> and certificate documentation (C:\Users\documentation\private\documentation.
↪crt)
Package c:\waptdev\tis-dropbox signed : signature : s9FOLFQvYw...c9T3Hv1A=
...done building. Package filename c:\waptdev\tis-dropbox_104.4.175-7_windows_
↪0f4137ed1502b5045d6083aa258b5c42_5.1_PROD.wapt
1 packages successfully built
0 packages failed
Building and uploading packages to https://srvwapt.mydomain.lan
Please get login for https://srvwapt.mydomain.lan/api/v3/upload_xxx:admin
Password:
c:\waptdev\tis-dropbox_104.4.175-7_windows_0f4137ed1502b5045d6083aa258b5c42_5.1_PROD.
```

(continues on next page)

(continued from previous page)

```
↪wapt[==============================] 126459984/126459984 - 00:00:40
Package uploaded successfully: 1 Packages uploaded, 0 errors
```

31.4.9 wapt-get duplicate

The **wapt-get duplicate <source_package> <duplicated_package>** command duplicates a package downloaded from the repository and opens it as a project using the IDE (Integrated Development Environment) that has been specified in the *configuration of the WAPT Console*.

> **Warning:** Do not use this command to duplicate a *host* package.

Table 1: Allowed arguments when duplicating a WAPT package.

Argument	Definition	Required
<directory> or <source_package>	Defines the directory path to the WAPT package or the name of a specific package having a *.wapt* file extension.	✓
<duplicated_package>	Defines the name of the new package.	✓
<duplicated_package_version>	Changes the package version in the `control` file. If the version is not defined, the same version is duplicated.	✗
<duplicated_package_target_directory>	Defines the path to the target directory of the duplicated package. If the target directory is not defined, the duplicated package will be stored in the same directory as the source package.	✗

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
```

31.4.10 wapt-get edit

> **Warning:** This method is to be used only if you can not use the WAPT Console to create a package.

The **wapt-get edit <package name>** command downloads and opens the package in an IDE for modification.

The command takes one argument, the name of the WAPT package or a list of WAPT packages with the repository prefix.

The command **wapt-get edit tis-firefox** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 1629411 /␣
↪54313787 (3%) (2686 KB/s)
```

(continues on next page)

(continued from previous page)

```
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 8147055 /␣
↪54313787 (15%) (5679 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 15207836 /␣
↪54313787 (28%) (7367 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 19552932 /␣
↪54313787 (36%) (7249 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 24984302 /␣
↪54313787 (46%) (7471 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 29329398 /␣
↪54313787 (54%) (7143 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 33674494 /␣
↪54313787 (62%) (6951 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 40735275 /␣
↪54313787 (75%) (7534 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 45623508 /␣
↪54313787 (84%) (7326 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 53227426 /␣
↪54313787 (98%) (7603 KB/s)
https://srvwapt.mydomain.lan/wapt/tis-firefox_94.0.1-106_x64_windows_
↪0f4137ed1502b5045d6083aa258b5c42_6.1_PROD_en_f5335369ad5805e8dbc1f8ae99b2258e.wapt : 54313787 /␣
↪54313787 (100%) (7663 KB/s)
Package edited. You can build and upload the new WAPT package by launching
```

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.
- You can edit a local package by going in the package folder then typing **wapt-get edit .**.
- Another method, you can edit a local package from its directory name or wapt package name, example **wapt-get edit tis-vlc.wapt**.

31.4.11 wapt-get edit-host

Warning: This method is to be used only if you can not use the WAPT Console to create a package.

The **wapt-get edit-host <host FQDN>** command edits a WAPT *host* package.

The command **wapt-get edit-host RND-0279A1F4-BBBE-43AE-A591-F82652E0104B** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
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\RND-0279A1F4-BBBE-43AE-A591-
↪F82652E0104B_0-wapt
```

31.4.12 wapt-get update-package-sources

The **wapt-get update-package-sources <path to the package>** command update run the `def update_package()` function in `setup.py` file.

The command **wapt-get update-package-sources tis-firefox** returns:

```
Using config file: C:\Users\documentation\AppData\Local\waptconsole\waptconsole.ini
Latis Mozilla Firefox version is: 95.0.2
Download URL is: https://download-installer.cdn.mozilla.net/pub/firefox/releases/95.0.2/win64/en-US/
↪Firefox%20Setup%2095.0.2.exe
Downloading: Firefox_Setup_95.0.2.exe
Firefox_Setup_95.0.2.exe[==============================] 54810424/54810424 - 00:00:07
Software version updated (from: 94.0.1 to: 95.0.2)
Packages updated :
c:\waptdev\tis-firefox_0-wapt
```

31.5 Using the command-lines for WaptWUA management

31.5.1 wapt-get waptwua-scan

The **wapt-get waptwua-scan** scans the status of Windows Updates against current rules and sends the result back to the WAPT Server.

The command **wapt-get waptwua-scan** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Ensure wuauserv Auto Update option is disabled
Scanning with windows updates rules:
{
"direct_download": false,
"default_allow": false,
"filter": "Type='Software' or Type='Driver'",
"download_scheduling": "7d",
```

(continues on next page)

(continued from previous page)

```
"install_scheduling": null,
"install_delay": null,
"postboot_delay": "10m"
}
Downloading wsusscn2.cab file from https://srvwapt.mydomain.lan/waptwua/wsusscn2.cab
wsusscn2.cab[==============================] 1024297844/1024297844 - 00:02:04
Windows updates rules have been changed
Looking for updates with filter: Type='Software' or Type='Driver'
  Connecting to local update searcher using offline wsusscn2 file...
  Offline Update searcher ready...
Waiting for WUA search to complete
Done searching
WUA Search completed !
Updates scan done.
Writing status in local wapt DB
Status: OK
(0, 0, 0)
None
re-enabling wuauserv previous state: 0
```

31.5.2 wapt-get waptwua-download

The **wapt-get waptwua-download** command scans the status of the Windows Update Agent against current rules, then downloads the missing kb and finally sends the result to the WAPT Server.

The command **wapt-get waptwua-download** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Ensure wuauserv Auto Update option is disabled
Downloading wsusscn2.cab file from https://srvwapt.mydomain.lan/waptwua/wsusscn2.cab
wsusscn2.cab[==============================] 1024297844/1024297844 - 00:00:26
Start of install for all pending Windows updates
Scanning with params:
{
"direct_download": false,
"default_allow": false,
"filter": "Type='Software' or Type='Driver'",
"download_scheduling": "7d",
"install_scheduling": null,
"install_delay": null,
"postboot_delay": "10m"
}
Scanning with windows updates rules:
{
"direct_download": false,
"default_allow": false,
"filter": "Type='Software' or Type='Driver'",
"download_scheduling": null,
"install_scheduling": null,
```

(continues on next page)

(continued from previous page)

```
"install_delay": null,
"postboot_delay": "10m"
}
Bypassing scan, no change since last successful scan
Writing status in local wapt DB
Status: OK
{'downloaded': [], 'missing': []}
None
re-enabling wuauserv previous state: 0
```

31.5.3 wapt-get waptwua-install

The **wapt-get waptwua-install** command installs pending Windows Updates on the host.

The command **wapt-get waptwua-install** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Ensure wuauserv Auto Update option is disabled
Downloading wsusscn2.cab file from https://srvwapt.mydomain.lan/waptwua/wsusscn2.cab
wsusscn2.cab[==============================] 1024297844/1024297844 - 00:00:26
Start of install for all pending Windows updates
Scanning with params:
{
"direct_download": false,
"default_allow": false,
"filter": "Type='Software' or Type='Driver'",
"download_scheduling": "7d",
"install_scheduling": null,
"install_delay": null,
"postboot_delay": "10m"
}
Scanning with windows updates rules:
{
"allowed_products": null,
"allowed_classifications": null,
"allowed_severities": null,
"allowed_updates": null,
"forbidden_updates": null,
"allowed_kbs": null,
"forbidden_kbs": null,
"default_allow": false
}
Looking for updates with filter: Type='Software' or Type='Driver'
  Connecting to local update searcher using offline wsusscn2 file...
  Offline Update searcher ready...
Waiting for WUA search to complete
Done searching
WUA Search completed !
Updates scan done.
```

(continues on next page)

(continued from previous page)

```
Installed 07609d43-d518-4e77-856e-d1b316d1b8a8 : MSXML 6.0 RTM Security Update (925673)
Installed bb49cc19-8847-4986-aa93-5e905421e55a : Security Update for Microsoft Visual C++ 2005␣
↪Service Pack 1 Redistributable Package (KB2538242)
Installed 729a0dcb-df9e-4d02-b603-ed1aee074428 : Security Update for Microsoft Visual C++ 2008␣
↪Service Pack 1 Redistributable Package (KB2538243)
Installed 719584bc-2208-4bc9-a650-d3d6347eb32e : Security Update for Microsoft Visual C++ 2010␣
↪Service Pack 1 Redistributable Package (KB2565063)
Installed a8761130-35b6-41ce-8b67-2d35bb2d0846 : 2021-02 Cumulative Update for .NET Framework 3.5␣
↪and 4.8 for Windows 10, version 20H2 for x64 (KB4601050)
Installed 30f75e5d-2c46-42be-aef6-97ae730452be : 2021-07 Cumulative Update for Windows 10 Version␣
↪20H2 for x64-based Systems (KB5004945)
Installed 6e88be6e-d470-4e7e-9f36-01479049aadb : 2021-08 Servicing Stack Update for Windows 10␣
↪Version 20H2 for x64-based Systems (KB5005260)
Installed a15155a4-1299-41ff-9a39-28a33ce7cadd : 2021-12 .NET Core 3.1.22 Security Update for x64␣
↪Client (KB5009193)
Installed 38db0ad6-27f8-4bf9-ab2a-cffc4d7bc390 : Windows Malicious Software Removal Tool x64 - v5.
↪96 (KB890830)
Scanning with windows updates rules:
{
"direct_download": false,
"default_allow": false,
"filter": "Type='Software' or Type='Driver'",
"download_scheduling": "7d",
"install_scheduling": null,
"install_delay": null,
"postboot_delay": "10m"
}
Windows updates rules have been changed
Writing status in local wapt DB
Status: OK
[]
None
re-enabling wuauserv previous state: 2
```

31.5.4 wapt-get waptwua-status

The **wapt-get waptwua-status** command returns the most recent Windows Update status for the host.

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
{'enabled': None,
'last_error': 'OperationalError: cannot rollback - no transaction is active',
'last_install_batch': [],
'last_install_date': None,
'last_install_reboot_required': None,
'last_install_result': None,
'last_scan_date': '2022-01-07T10:20:50.213644',
'last_scan_duration': 1490.500022649765,
'missing_downloads': [],
'rules_packages': [],
```

(continues on next page)

(continued from previous page)

```
'status': 'SCANNING',
'wsusscn2cab_date': '2021-12-14T04:06:46'}
None
```

31.6 Using the command-line for interacting with users

31.6.1 wapt-get propose-upgrade

The **wapt-get propose-upgrade** command offers logged in users to launch pending upgrades.

The command **wapt-get propose-upgrade** returns:

```
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
{'result': 1, 'summary': 'waptexit launched for 1 sessions'}
```

31.7 Using the command-lines for initial setup

31.7.1 wapt-get create-keycert

The `wapt-get create-keycert [<options>]` command creates a RSA key pair and X509 certificate.

Table 2: List of options available for creating certificates.

Option	Description	Default Value
`--CommonName`	Displays the name of the certificate.	/
`--CommonName64`	Displays the name of the certificate, encoded in base64 (if accents, spaces etc..).	/
`--CodeSigning` WAPT	Defines whether the certificate / key pair will be allowed to sign software packages.	`0`
`--CA` WAPT	Defines whether the certificate / key pair can be used to sign other certificates (i.e. to be allowed to behave as a main or an intermediate Certificate Authority).	`0`
`--ClientAuth`	Defines a property (use) of the certificate.	1 for a non-CA certificate
`--PrivateKeyPassword`	Defines the password for unlocking the key if `--NoPrivateKeyPassword` is not used.	Randomly generated password
`--PrivateKeyPassword64` if `--PrivateKeyPassword` is not used	Defines the password for unlocking the key, encoded in base64 (if accents, spaces etc..).	Randomly generated password
`--NoPrivateKeyPassword`	Sets the private key as not being password protected if `--PrivateKeyPassword` or `--PrivateKeyPassword64` are not used.	Empty
`-F`	Forces the overwriting of an existing certificate.	/
`--Country`	Defines the name of the certificate holder's country to register in the certificate.	/
`--Locality`	Defines the name of the certificate holder's city to register in the certificate.	/
`--Organization`	Defines the name of the certificate holder's Organization to register in the certificate.	/
`--OrgUnit`	Defines the name of the certificate holder's Organization Unit (Service) to register in the certificate.	/
`--Email`	Defines the email address of the certificate holder to register in the certificate.	/
`--CAKeyFilename`	Defines the path to the private key (`.pem`) of a Certificate Authority.	Parameter `default_ca_key_path` in `waptconsole.ini`
`--CACertFilename`	Defines the path to the public certificate (`.crt`) of a Certificate Authority.	Parameter `default_ca_cert_path` in `waptconsole.ini`
`--CAKeyPassword`	Defines the password for unlocking the key of a Certificate Authority.	/
`-NoCAKeyPassword`	Sets the Certificate Authority key as not being password protected.	/
`--BaseDir`	Defines the folder where the private key and the public certificate will be stored.	Directory `personal_certificate_path` in `waptconsole.ini`
`-EnrollNewCert`	Copies the certificate to `waptssl`	/
`-SetAsDefaultPersonalCert`	Defines the `personal_certificate_path` in `waptconsole.ini`.	/

The command `wapt-get create-keycert` returns:

```
Using config file C:\Users\Administrator\AppData\Local\waptconsole\waptconsole.ini
```

(continues on next page)

(continued from previous page)

```
BaseDir: C:\private\
Common name of certificate to create: documentation
Private Key Filename: C:\private\documentation.pem
Certificate Filename: C:\private\documentation.crt
New private key password: QR.-DVp6MPGW
```

Warning: If `default_ca_key_path` and `default_ca_cert_path` are defined in `waptconsole.ini`, then you must place the CA certificate in the same location otherwise this error appears:

```
wapt-get create-keycert
Using config file C:\Users\tisadmin\AppData\Local\waptconsole\waptconsole.ini
BaseDir: C:\Users\tisadmin\private\
Common name of certificate to create: CRT
Exception at 00483595: Exception:
CA Certificate C:\Program Files (x86)\wapt\ssl does not exist.
```

31.7.2 wapt-get build-waptagent

The **`wapt-get build-waptagent [</ConfigFilename>]`** compiles and uploads a **`waptagent.exe`** and a **`waptupgrade.exe`** packages using the `/ConfigFilename` parameter to specify the content of the `wapt-get.ini` *configuration file* of the WAPT Agents.

Note: By default, the command uses configuration elements from the `waptconsole.ini` configuration file of the WAPT Console.

The command **`wapt-get build-waptagent`** returns:

```
Building customized waptagent.exe installer
................................................................................
Built C:\Users\documentation\AppData\Local\Temp\wapt20220107T12203700000.tmp\waptupgrade\waptagent.
↪exe
Private key Password for C:\Users\documentation\private\documentation.crt : ********
Building waptupgrade package
Waptserver https://srvwapt.mydomain.lan Admin User () :admin
Waptserver Password: ************
Uploading customized waptagent.exe installer
Uploading C:\Users\documentation\AppData\Local\Temp\wapt20220107T12203700000.tmp\waptupgrade\
↪waptagent.exe to waptserver https://srvwapt.mydomain.lan
OK
Uploading C:\Users\documentation\AppData\Local\Temp\wapt20220107T12203700000.tmp\tis-waptupgrade_2.
↪1.2.10605-0_all_PROD_all.wapt to waptserver https://srvwapt.mydomain.lan
OK : 1 Packages uploaded, 0 errors. Errors:
```

31.8 Using the command-lines with options

Option	Definition
--version	Shows the WAPT version number and exits.
-h | --help	Shows this help message and exits.
-c CONFIG | --config=CONFIG	Defines the path to another file like wapt-get.i
-l LOGLEVEL | --loglevel=LOGLEVEL	Defines the level of log files following this list: d
-D | --direct	Instructs not to use http service for **update** / **upg**
-S | --service	Requests a Waptservice user.
-u | --update-packages	Runs **wapt-get update** before the specified co
-f | --force	Forces the command.
-p PARAMS | --params=PARAMS	Sets up parameters as a JSon Object.
-r WAPT_URL | -repo WAPT_URL | --repository=WAPT_URL	Overrules the URL of the main WAPT repository
-y | --hide	Hides the WAPT Console during the execution o
-F FILTER_ON_HOST_CAP | --use-host-caps=FILTER_ON_HOST_CAP	Filters the packages based on the current host ca
-i | --inc-release	Increments the release version number when bui
-a UPDATE_SERVER_STATUS | --update-server-status=UPDATE_SERVER_STATUS	Sends the updated status of the host (soft, packag
keep-signature-date	Keeps the current package signature date, and fil
-s SECTION_FILTER | --sections=SECTION_FILTER	Adds a filter **section** to **wapt-get search**. Sec
-o REDIRECT_OUTPUT | --output=REDIRECT_OUTPUT	Redirects the output to a *.ini* file whose path is
-j | -json	Switches to *json* formatted output for the purpo
-e ENCODING | --encoding=ENCODING	Changes character encoding for the output.
-x EXCLUDES | --excludes=EXCLUDES	Defines a comma separated list of files or directo
-k PERSONAL_CERTIFICATE_PATH | --certificate=PERSONAL_CERTIFICATE_PATH	Defines the path to the PEM X509 certificate to s
-w PRIVATE_KEY_PASSWD | --private-key-passwd=PRIVATE_KEY_PASSWD	Defines the path to the file containing the private
-U USER | --user=USER	Defines an interactive user.
-g USERGROUPS | --usergroups=USERGROUPS	Defines groups of the end-user as a *json* format
-t MAX_TTL | --maxttl=MAX_TTL	Defines the maximum run time in minutes of the
-L LANGUAGE | --language=LANGUAGE	Overrides the locale for installing WAPT packag
-m MD | --message-digest=MD	Defines the message digest type for **wapt-get s**
-n | --newest-only	Returns only the newest version of WAPT packag
--locales=LOCALES	Override packages locales filter when using **wapt**
--maturity=MATURITY	Sets / changes the package maturity when buildi
--pin-server-cert	Pins the server certificate when registering the W
--wapt-server-url=SET_WAPTSERVER_URL	Defines the url of the WAPT Server when the pa
--wapt-repo-url=SET_WAPTREPO_URL	Defines the url of the WAPT repository when the
--wapt-server-user=WAPT_SERVER_USER	Defines the user allowed to upload packages to th
--wapt-server-passwd=WAPT_SERVER_PASSWD	Defines the user password allowed to upload pac
--log-to-windows-events	Logs steps to the Windows event log.
--use-gui	Forces the use of GUI Helper even if not in deve
--no-ide	Tells WAPT not to launch the IDE when editing

CHAPTER

THIRTYTWO

CONFIGURING THE WAPT AGENT WITH ADVANCED OPTIONS

The configuration file `wapt-get.ini` defines the behavior of the WAPT Agent.

Table 1: Location of `wapt-get.ini` by system

System	Location
Windows	`C:\Program Files(x86)\wapt\wapt-get.ini`
Linux	`/opt/wapt/wapt-get.ini`
Mac OS	`/opt/wapt/wapt-get.ini`

The `[global]` section is required.

```
[global]
```

After standard installation, the default configuration is:

```
[global]
waptupdate_task_period=120
wapt_server=https://srvwapt.mydomain.lan
repo_url=https://srvwapt.mydomain.lan/wapt/
use_hostpackages=1
```

All parameters are not set when the WAPT Agent is generated. It is possible to make changes in `wapt-get.ini` manually or by deploying a WAPT package with the new configuration settings.

An example package is available from the Tranquil IT repository.

```
# -*- 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)
```

(continues on next page)

(continued from previous page)

```
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)

print('Reload WAPT configuration')
WAPT.reload_config_if_updated()
```

The function `inifile_writestring` definition is:

```
inifile_writestring(inifilename,section,key,value)
```

32.1 Description of available sections

Table 2: Description of available sections for the WAPT Agent

Section	Description
[global]	Global WAPT Agent options.
[wapt]	Main repository options.
[wapt-templates]	External remote repository options.
[wapt-host]	Repository for host packages options.
[waptwua]	WUA Agent options.
[repo-sync]	For synching multiple repositories.

All sections are detailed below.

32.2 Description of available options by section

32.2.1 [global]

General settings

Table 3: Description of available options for the WAPT Agent in the [global] section

Options (Default Value)	Description	Example
allow_remote_reboot (default False)	Allows to reboot the selected host(s) remotely from the WAPT Console.	allow_remote_reboot = True
allow_remote_shutdown (default False)	Allows to shut down the selected host(s) remotely from the WAPT Console.	allow_remote_reboot = True
check_certificates_validity (default False)	Forces the package certificate's date and CRL to be verified.	check_certificates_validity = True
dbpath (default \wapt\db\waptdb.sqlite)	Path to the local database file.	dbpath = C:\Program Files (x86)\db\waptdb.sqlite
download_after_update_with_waptupdate_task_period (default True)	Defines whether package downloading should be started after an update with waptupdate_task_period.	download_after_update_with_waptupda = False
host_organizational_unit_dn (default None)	Allows to force an Organizational Unit on the WAPT Agent (convenient for assigning a *fake OU* for out-of-domain PC). Make sure it respects a consistent case (do not mix "dc"s and "DC"s, for example), which you can find in the Console (in the DN/computer_ad_dn fields for each host)	host_organizational_unit_dn = OU=TOTO,OU=TEST,DC=MYD
host_profiles (default None)	Allows to define a WAPT package list that the WAPT Agent **MUST** install.	host_profiles = tis-firefox,tis-java
language (default language on the WAPT Client)	Forces the default language for the GUI (not for package filtering)	language = en
locales (default locale on WAPT Client)	Allows to set the list of WAPT Agent languages to pre-filter the list of packages visible by the WAPT Agent (for package filtering). The parameter accepts multiple entries ordered by preference (eg. locales = fr,en).	locales = en
log_to_windows_events (default False)	Sends the WAPT logs in the Window event log.	log_to_windows_events = True
loglevel (default warning)	Log level of the WAPT Agent. Possible values are: debug, info, warning, critical.	loglevel = critical
maturities = (default PROD)	List of package maturities than can be viewed and installed by WAPT Agent. Default value is PROD. Only DEV, PREPROD and PROD values are used by Tranquil IT, however any value can be used to suit your internal processes.	maturities = PROD, PREPROD
repo_url (default your WAPT repo address)	Address of the main WAPT repository.	repo_url = https://srvwapt.mydomain.lan/wapt
repositories (default None)	List of enabled repositories, separated by a comma. Each value defines a section of the wapt-get.ini file. More info *here*.	repositories = repo1, repo2
send_usage_report (default True)	Allows the WAPT Console to send anonymous statistics to Tranquil IT. Set to False to disable telemetry.	send_usage_report = True
service_auth_type (default system)	Sets how the self service authentication works. Possible values are: system, waptserver-ldap or waptagent-ldap.	service_auth_type = waptserver-ldap
uninstall_allowed (default True)	Defines whether or not it is possible for the user to uninstall applications via the self-service.	uninstall_allowed = False
use_ad_groups (default False)	For using *group packages*.	use_ad_groups = True
use_fqdn_as_uuid (default False)	Allows to use the FQDN rather than the BIOS UUID as the unique host identifier in WAPT.	use_fqdn_as_uuid = True
use_hostpackages (default False)	Defines whether *host packages* are to be used. use_hostpackages = False disables implicit updates (host packages, unit packages, profile packages). It's useful if you want to isolate a host and use WAPT locally.	use_hostpackages = True
use_repo_rules (default False)	Defines whether *repositories are replicated*.	use_repo_rules =

Note:

- If there is no `repo_url` attribute in the [global] section, then 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.
- If there is no `wapt_server` attribute in the `[global]` section, then no WAPT Server will be used.

Settings for the WAPT Server

These options will set the WAPT Agent behavior when connecting to the WAPT Server.

Table 4: Description of available options for the WAPT Agent in the [global] section for the WAPT Server configuration

Options (Default Value)	Description	Example
`public_certs_dir` (default `None`)	Folder of certificates authorized to verify the signature of WAPT packages.	`public_certs_dir = C:\Program Files (x86)\wapt\ssl` (on Windows). `public_certs_dir = /opt/wapt/ssl/` (on Linux and MacOS)
`use_kerberos` (default `False`)	Use kerberos authentication for initial registration on the WAPT Server.	`use_kerberos = True`
`verify_cert` (default `False`)	See the documentation on activating the *verification of HTTPS certificates*.	`verify_cert = True`
`wapt_server` (default `None`)	WAPT Server URL. If the attribute is not present, no WAPT Server will be contacted.	`wapt_server = https://srvwapt.mydomain.lan`
`wapt_server_timeout` (default `30`)	WAPT Server HTTPS connection timeout in seconds.	`wapt_server_timeout = 10`

Settings for the WAPT Exit utility

Table 5: Description of available options for the WAPT Agent in the [global] section for the WAPT Exit utility

Options (Default Value)	Description	Example
`allow_cancel_upgrade` (default `True`)	Prevents users from canceling package upgrades on computer shutdown. If disabled, users will not be able to cancel an upgrade on computer shutdown. If this value is not indicated the default value will be **10**.	allow_cancel_upgrade = True
`hiberboot_enabled` (default `None`)	Disables Hiberboot on Windows 10 to make **`waptexit`** work correctly.	hiberboot_enabled = True
`max_gpo_script_wait` (default `None`)	Timeout for GPO execution at computer shutdown.	max_gpo_script_wait = 180
`pre_shutdown_timeout` (default `None`)	Timeout for scripts at computer shutdown.	pre_shutdown_timeout = 180
`upgrade_only_if_not_process_running` (default `False`)	Prevents the software upgrade if the software is currently running on the host (*impacted_process* attribute of the package).	upgrade_only_if_not_process_ru = True
`upgrade_priorities` (default `None`)	Only upgrade packages with a specific priority.	upgrade_priorities = high
`waptexit_countdown` (default 1)	Delay (in seconds) before the automatic start of the installations.	waptexit_countdown = 25

Settings for the WAPT Self-Service and the WAPT service Authentification

Table 6: Description of available options for the WAPT Agent in the [global] section for the WAPT Self-service and the WAPT service Authentification

Options (Default Value)	Description	Example
`ldap_auth_base_dn` (default `None`)	Useful with `service_auth_type = waptagent-ldap`, defines the *base dn* for the LDAP request.	ldap_auth_base_dn = dc=mydomain,dc=lan
`ldap_auth_ssl_enabled` (default `False`)	Useful with `service_auth_type = waptagent-ldap`, defines whether the LDAP request must be encrypted.	ldap_auth_ssl_enabled = True
`ldap_auth_server` (default `None`)	Useful with `service_auth_type =``waptagent-ldap``, defines the LDAP server to contact.	ldap_auth_server = srvads.mydomain.lan
`service_auth_type` (default `system`)	Defines the authentication system of the WAPT service, available value are `system`, `waptserver-ldap`, `waptagent-ldap`.	service_auth_type = waptagent-ldap
`verify_cert_ldap` (default `False`)	Useful with `service_auth_type = waptagent-ldap`, define whether the certificate should be verified.	verify_cert_ldap = True
`waptservice_admin_filter` (default `False`)	Apply *selfservice package* view filtering for Local Administrators.	waptservice_admin_filter = True
`waptservice_password` (default `None`)	sha256 hashed password when *waptservice_user* is used (the value *NOPASSWORD* disables the requirement for a password).	waptservice_password = 5e884898da
`waptservice_user` (default `None`)	Forces a user to authenticate on the WAPT service.	waptservice_user = admin

Settings for the the WAPT System Tray utility

Table 7: Description of available options for the WAPT Agent in the [global] section for the WAPT Tray utility

Options (Default Value)	Description	Example
`notify_user` (default `False`)	Prevents the WAPT System Tray utility from sending notifications (popup).	notify_user = True

Settings for the Proxy

Table 8: Description of available options for the WAPT Agent in the [global] section for the proxy

Options (Default Value)	Description	Example
`http_proxy` (default `None`)	Defines the address of the HTTP proxy.	`http_proxy = http://user:pwd@host_fqdn:port`
`use_http_proxy_for_repo` (default `False`)	Use a proxy to access the repositories.	`use_http_proxy_for_repo = True`
`use_http_proxy_for_server` (default `False`)	Use a proxy to access the WAPT Server.	`use_http_proxy_for_server = True`

Settings for creating WAPT packages

Table 9: Description of available options for the WAPT Agent in the [global] section for creating WAPT packages

Options (Default Value)	Description	Example
`default_package_prefix` (default `tis`)	Defines the default prefix for new or imported packages. Prefix is case sensitive, we recommand to use lower case.	`default_package_prefix = doc`
`default_sources_root` (default `C:\waptdev` on Windows or `~/waptdev` on Linux)	Defines the directory for storing packages while in development.	`default_sources_root = C:\\waptdev`
`personal_certificate_path` (default `None`)	Defines the path to the Administrator's private key.	`personal_certificate_path = None TODO`

32.2.2 [waptwua]

Refer to *configuring WAPTWUA on the WAPT Agent*.

32.2.3 [wapt]

If this section does not exist, parameters are read from the `[global]` section.

32.2.4 [wapt-templates]

External remote repositories that will be used in the WAPT Console for importing new or updated packages. The Tranquil IT repository is set by default.

32.2.5 [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*.

32.2.6 [repo-sync]

Configuration for secondary repo, this section must exist **only** if your WAPT Agent is a secondary repo.

More information on that usage can be found in *this article on configuring multiple repositories*.

32.3 Settings for using multiple repositories

To add more repositories, new `[repository_name]` sections can be added in the `wapt-get.ini`.

Active repositories are listed in the `repositories` attribute of the `[global]` section.

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*.

CHAPTER

THIRTYTHREE

GETTING START CREATING WAPT PACKAGES

33.1 Setting up your WAPT development and test environment

33.1.1 Prerequisites

Attention:

- It is **required** to be a *Local Administrator* of the host 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 host (like Virtualbox) is recommended.
- Import the *tis-waptdev* package in your local repository and install it on your development computer.

33.1.2 Recommendations regarding the test environment

The recommended method to correctly test your WAPT packages is to use representative sample of hosts in your inventory. So the more heterogeneous your installed base of hosts, the larger your sample should 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.

33.1.3 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 hosts;
- Laptops.

When possible, RC and Beta version of Operating Systems should be tested.

State of Windows Updates

- **Microsoft Windows host 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 host 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

- **Hosts with many installed packages**: the objective is to detect a possible dependency with an existing application;
- **Hosts 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;

33.2 Principles of creating package template from the WAPT Console

Attention: To create WAPT packages directly from the WAPT Console, it is necessary to have installed the WAPT development environment **tis-pyscripter** a minima.

We recommand you to download the **waptdev** package instead and install it on your computer where you'll create WAPT packages.

If you don't know anymore how to download a package from our store, please see *how to download package in your private repository*.

If you don't know anymore how to install a package , please see *how to install a package on a host*

33.2.1 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 7-zip MSI x64.
- Create a WAPT package Template from the installer.

  In the WAPT Console, click on *Tools* → *Make package template from setup file*:

  Select the downloaded MSI setup file and fill in the required fields. Verify that the package name does not contains any version number.
- Two solutions are available:
  - Click on *Make and edit ...* (recommended) to verify the WAPT package and customize it to your Organization's specific needs.

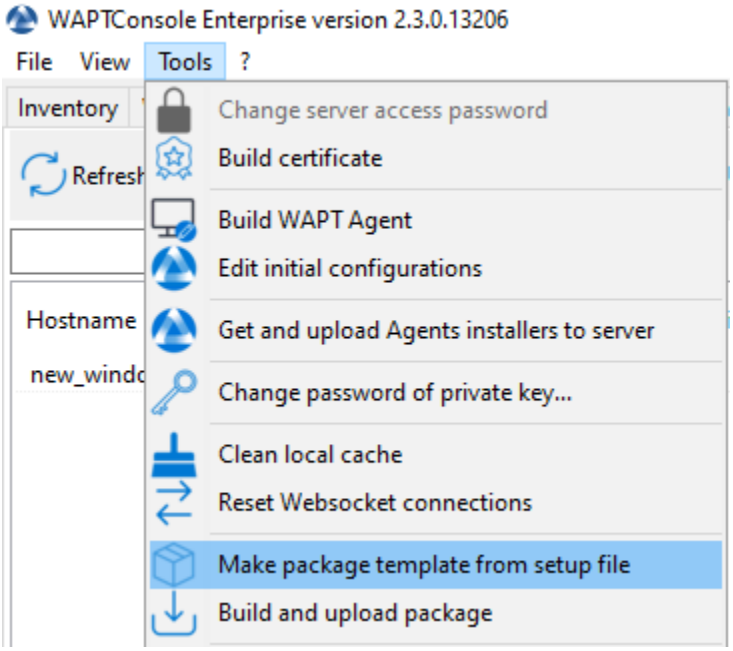


Fig. 1: Menu option for creating a WAPT package template in the WAPT Console

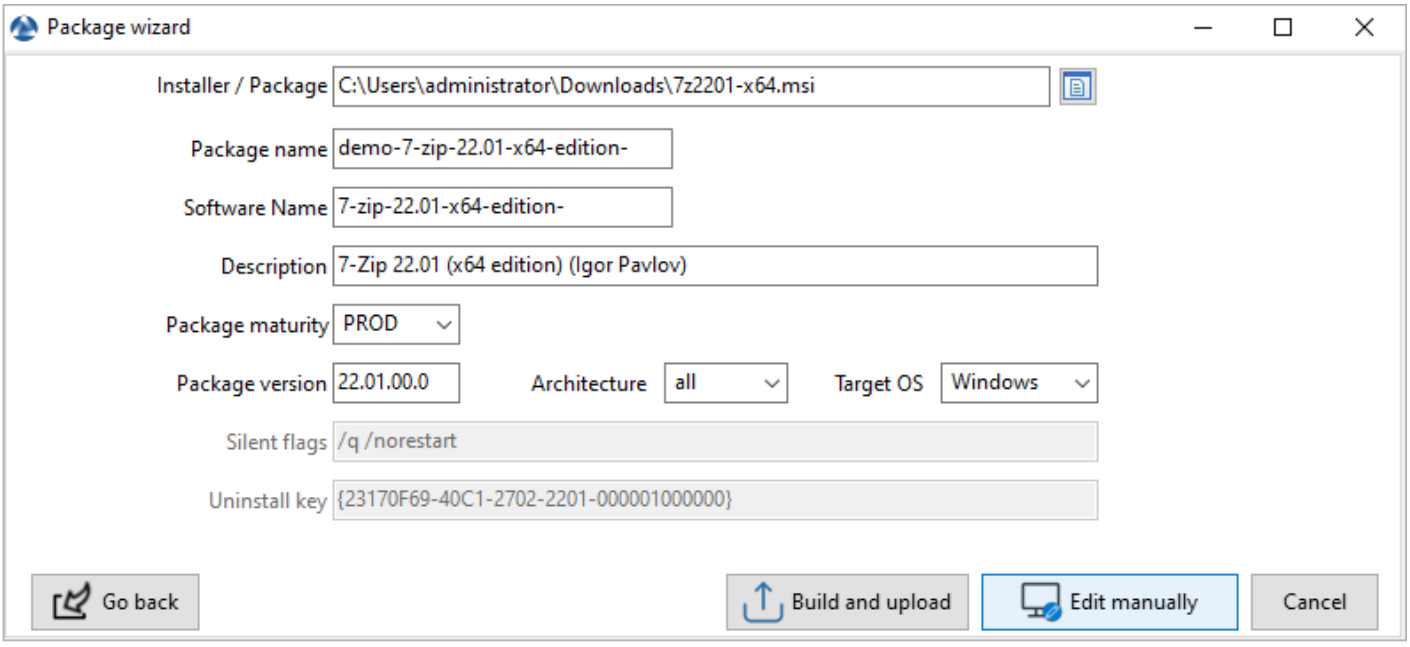


Fig. 2: Dialog box requesting information when creating the WAPT package in the WAPT Console

– Click on *Build and upload* to directly build and upload the package into your private repository.

> **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 first method that consists of first testing locally the package before uploading is the recommended method.

Note: *An old command line method is also available.*

33.2.2 Customizing the WAPT package before uploading it to the repository

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 ...*.

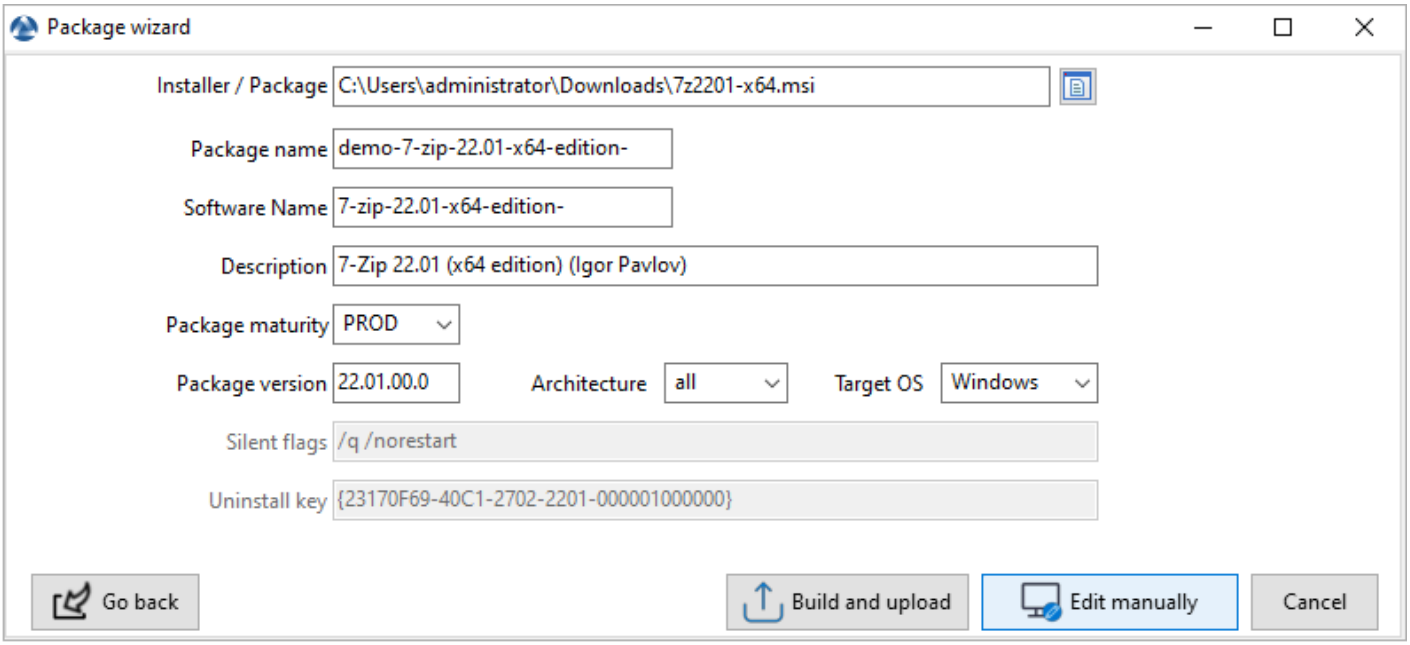


Fig. 3: Dialog box highlighting the "Make and edit ..." button when creating the WAPT package in the WAPT Console

The **PyScripter** IDE launches automatically to allow you to edit files in the WAPT package.

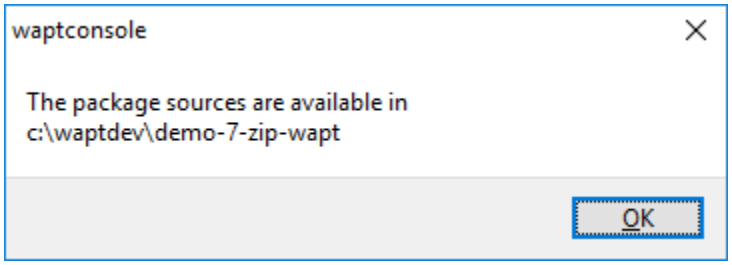


Fig. 4: Message window showing in the WAPT Console that the WAPT package has been downloaded into the WAPT repository

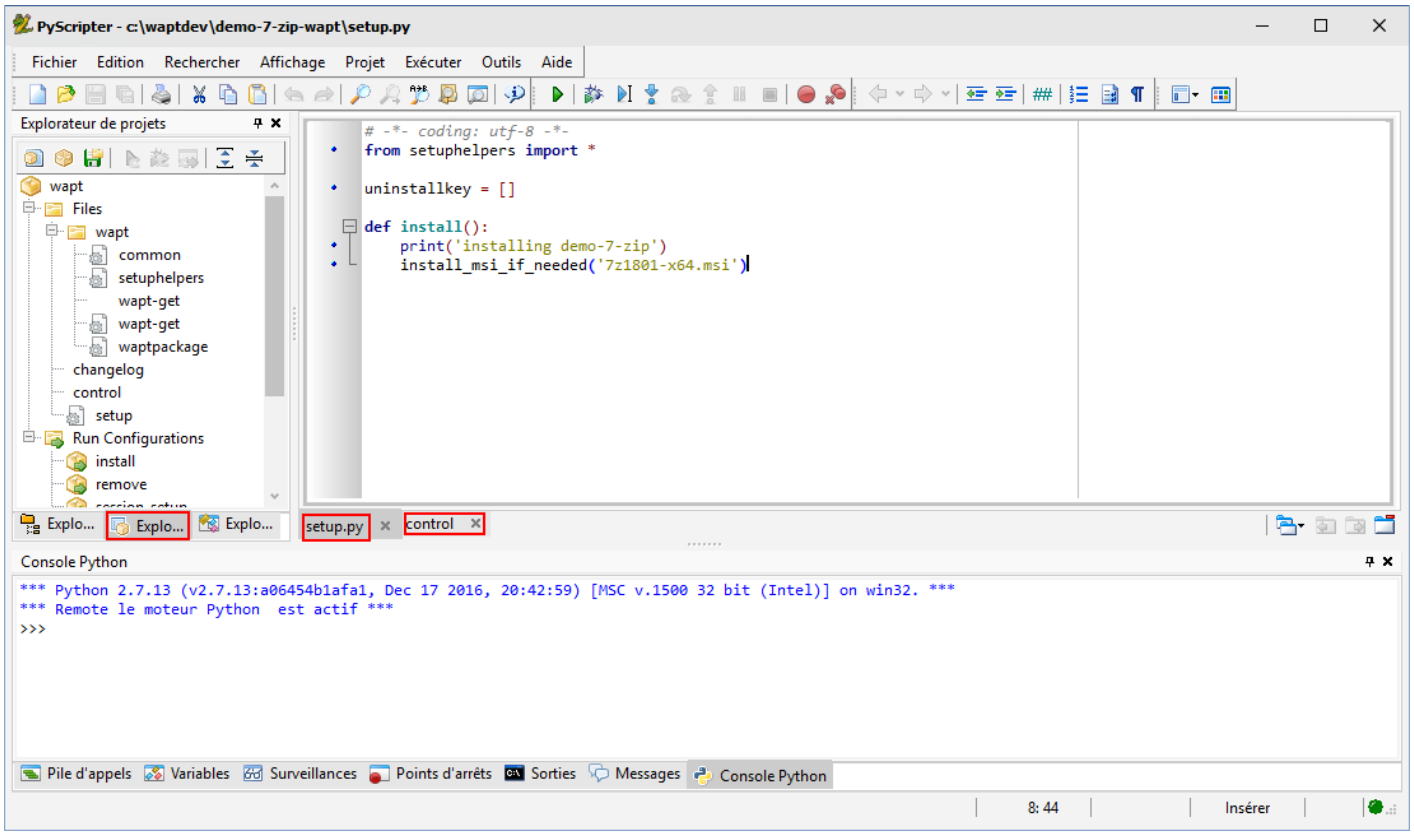


Fig. 5: PyScripter - Customizing a WAPT package within PyScripter

33.2.3 Presentation of PyScripter

PyScripter project explorer

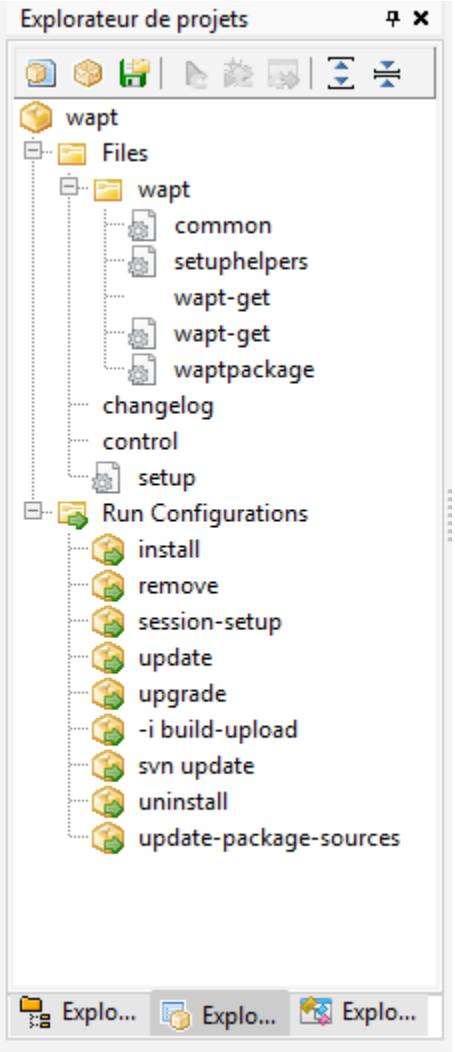


Fig. 6: PyScripter - Navigating a project within the PyScrypter file explorer

The PyScripter project explorer lists the different files that you might need, notably the `control` file and the `setup.py` file.

Run Configurations

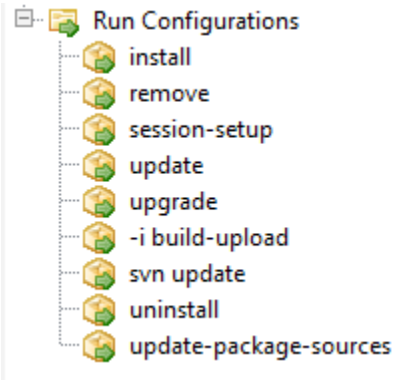


Fig. 7: PyScripter - Navigating the Run configurations of a project in PyScripter

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

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*.

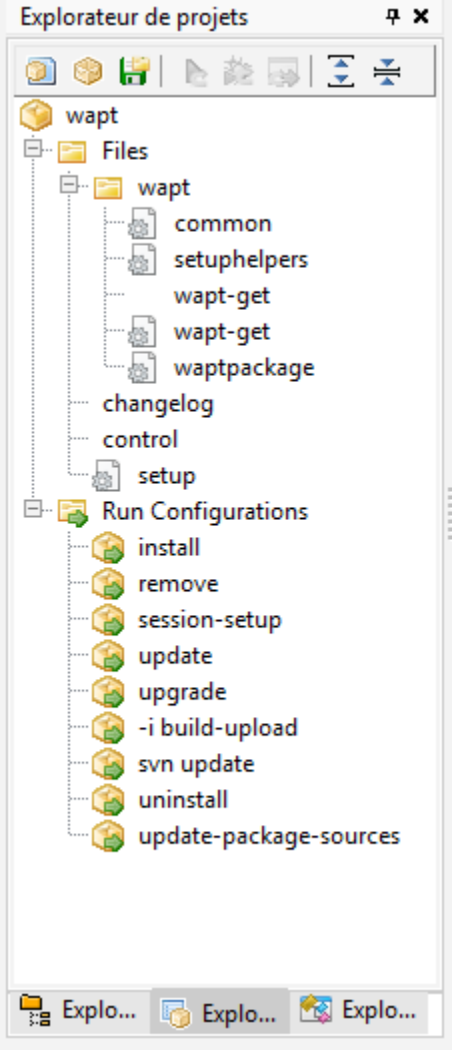


Fig. 8: PyScripter - Editing a WAPT package with PyScripter

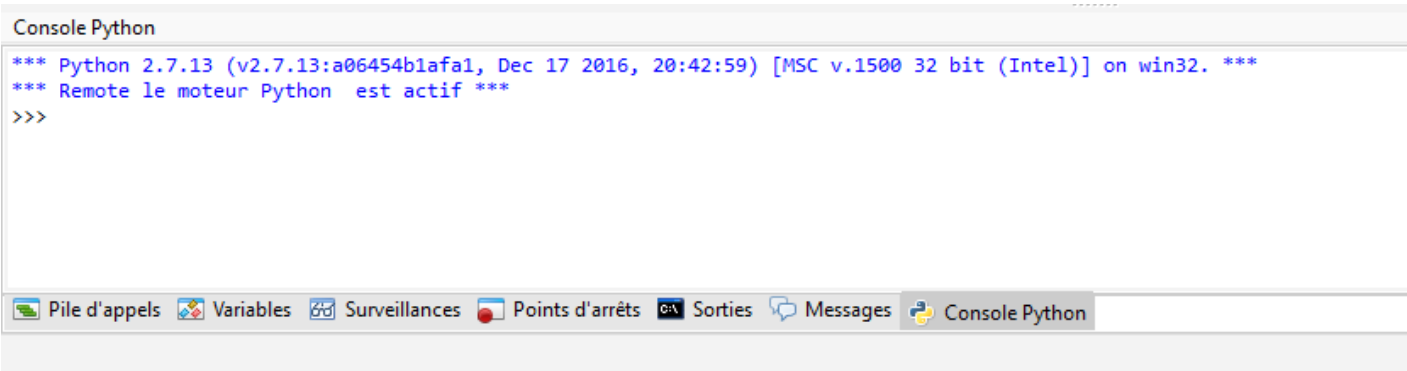
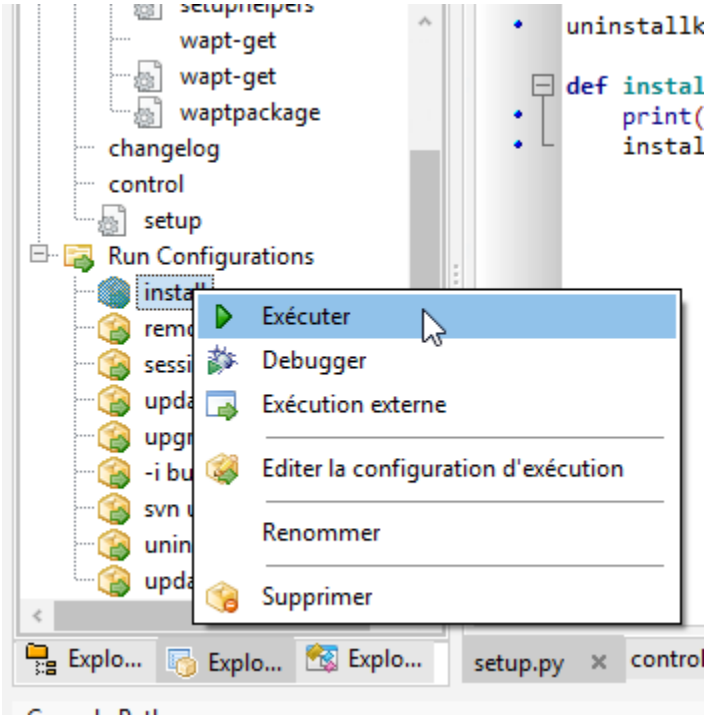


Fig. 9: PyScripter - Running the Python console from within Pyscripter

Testing locally the installation of the WAPT package

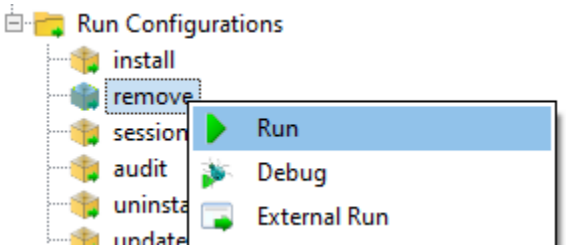
You can then test the launch of an installation on your development station.



The PyScripter console allows you to check whether the installation went well.

Testing locally the uninstallation of the WAPT package

You can then test the uninstall script on your development station.



The PyScripter console allows you to check whether the uninstallation went well.

33.3 Creating your first WAPT Packages

33.3.1 Packaging .msi packages (example)

For this example we will take **tightvnc**.

You can download it here: https://www.tightvnc.com/download.php

Now, you can then generate your package template, please refer to the *documentation for creating packages from the WAPT Console*.

Edit the `control` file (`architecture`, `impacted_process`, `target_os`, `description`, `maintainer` ...). For more information, visit the *documentation on the control file structure*.

Your **PyScripter** opens, go to your `setup.py`:

```
# -*- 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 function will test whether a version of the software is already installed on the host 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 1: List of arguments available with *install_msi_if_needed*

Options (Default Option)	Description
`min_version` (default `None`)	Defines the minimal version above which the software will update.
`killbefore` (default `None`)	Lists the programs to kill before installing the package.
`accept_returncodes` (default `[0,3010]`)	Defines the accepted codes other than 0 and 3010 returned by the function.
`timeout` (default `300`)	Defines the maximum installation wait time (in seconds).
`properties` (default `None`)	Defines the additional properties to pass as arguments to MSI setup file.
`get_version` (default `None`)	Defines the value passed as parameter to control the version number instead of the value returned by the *installed_softwares* function.
`remove_old_version` (default `False`)	Automatically removes an older version of a software whose *uninstall key* is identical.
`force` (default `False`)	Forces the installation of the software even though the same *uninstall key* has been found.

The **wapt-get 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.

You also do not have to fill in `killbefore` if the value specified in the `impacted_process` field of the `control` file is correct.

Note: The `setup.py` could have looked like this, but the method is less elegant because it does less checking:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ["{8B9896FC-B4F2-44CD-8B6E-78A0B1851B59}"]

def install():
    print('installing tis-tightvnc')
    run('msiexec /norestart /q /i "tightvnc-2.8.5-setup-64bit.msi"')
```

Run the installation and see what happens when the software is already installed.

```
wapt-get -ldebug install C:\waptdev\tis-tightvnc-wapt
Installing WAPT file C:\waptdev\tis-tightvnc-wapt
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)
```

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')
    install_msi_if_needed('tightvnc-2.8.5-setup-64bit.msi', properties = properties )
```

Note: The `setup.py` could have looked like this, but the method is less elegant because it does less checking:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = ["{8B9896FC-B4F2-44CD-8B6E-78A0B1851B59}"]

def install():
```

(continues on next page)

(continued from previous page)

```
    print('installing tis-tightvnc')
    run('msiexec /norestart /q /i "tightvnc-2.8.5-setup-64bit.msi" SERVER_REGISTER_AS_
↪SERVICE=0 SERVER_ADD_FIREWALL_EXCEPTION=0')
```

Video demonstration

https://youtu.be/Z6wr6emPGCU

33.3.2 Packaging .exe packages (example)

- Download the *.exe* installer from a reliable source.

  Download the installer in exe format Firefox ESR x64 on https://download.mozilla.org/?product=firefox-esr-latest-ssl&os=win64.

- 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*.
- Then generate your package template, please refer to the *documentation for creating packages from the WAPT Console*. **PyScripter** loads up and opens the *.exe* package project.
- Edit the `control` file (`architecture`, `impacted_process`, `target_os`, `description`, `maintainer` ...). For more information, visit the *documentation on the control file structure*.
- 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
impacted_process :
```

  - Modified `control` file.

```
package          : tis-firefox-esr
version          : 52.6.0-1
architecture     : all
```

(continues on next page)

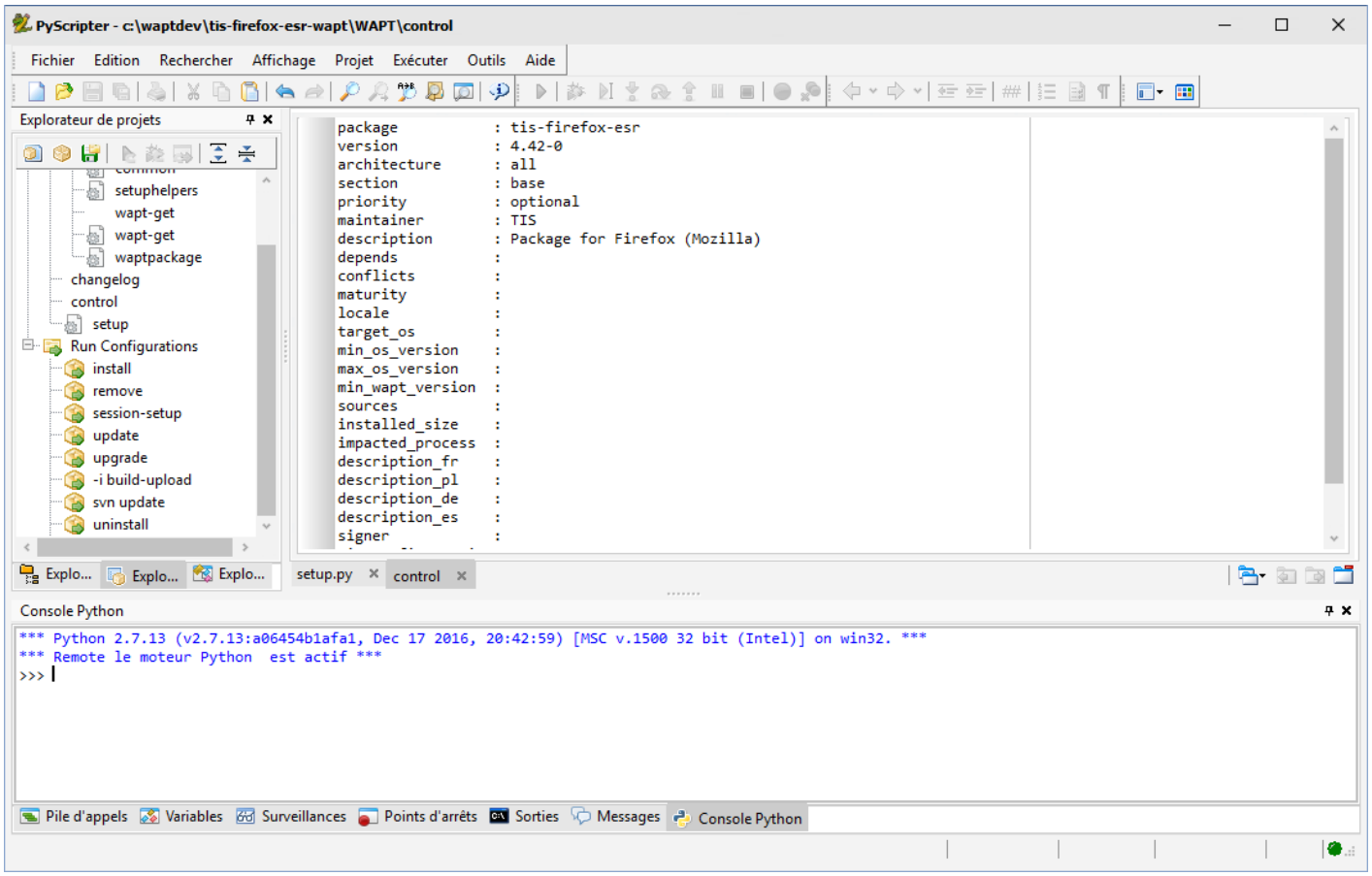


Fig. 10: PyScripter - Opening the FirefoxESR WAPT package

(continued from previous page)

```
section          : base
priority         : optional
maintainer       : Tranquil-IT Systems
description      : Mozilla Firefox 52.6.0 ESR
impacted_process : firefox.exe
```

A sub-version *-1* has been appended to the software version number; it is the packaging version of the WAPT package.

It allows the Package Developer to release several WAPT package versions of the same software, very useful for very rapid and iterative development.

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.

```
# -*- 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='',min_version="4.42.
↪0.0")
```

Table 2: List of arguments available with *install_exe_if_needed*

Options (Default Option)	Description
`silentflags` (default `None`)	Defines the silent parameters to pass as arguments to the installer.
`key` (default `None`)	Defines the software *uninstall key*.
`min_version` (default `None`)	Defines the minimal version above which the software will update.
`killbefore` (default `None`)	Lists the programs to kill before installing the package.
`accept_returncodes` (default `[0,3010]`)	Defines the accepted codes other than 0 and 3010 returned by the function.
`timeout` (default `300`)	Defines the maximum installation wait time (in seconds).
`get_version` (default `None`)	Defines the value passed as parameter to control the version number instead of the value returned by the *installed_softwares* function.
`remove_old_version` (default `False`)	Automatically removes an older version of a software whose *uninstall key* is identical.
`force` (default `False`)	Forces the installation of the software even though the same *uninstall key* has been found.

The package will then have this behavior:

- Firefox will install only if Firefox 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.

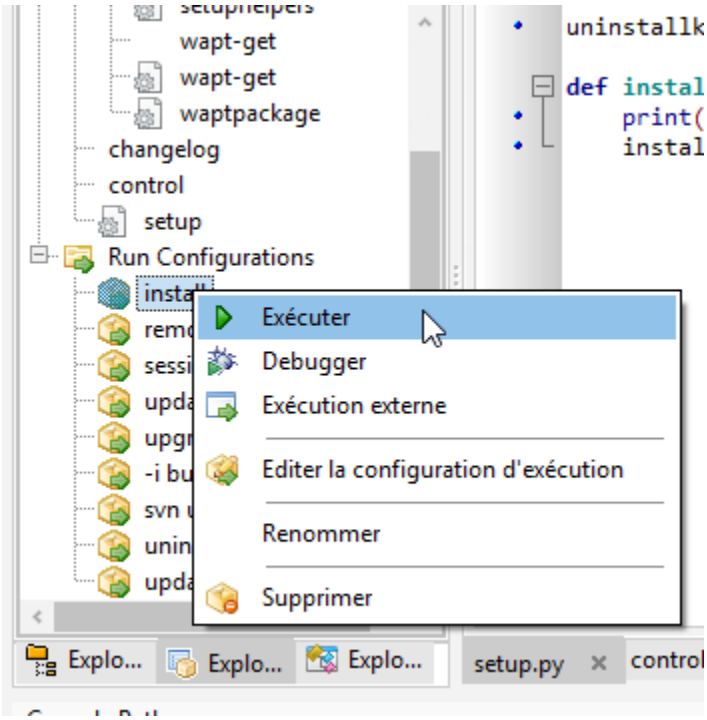
- On installing, the running **firefox.exe** processes will be killed (with the value indicated in `impacted_process` of the `control` file).
- The function will add by itself the *uninstall key*, so leave the *uninstall key* argument empty.
- When finishing the installation of the package, the function will check that the *uninstall key* is present on the host and that the version of Firefox is greater than 45.5.0; if this not the case, the package will be flagged as **ERROR**.

Finding the uninstallation key

Unlike *.msi* files, the key to uninstall an *.exe* is not in the file properties.

So you need to install the software first to know the uninstall key.

Therefore you **MUST** start once the installation from **PyScripter** with the *run configuration* and then *install*.



Once the software is installed, go to the WAPT Console, then find your development host.

In the *Software inventory* tab find the software title and copy the value indicated in the uninstallkey column.

You **MUST** also check the value of the version with the value indicated in `min_version` in your `setup.py`.

Modify your `setup.py` with the new parameters:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    print('installing tis-firefox-esr')
```

(continues on next page)

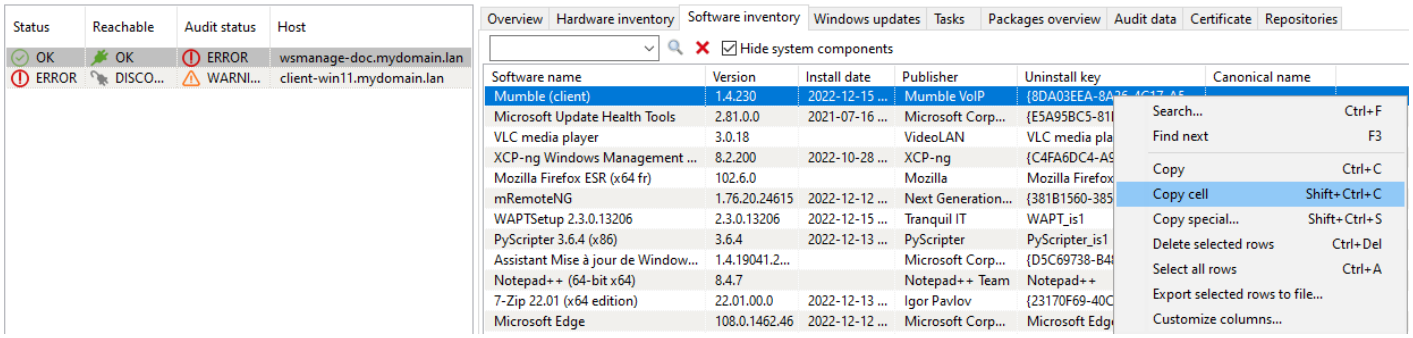
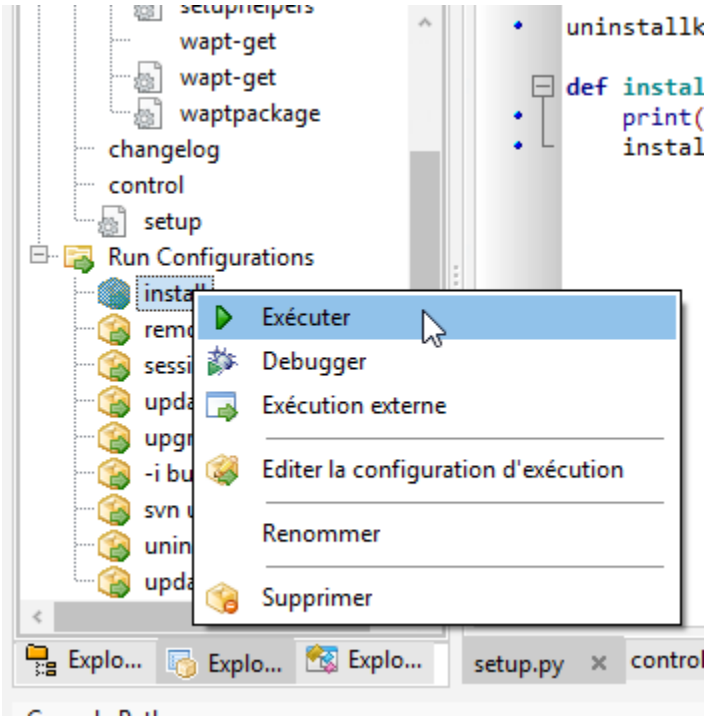


Fig. 11: Retrieving an uninstallkey from the WAPT Console

(continued from previous page)

```
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",silentflags="-ms",key='Mozilla Firefox 45.5.
↪0 ESR (x64 fr)',min_version="45.5.0")
```

To test that your key works correctly, you **MUST** run an installation again in **PyScripter**.



WAPT should not attempt to install the software because it is already present, the following message should display:

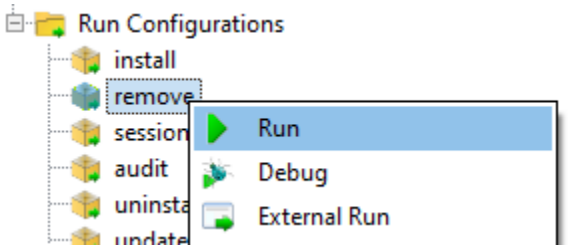
```
>>>
*** Remote Interpreter Reinitialized ***
Command Line : install "c:\waptdev\tis-firefox-esr_x64_PROD_fr-wapt\WAPT\.."
Using config file: C:\Program Files (x86)\wapt\wapt-get.ini
Installing WAPT files c:\waptdev\tis-firefox-esr_x64_PROD_fr-wapt
Exe setup Firefox_Setup_78.7.1esr.exe already installed. Skipping
```

(continues on next page)

(continued from previous page)

```
Results:

=== install packages ===
c:\waptdev\tis-firefox-esr_x64_PROD_fr-wapt | tis-firefox-esr (78.7.1-102)
```

Now you can now test the uninstallation:



You can now build and upload your package, please refer to the *documentation for build and upload packages from the WAPT Console*.

Note: If you leave the uninstallkey blank, uninstalling your package will not work.

Special case of a non-silent uninstaller

In some particular cases, 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():
    install_exe_if_needed("setup.exe",
                          silentflags="/s",
                          key='{D9E87643-0005-447E-9111-78697A9C1595}',
                          min_version="14.0")
    uninstallkey.remove('{D9E87643-0005-447E-9111-78697A9C1595')

def uninstall():
    run(r'"C:\Program Files\Kutl\uninstall.exe" /supersilent')
```

Hint: The uninstall feature can also be used to run code in addition to uninstalling software, ex: delete folder, delete shortcut ...

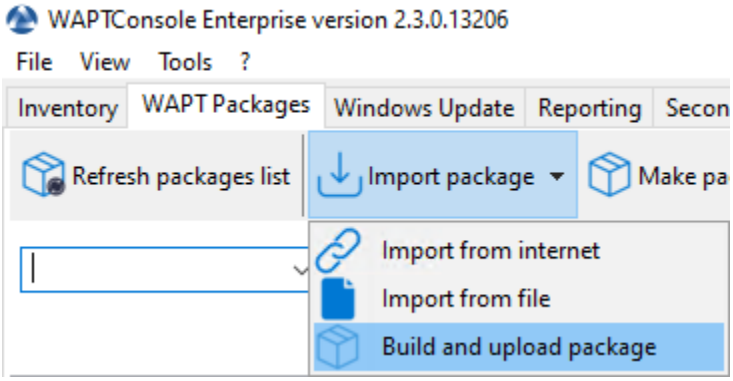
Video demonstration

https://youtu.be/z_EN2CBCTcY

33.3.3 Packaging empty packages

33.3.4 Building the package and sending it to the WAPT Server

- Once the package is ready, build it and send it to the WAPT Server.



- Select the package in the `c:\waptdev` folder.
- Confirm the selected package.

You have just uploaded your first wapt package.

Note: *A command line method is available here.*

Warning: Once your package has uploaded, refresh the package list using the *Refresh packages list* button or by pressing `F5` on your keyboard.

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 successful 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
```

(continues on next page)

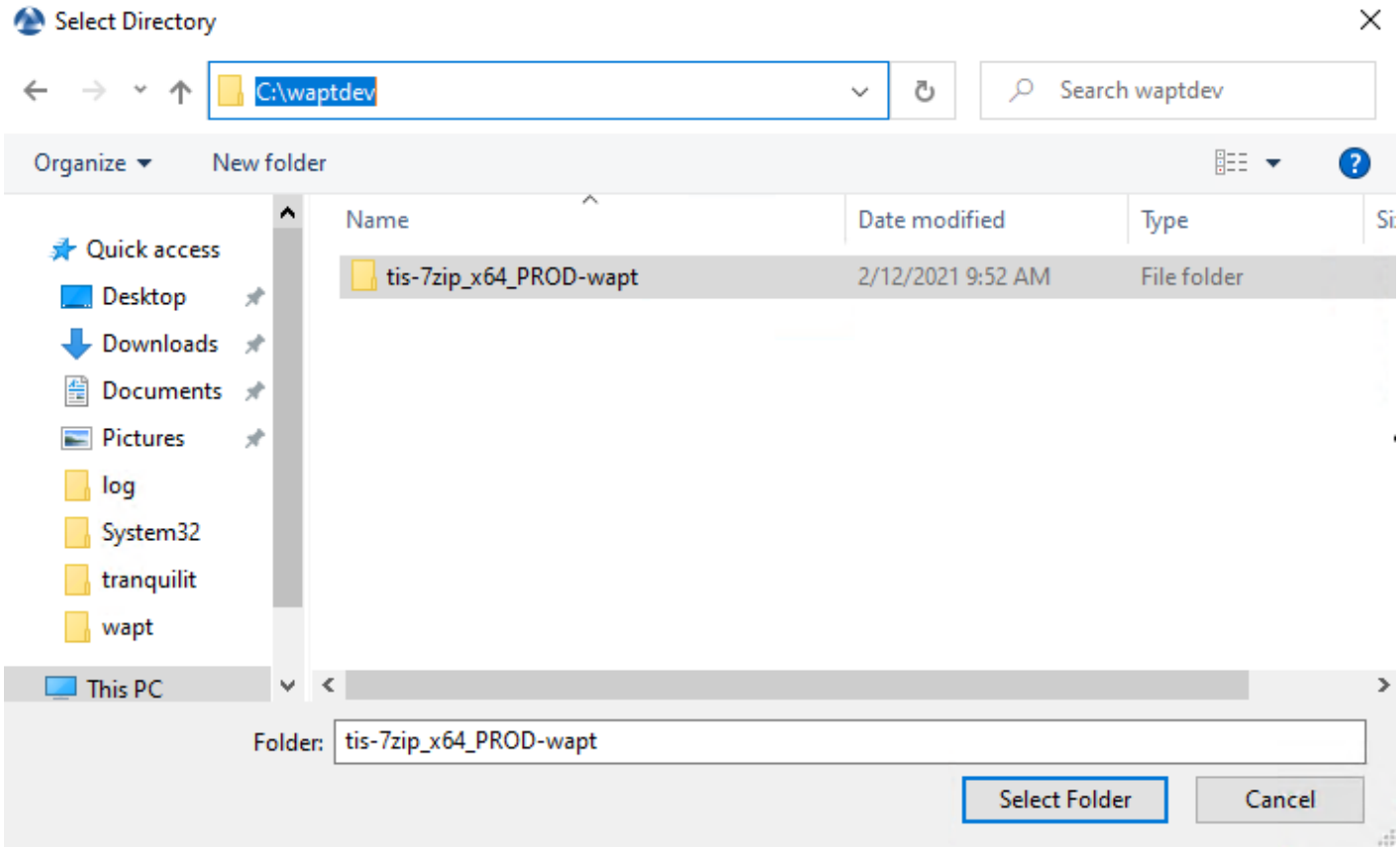


Fig. 12: Browser window for selecting the WAPT package to import into the private repository

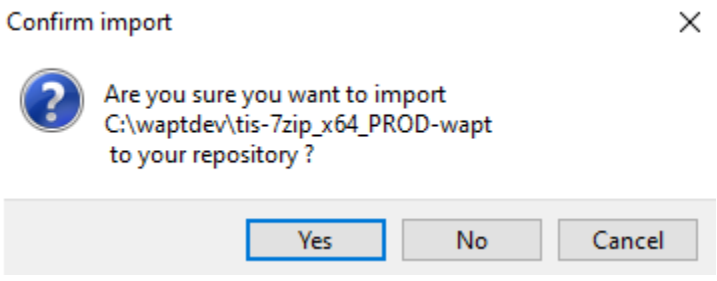


Fig. 13: WAPT Console dialog box for confirming the importation of a WAPT package into the private repository

(continued from previous page)

```
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.

CHAPTER

THIRTYFOUR

HOW TO CODE WAPT PACKAGES

34.1 Simple examples of commonly used setuphelper functions

Presentation of several functions implemented in *Setuphelpers* and frequently used to develop WAPT packages.

34.1.1 Testing and manipulating folders and files

Creating a path recursively

Command **makepath** makes the path variable for `C:\Program Files (x86)\Mozilla\Firefox`.

```
makepath(programfiles,'Mozilla','Firefox')
```

Creating and destroying directories

Command **mkdirs** creates the directory `C:\test`.

```
mkdirs('C:\\test')
```

Command **remove_tree** destroys the directory `C:\tmp\target`.

```
remove_tree(r'C:\tmp\target')
```

Checking if a path is a file or a folder

Command **isdir** checks whether `C:\Program Files (x86)\software` is a directory.

```
isdir(makepath(programfiles32,'software')):
    print('The directory exists')
```

Command **isfile** checks whether `C:\Program Files (x86)\software\file` is a file.

```
isfile(makepath(programfiles32,'software','file')):
    print('file exist')
```

Checking whether a directory is empty

Command **dir_is_empty** checks that directory `C:\Program Files (x86)\software` is empty.

```
dir_is_empty(makepath(programfiles32,'software')):
    print('dir is empty')
```

Copying a file

Command **filecopyto** copies `file.txt` into the `C:\Program Files (x86)\software` directory.

```
filecopyto('file.txt',makepath(programfiles32,'software'))
```

Copying a directory

Command **copytree2** copies the `sources` folder into the `C:\projet` directory.

```
copytree2('sources','C:\\projet')
```

34.1.2 Manipulating registry keys

Checking the existence of a registry key

Command **registry_readstring** checks if registry key *{8A69D345-D564-463c-AFF1-A69D9E530F96}* exists in registry path `SOFTWARE\Google\Update\Clients` of *HKEY_LOCAL_MACHINE*.

```
if registry_readstring(HKEY_LOCAL_MACHINE, "SOFTWARE\\Google\\Update\\Clients\\{8A69D345-D564-463c-
↪AFF1-A69D9E530F96}", 'pv'):
    print('key exist')
```

Showing the value of a registry key

Command **registry_readstring** reads the value *{8A69D345-D564-463c-AFF1-A69D9E530F96}* stored in the registry path `SOFTWARE\Google\Update\Clients` of *HKEY_LOCAL_MACHINE*.

```
print(registry_readstring(HKEY_LOCAL_MACHINE, r'SOFTWARE\Google\Update\Clients\{8A69D345-D564-463c-
↪AFF1-A69D9E530F96}', 'pv'))
```

Modifying the value of a registry key

Command **registry_setstring** modifies the value of the registry key *TOUVersion* stored in the registry path `SOFTWARE\Microsoft\Windows Live` of *HKEY_CURRENT_USER*.

```
registry_setstring(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows Live\\Common",'TOUVersion','16.
↪0.0.0', type=REG_SZ)
```

34.1.3 Creating and destroying shortcuts

With WAPT setuphelper it is possible to create different types of shortcuts.

Creating a desktop shortcut for all users

Command **create_desktop_shortcut** 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.

```
create_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\waptconsole.
↪exe')
```

Removing a desktop shortcut for all users

Command **remove_desktop_shortcut** deletes the *WAPT Console Management* shortcut from the folder `C:\Users\Public`; the shortcut is deleted for all users.

```
remove_desktop_shortcut('WAPT Console Management')
```

Firefox places a shortcut on the all users desktop, we are going to delete it.

We will use the **remove_desktop_shortcut** function:

- Modify your `setup.py` and use the function like this.

```
# -*- coding: utf-8 -*-
from *SetupHelpers* import *

uninstallkey = []

def install():
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",silentflags="-ms",key='Mozilla Firefox␣
↪45.5.0 ESR (x64 fr)',min_version="45.5.0")
    remove_desktop_shortcut('Firefox')
```

- If you restart the installation from **PyScripter**, you will notice that the “all users” desktop shortcut has disappeared.

Creating a menu shortcut for an application

Command **create_programs_menu_shortcut** creates the shortcut *WAPT Console Management* into start menu pointing to `C:\Program Files (x86)\wapt\waptconsole.exe`; the shortcut is available for all users.

```
create_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\waptconsole.
↪exe')
```

Removing a menu shortcut for an application

Command **remove_programs_menu_shortcut** deletes the *WAPT Console Management* shortcut from start menu.

```
remove_programs_menu_shortcut('WAPT Console Management')
```

Creating a desktop shortcut for a logged in user

Hint: These functions are used in session_setup context.

Command **create_user_desktop_shortcut** creates the shortcut *WAPT Console Management* on user desktop pointing to `C:\Program Files (x86)\wapt\waptconsole.exe`.

```
create_user_desktop_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

Removing a desktop shortcut for a logged in user

Command **remove_user_desktop_shortcut** deletes the *WAPT Console Management* shortcut from the logged in user's desktop.

```
remove_user_desktop_shortcut('WAPT Console Management')
```

Creating a menu shortcut to an application for a specific user

Hint: These functions are used in session_setup context.

Command **create_user_programs_menu_shortcut** creates the shortcut *WAPT Console Management* on user start menu pointing to `C:\Program Files (x86)\wapt\waptconsole.exe`.

```
create_user_programs_menu_shortcut(r'WAPT Console Management',target=r'C:\Program Files (x86)\wapt\
↪waptconsole.exe')
```

Removing a menu shortcut to an application for a specific user

Command **remove_user_programs_menu_shortcut** deletes the *WAPT Console Management* shortcut from the logged in user's start menu.

```
remove_user_programs_menu_shortcut('WAPT Console Management')
```

34.1.4 Manipulating ini files

Read a value in a section of a ini file

Command **inifile_readstring** will read a value from a key and a section of a ini file.

```
inifile_writestring("file.ini","global","key")
```

Write a value in a section of a ini file

Command **inifile_writestring** will modify a value from a key and a section of a ini file.

```
inifile_writestring("file.ini","global","key","value")
```

Delete a key in a section of a ini file

Command **inifile_deleteoption** will delete a key in a given section of a ini file.

```
inifile_deleteoption("file.ini","global","key")
```

Delete an entire section of a ini file

Command **inifile_deletesection** will delete a section of a ini file and all of its content.

```
inifile_deletesection("file.ini","global")
```

34.1.5 Windows environment/ Software/ Services

Retrieving the version of a file

Command **get_file_properties** shows package properties.

```
get_file_properties(makepath(programfiles32,'InfraRecorder','infrarecorder.exe'))['ProductVersion']
```

Checking the Windows version

Command **windows_version** checks that the Windows version is strictly inferior to *6.2.0*.

```
windows_version()<Version('6.2.0'):
```

Hint: For more informations you can visit Microsoft Windows version number.

Checking for 64bits architecture

Command **iswin64** checks that the system architecture is 64bits.

```
if iswin64():
    print('Pc x64')
else:
    print('Pc not x64')
```

Checking for the Program Files variable

- programfiles;

  ```
  print(programfiles())
  ```

- programfiles32;

  ```
  print(programfiles32())
  ```

- programfiles64;

  ```
  print(programfiles64())
  ```

Each command returns a different *ProgramFiles* location.

For example, command **programfiles64** returns native Program Files directory, eg. `C:\Program Files (x86)` on either win64 or win32 architecture and **programfiles()** will return the path of the 32bit Program Files directory, eg. `Programs Files (x86)` on win64 architecture, and `Programs Files` on win32 architecture.

Checking for the AppData variable

user_appdata/ user_local_appdata

Hint: These functions are used with **session_setup**

Command **user_appdata** returns roaming *AppData* profile path of logged on user (`C:\Users\%username%\AppData\Roaming`).

```
print(user_appdata())
```

Command **user_local_appdata** returns the local *AppData* profile path of the logged on user (`C:\Users\%username%\AppData\Local`).

```
print(user_local_appdata())
```

Disabling temporarily the wow3264 file redirection

Command **disable_file_system_redirection** disables wow3264 redirection in the current context.

```
with disable_file_system_redirection():
    filecopyto('file.txt',system32())
```

Obtaining the current logged in user

Command **get_current_user** shows the currently logged on username.

```
print(get_current_user())
```

Obtaining the computer name

Command **get_computername** shows the name of the computer.

```
print(get_computername())
```

Obtaining the AD domain to which the computer is joined

Command **get_domain_fromregistry** returns the FQDN of the computer.

```
get_domain_fromregistry()
```

34.1.6 Actions on installed software

Checking installed software

Command **installed_softwares** returns the list of installed software on the computer from registry in an array.

```
installed_softwares('winscp')
```

```
[{'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}]
```

Get uninstall command with registry

Command **uninstall_cmd** returns the silent uninstall command.

```
uninstall_cmd('winscp3_is1')
```

```
"C:\Program Files\WinSCP\unins000.exe" /SILENT
```

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*.

Kill task

Command **killalltasks** kills all tasks with the specified name.

```
killalltasks('firefox')
```

34.1.7 Using control file fields

It's possible to use control file informations on `setup.py`

Get packages version

```
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.

Get software name

Todo: upcoming documentation

34.1.8 Managing a WAPT package with another 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.

Forgetting 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*.

34.2 Improving my package

34.2.1 Copying a file

It is possible to configure **Firefox** with a `policies.json` file. See https://github.com/mozilla/policy-templates/blob/master/README.md.

This file **MUST** be placed in the `distribution` folder at the root of Firefox.

To help you create this `policies.json` file you can use the enterprise policy generator generator for Firefox.

When you have generated your `policies.json` file, place it in `c:\waptdev\prefix-firefox-esr-wapt\policies.json`.

The `distribution` folder at the root of Firefox may not exist, so we will test its existence and create it with the **mkdirs** command if it does not exist:

```
if not isdir(r'C:\Program Files\Mozilla Firefox\distribution'):
    mkdirs(r'C:\Program Files\Mozilla Firefox\distribution')
```

Important: If you have backslashes in your path, you should always put an **r** in front of the string, like in the previous example.

You will also need to use the `filecopyto` function to copy the `policies.json` file:

```
filecopyto('policies.json',r'C:\Program Files\Mozilla Firefox\distribution')
```

Hint: There is no need to put the full path for the source file since the `policies.json` file is at the root of the WAPT package, so we use the relative path.

Modify your `setup.py`:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",silentflags="-ms",key='Mozilla Firefox 45.5.
↪0 ESR (x64 fr)',min_version="45.5.0")
    remove_desktop_shortcut('Firefox')

    if not isdir(r'C:\Program Files\Mozilla Firefox\distribution'):
        mkdirs(r'C:\Program Files\Mozilla Firefox\distribution')

    filecopyto('policies.json',r'C:\Program Files\Mozilla Firefox\distribution')
```

Your package is now ready to apply a configuration. You can launch an installation with **PyScripter** and validate that the package works according to your objective.

Finally, launch your **Firefox** to verify that it will work for your users.

34.2.2 Uninstalling unwanted versions

Hint: At each step of these examples you can run an installation to test the result.

In our case we want to uninstall the non ESR version of **Firefox**.

We will look for the other software installed on the host to check if a non-esr version of **Firefox** is installed.

To reproduce our example, download and install the latest consumer version of Firefox:

- To search unwanted version of **Firefox** we will use the `installed_softwares` function. This function returns a dictionary list containing the software properties:

```
print(installed_softwares('Firefox'))

[
   {
 'install_date': '',
   'install_location': 'C:\\Program Files\\Mozilla Firefox',
   'key': 'Mozilla Firefox 78.7.1 ESR (x64 fr)',
   'name': 'Mozilla Firefox 78.7.1 ESR (x64 fr)',
   'publisher': 'Mozilla',
   'system_component': 0,
   'uninstall_string': '"C:\\Program Files\\Mozilla Firefox\\uninstall\\helper.exe"',
   'version': '78.7.1',
   'win64': True
 },

   {
  'install_date': '',
    'install_location': 'C:\Program Files (x86)\\Mozilla Firefox',
    'key': 'Mozilla Firefox 79.0 (x86 fr)',
    'name': 'Mozilla Firefox 79.0 (x86 fr)',
    'publisher': 'Mozilla',
    'system_component': 0,
    'uninstall_string': '"C:\Program Files (x86)\\Mozilla Firefox\\uninstall\\helper.exe"',
    'version': '79.0',
    'win64': False
  }
]
```

- Check the name of each software.

```
for uninstall in installed_softwares('Mozilla Firefox'):
    print(uninstall['name'])
```

- Show the name of each software found.

```
for uninstall in installed_softwares('Mozilla Firefox'):
    if not 'ESR' in uninstall['name']:
        print(uninstall['name'])
```

- Show the name of each software found which does not include the string *ESR* in its name and its uninstallkey.

```
for uninstall in installed_softwares('Mozilla Firefox'):
    if not 'ESR' in uninstall['name']:
        print(uninstall['name'])
        print('Uninstall ' + uninstall['key'])
```

We will now use a WAPT trick using the `uninstall_cmd` function:

- Install cmd accepts an uninstall key as an argument and will send the command to run to start the silent uninstall.

```
for uninstall in installed_softwares('Mozilla Firefox'):
    if not 'ESR' in uninstall['name']:
```

(continues on next page)

(continued from previous page)

```
        print(uninstall['name'])
        print('Uninstall ' + uninstall['key'])
        silent_uninstall = uninstall_cmd(uninstall['key'])
        print('Run ' + silent_uninstall)
```

- Start the uninstallation.

```
for uninstall in installed_softwares('Mozilla Firefox'):
    if not 'ESR' in uninstall['name']:
        print(uninstall['name'])
        print('Uninstall ' + uninstall['key'])
        silent_uninstall = uninstall_cmd(uninstall['key'])
        print('Run ' + silent_uninstall)
        run(silent_uninstall)
```

We can also uninstall the Mozilla maintenance service:

```
for uninstall in installed_softwares('MozillaMaintenanceService'):
    run(uninstall_cmd(uninstall['key']))
```

- Finally, modify your `setup.py`:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():
    #Install firefox if necessary
    install_exe_if_needed("Firefox Setup 45.5.0esr.exe",silentflags="-ms",key='Mozilla Firefox␣
↪45.5.0 ESR (x64 fr)',min_version="45.5.0")

    #Removal of the firefox shortcut on the all user desktop
    remove_desktop_shortcut('Firefox')

    #Creation of the distribution folder if it does not exist
    if not isdir(r'C:\Program Files\Mozilla Firefox\distribution'):
        mkdirs(r'C:\Program Files\Mozilla Firefox\distribution')

    #Copy of the policies.json file found at the root of the package in the destination of the␣
↪distribution folder
    filecopyto('policies.json',r'C:\Program Files\Mozilla Firefox\distribution')

    #For each Mozilla Firefox installed
    for uninstall in installed_softwares('Mozilla Firefox'):
        #If the software does not have the word ESR in the name
        if not 'ESR' in uninstall['name']:
            print(uninstall['name'])
            print('Uninstall ' + uninstall['key'])
```

(continues on next page)

(continued from previous page)

```
            #Looking for how we can uninstall it silently
            silent_uninstall = uninstall_cmd(uninstall['key'])
            print('Run ' + silent_uninstall)

            #We launch the previous command.
            run(silent_uninstall)

    #Uninstalling mozilla maintenance service
    for uninstall in installed_softwares('MozillaMaintenanceService'):
        run(uninstall_cmd(uninstall['key']))
```

Your code now handles the uninstallation of unwanted versions of **Firefox**.

34.2.3 Improving setup.py to use variables

Examples of variable usage:

```
version_firefox = "45.0"

uninstallkey = "Mozilla Firefox " + version_firefox + " ESR (x64 fr)"
print(uninstallkey)

uninstallkey = "Mozilla Firefox %s ESR (x64 fr)" % (version_firefox)
print(uninstallkey)

uninstallkey = "Mozilla Firefox {} ESR (x64 fr)".format(version_firefox)
print(uninstallkey)

uninstallkey = f"Mozilla Firefox {version_firefox} ESR (x64 fr)"
print(uninstallkey)
```

Important: The last example is the best example but this operation only works with **Python3**.

We can now use variables in our `setup.py`:

```
# -*- coding: utf-8 -*-
from setuphelpers import *

uninstallkey = []

def install():

    version_firefox = "45.5.0"

    #Install firefox if necessary
    install_exe_if_needed("Firefox Setup %sesr.exe" % version_firefox,silentflags="-ms",
↪key='Mozilla Firefox %s ESR (x64 fr)' % version_firefox,min_version=version_firefox)
```

(continues on next page)

(continued from previous page)

```
    #Removal of the firefox shortcut on the all user desktop
    remove_desktop_shortcut('Firefox')

    distribution_folder=r'C:\Program Files\Mozilla Firefox\distribution'

    #Creation of the distribution folder if it does not exist
    if not isdir(distribution_folder):
        mkdirs(distribution_folder)

    ... The rest of the code does not change ...
```

Hint: You can retrieve the version number shown in the `control` file like this:

```
version_firefox = control.get_software_version()
```

34.2.4 Customizing the user environment

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 is executed for each user using:

```
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: The *session_setup* script 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 the *session_setup* scripts that have been already been played.

Output example of `wapt-get session-setup ALL`:

Note: The logged in user *session_setup* has already previously 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 *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: `c:\cachefile`).

Example: creating 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():
```

(continues on next page)

(continued from previous page)

```
    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")

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")
```

- Now start the `session-setup` directly from **PyScripter**.

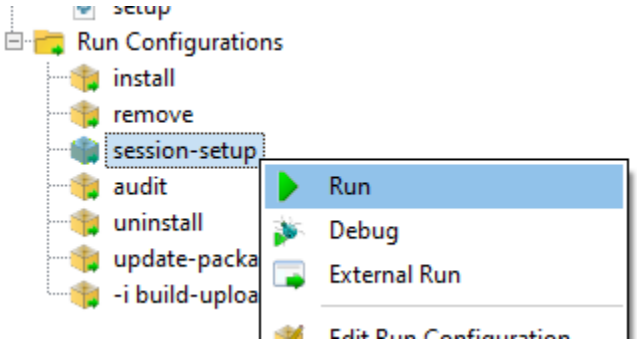


Fig. 1: PyScripter - running session-setup

- Finally, check that the icon is present on the desktop.

34.2.5 Using the audit functions for compliance

Note: This feature is available in the **Enterprise** version.

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 hosts 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

The `audit` tasks are launched once after every **`wapt-get upgrade`**, then regularly as defined by the `audit_schedule` attribute.

To manually launch an audit check, you may also use the following command:

```
wapt-get audit
```

Note: By default, the `audit` function will not launch if the audit is not necessary.

To force the execution, you may launch the following command:

```
wapt-get audit -f
```

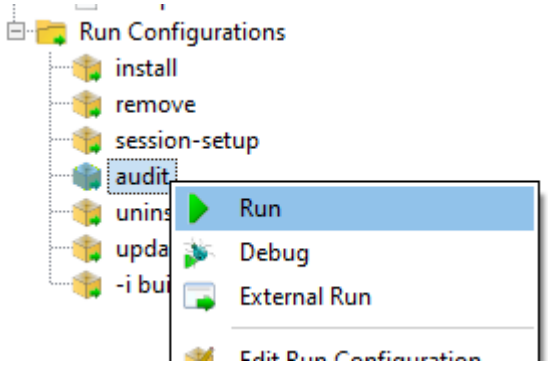
The `audit` script is defined in the package's `setup.py` with a function `def audit()`:

In this example, we are improving the Firefox package previously studied in this documentation.

- Add the `audit` function in the `setup.py`.

```
def audit():
    if isfile(r'C:\Program Files\Mozilla Firefox\distribution\policies.json'):
        print('File policies.json found')
        return "OK"
    else:
        print('File policies.json not found')
        return "ERROR"
```

- Start the audit from **`PyScripter`**.



- Test with the file then delete the `C:\Program Files\Mozilla Firefox\distribution\policies.json` file and test again with **`PyScripter`**.

You can directly see the status of the audit in the WAPT Console (Click on the package then on the audit column):

The audit function returns one of these 3 values:

- **OK**;
- **WARNING**;
- **ERROR**.

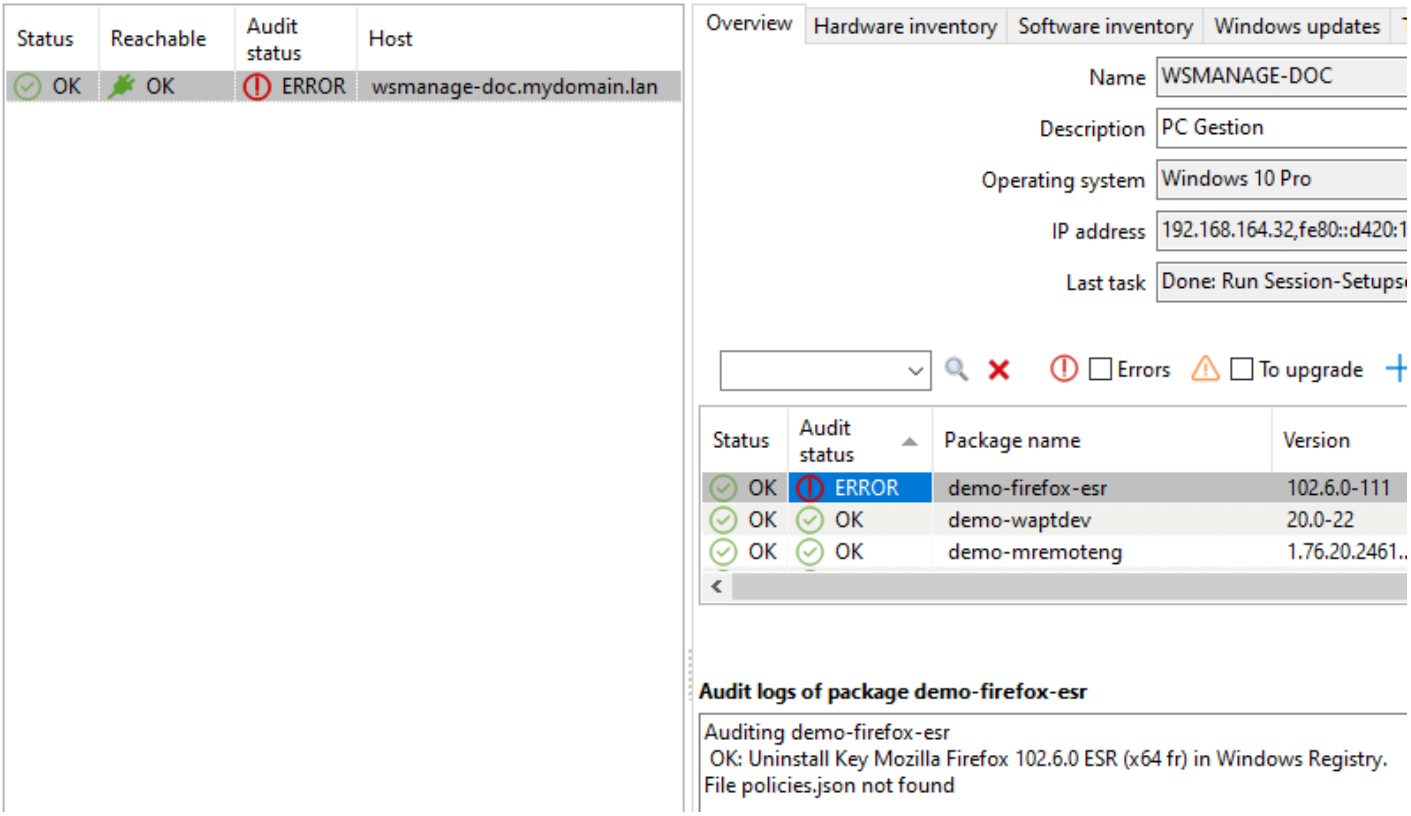


Fig. 2: Checking an audit status in the WAPT Console

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** instruct to copy the file(s) to a temporary folder when the WAPT package installs.

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 the WAPT package.

By default, if `audit_schedule` is empty, the audit task can be launched manually from the WAPT Console or be launched automatically if you have defined the option `waptaudit_task_period` in the `wapt-get.ini` of the WAPT Agent. For more information about the last method, please see *this documentation*.

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.

34.2.6 Auditing configurations to insure compliance

Note: This feature is available in the **Enterprise** version.

The `audit_data` function allows to make regular checks to desktop configurations and to centralize the results of these checks in the WAPT Console. There is historization and you can encrypt your data and decrypt it with your WAPT certificate.

For example you can:

- Change an administrator password, encrypt information and display it on your WAPT Console.
- Regularly check the modification your computer needs like CVE or GLPI inventory.
- Ascertain the security settings of a workstation and historize issues.

  The `audit_data` function is usable in the `audit` function only.

Working principle

The `audit_data` functions are launched if they are defined in the `def audit()` section of the `setup.py` file.

On the server side, audit data is stored in the HostAuditData table. The content of the table can be queried using the *Reporting* tab in the WAPT Console. The Data is automatically purged according to expiration date. When WAPT host `update_status()` is launched, the newer audit data is sent to the WAPT Server.

On the Client side, the audit data is stored in the host database with an expiration date (date_expiration) and the max count (max_count) of the stored data is defined in the code.

In this example, we are checking public IP on the computer.

- Add the `audit_data` function inside the `audit` function in the `setup.py`.

```
def audit():
  ip = wgets('https://api.ipify.org',verify_cert=False)
  print(f'My public IP address is: {ip}')
  WAPT.write_audit_data_if_changed('Public IP','log for %s' % get_computername(),ip,max_
↪count=5)
  return 'OK'
```

Here are the functions related to `audit_data`:

```
def write_audit_data_if_changed(self, section, key, value, ptype=None, value_date=None,␣
↪expiration_date=None, max_count=2, keep_days=None):
"""Write data only if different from last one
"""
def write_audit_data(self, section, key, value, ptype=None, value_date=None, expiration_
↪date=None, max_count=2, keep_days=None):
"""Stores in database a metrics, removes expired ones

    Args:
        section (str)
        key (str)
        value (any)
        value_date
        expiration_date (str) : expiration date of the new value
        max_count (int) : keep at most max_count value. remove oldest one.
        keep_days (int) : set the expiration date to now + keep_days days. override␣
↪expiration_date arg if not None

    Returns:
        None
"""
def read_audit_data(self, section, key, default=None, ptype=None):
    """Retrieve the latest value associated with section/key from database"""

def read_audit_data_set(self, section, key):
    """Retrieve all the values associated with section/key from database"""

def delete_audit_data(self, section, key):
```

(continues on next page)

(continued from previous page)

```
def read_audit_data_since(self, last_query_date=None):
    """Retrieve all the values since a date from database"""
```

34.2.7 Updating automatically a software package

Note: This part of the documentation is for advanced users of WAPT.

The `update_package` functions are very practical, they allow to gain a lot of time when needing 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():
      import re,requests,glob

      #Retrieving the last file name
      url = requests.head('https://download.mozilla.org/?product=firefox-esr-latest&os=win64',
↪proxies={}).headers['Location']
      filename = url.rsplit('/',1)[1].replace('%20',' ')

      #download of it if is not in the package
      if not isfile(filename):
          print('Downloading %s from %s'%(filename,url))
          wget(url,filename)

      #removing old exe with wrong name
      for fn in glob.glob('*.exe'):
          if fn != filename:
              remove_file(fn)

      # updates control version from filename, increment package version.
```

(continues on next page)

(continued from previous page)

```
    control.version = '%s-0'%(re.findall('Firefox Setup (.*)esr\.exe',filename)[0])
    control.save_control_to_wapt()
```

You may launch the *update_package* in **PyScripter**:

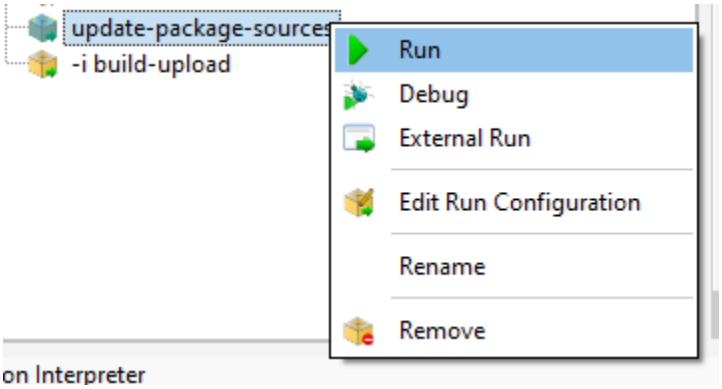


Fig. 3: PyScripter - Running an update-package-source

You will find many inspiring examples of *update_package* scripts in packages hosted in the Tranquil IT store.

34.2.8 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 is running.

Example with ADWCleaner

First, download Adwcleaner.

You can then generate your package template, please refer to the *documentation for creating packages from the WAPT Console*.

The file `C:\waptdev\tis-adwcleaner-wapt` is created.

Here you will find an example of a portable package that takes almost all the WAPT functions of a `setup.py`:

```
from setuphelpers import *

uninstallkey = []

exe_name = 'AdwCleaner.exe'
path_adw = makepath(programfiles,'AdwCleaner')
```

(continues on next page)

(continued from previous page)

```
path_exe = makepath(path_adw,exe_name)
nameshortcut = 'AdwCleaner'

def install():
    mkdirs(path_adw)
    filecopyto(exe_name,path_exe)
    create_desktop_shortcut(nameshortcut,path_exe)

def uninstall():
    remove_tree(path_adw)
    remove_desktop_shortcut(nameshortcut,path_exe)

def audit():
    if not isfile(path_exe):
        print('File not found')
        return "OK"
    else:
        print('File Found')
        return "ERROR"

def update_package():
    wget('https://downloads.malwarebytes.com/file/AdwCleaner',exe_name)
    control.version = get_file_properties(exe_name)['FileVersion'] + '-0'
    control.save_control_to_wapt()
```

34.2.9 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 KB4522355 MSU package from Microsoft Catalog website.
- Create a WAPT package template from the downloaded *.msu* file. In the WAPT Console, click on *Tools* → *Package Wizard*.
- 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, then build, then sign, then upload and finally affect the desired WAPT packages to your selected 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.

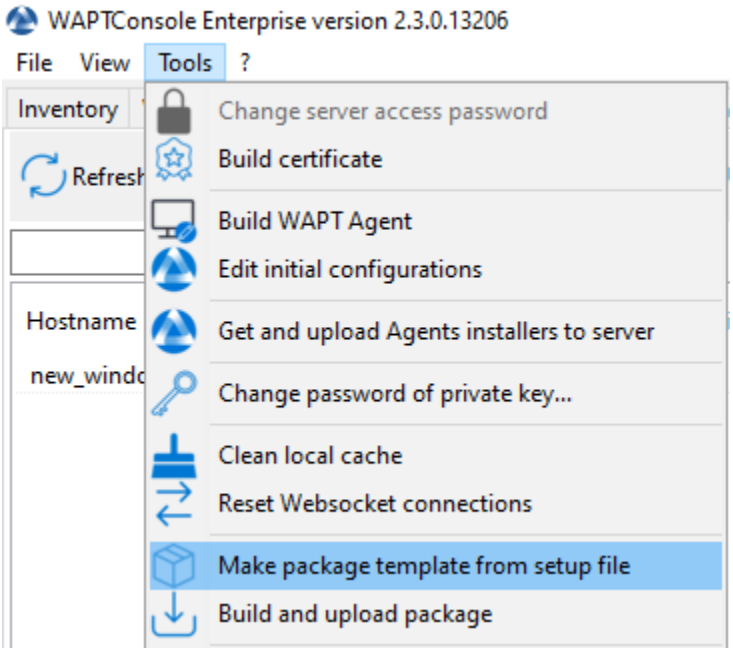


Fig. 4: PyScripter - WAPT Console window for creating a package template

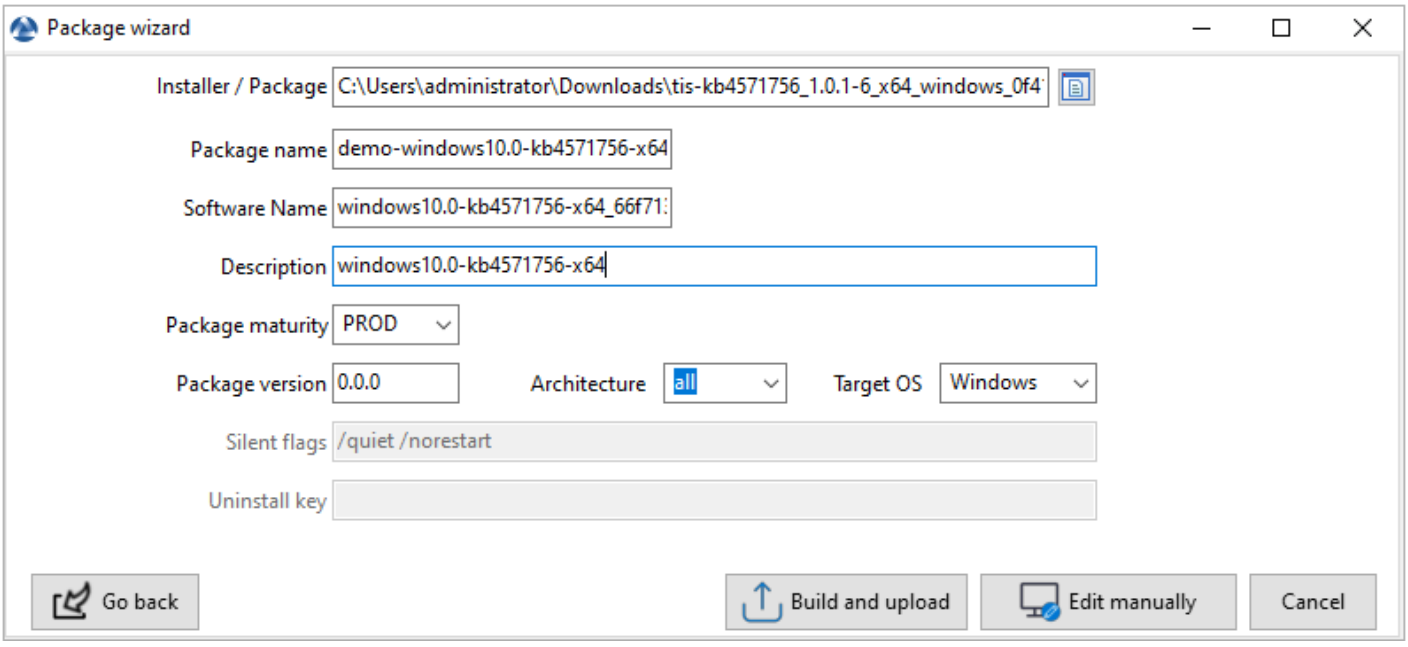


Fig. 5: Informations required for creating the MSU package

34.2.10 Packaging simple Linux packages

Before starting, we assume several conditions:

- You have a graphical interface on your Linux system that you use for developing and testing packages.
- You have installed the **vscode** package from the Tranquil IT repository.
- Your user is named *linuxuser* and is a member of the *sudoers* group.

Creating 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 a sudo.

When you create a template, there will be several files in the `.vscode` folder inside the WAPT 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 WAPT packages are stored in *linuxuser*'s home (home of the currently logged in user).

- VSCode loads up and opens the WAPT package project.

- Check the `control` file content.

  You have to give a `description` to the WAPT package, define the `os_target` and the `version` of the WAPT package.

Hint: `os_target` for unix is *linux*.

Warning: The software `version` number in your `control` file **MUST** start at 0, and not the version number of the software title, as the version number may not be the same as displayed in the DEB / YUM repository.

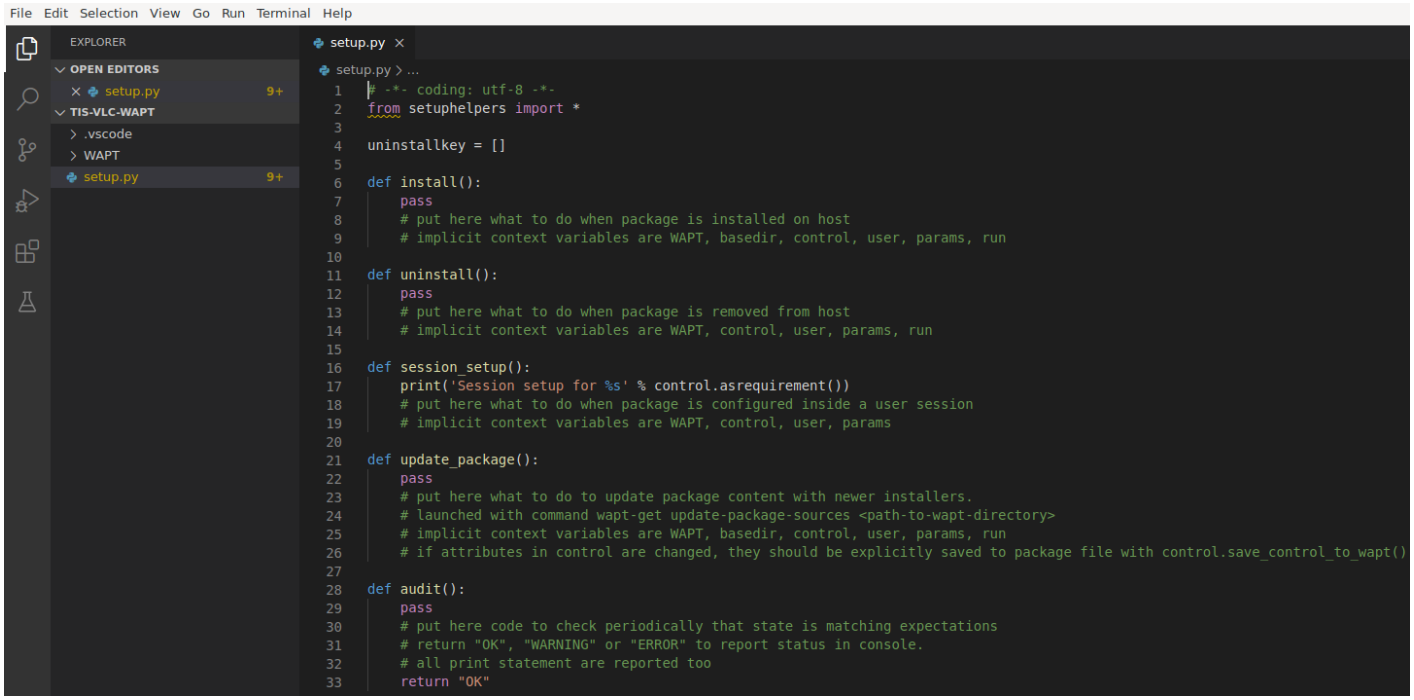


Fig. 6: VSCode opening with focus on the *setup* file

– Original `control` file.

```
package            : tis-vlc
version            : 0-0
architecture       : all
section            : base
priority           : optional
maintainer         : user
description        : automatic package for vlc
```

– 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 the WAPT package.

It allows the WAPT package Developer to release several WAPT package versions of the same software, very useful for very

rapid and iterative development.

- 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*.

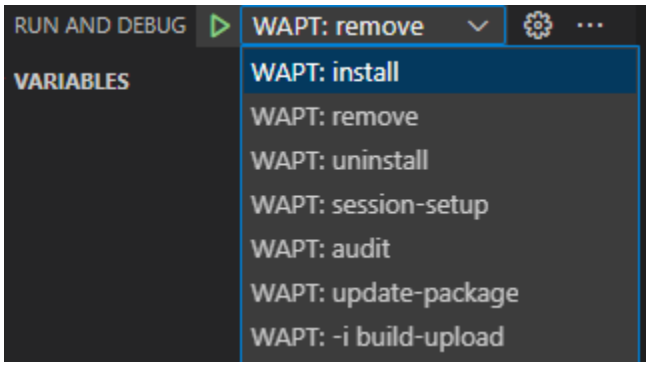


Fig. 7: After uninstallation, the software is correctly removed

- 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 a `session-setup`;

  In this example, we will create a file: file: *vlcrc* by default in the user profile.

```
def session_setup():
  vlcrc_content="""[qt] # Qt interface
qt-notification=0
qt-privacy-ask=0
metadata-network-access=0
"""

  vlcdir = os.path.join(os.environ['HOME'], '.config', 'vlc')
  path_vlrc = makepath(vlcdir,'vlcrc')
  ensure_dir(vlcdir)
  if not isfile(path_vlrc):
      with open(makepath(vlcdir,'vlcrc')) as f:
          f.write(vlcrc_content)
```

- Launch a *session-setup* from VSCode *Run Configurations*.

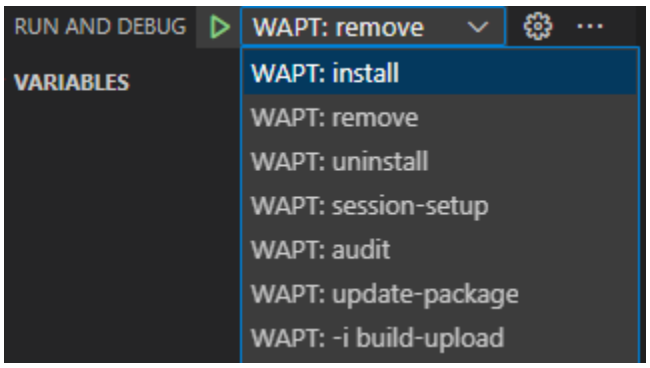


Fig. 8: After uninstallation, the software is correctly removed

Building and uploading the WAPT package

You will find the WAPT package in your `~/waptdev` folder.

You need to transfer the WAPT package folder to the Windows host that has the private key that you use to sign your WAPT packages.

Then, please refer to the *documentation for building and uploading packages from the WAPT Console*.

34.2.11 Encrypting sensitive data contained in a WAPT package

Note: This part of the documentation is for advanced users of WAPT.

This feature is available only in the **Enterprise** version.

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

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 WAPT 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 the *Local Administrators*.

We will therefore encrypt the sensitive data contained inside the package with the certificate belonging to the host.

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 host and then encrypt the sensitive text with it.

The encrypted text for each host is then stored in a `encrypt-txt.json` file at the root of the WAPT package.

When the WAPT package installs, the WAPT Agent will take the encrypted text and decipher it with his own private key.

You can test it by yourself by downloading the example package *tis-encrypt-sample* (https://store.wapt.fr/store/tis-encrypt-sample)

Attention: The python output (log install of the WAPT package) is readable by the users on the host, so **you should not display the deciphered text with a print during installation**.

CHAPTER

THIRTYFIVE

USING DIFFERENT IDES FOR DEVELOPING WAPT PACKAGES

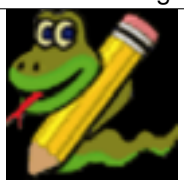
35.1 Configuring WAPT to use supported IDEs

If you are used to work with another *IDE*, you can be relieved now as WAPT supports many popular text editors.

Note: Using a supported IDE will launch the WAPT package project with a valid debug configuration.

35.1.1 On Windows

Table 1: Natively supported text editors in WAPT on Windows

Text editor name	Text editor logo
PyScripter	
Visual Studio Code	
Visual Studio Codium	

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.

```
[global]
...
editor_for_packages = vscode
```

35.1.2 On Linux / macOS

Table 2: Natively supported text editors in WAPT on Windows

Text editor name	Text editor logo
Visual Studio Code	
Visual Studio Codium	
Nano	
Vim	

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**.

```
[global]
...
editor_for_packages = vim
```

35.2 Configuring WAPT to use a custom editor

Windows

```
[global]
...
editor_for_packages = C:\Program Files\Notepad++\notepad++.exe {setup_filename}
```

Linux/ macOS

```
[global]
...
editor_for_packages = /opt/pycharm/bin/pycharm_x64 {wapt_sources_dir}
```

35.2.1 Custom arguments

Table 3: Arguments for the `editor_for_packages` option

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 the custom text editor and opens the WAPT package folder.
`{wapt_base_dir}`	Launches the custom text editor and opens the WAPT install folder.

CHAPTER

THIRTYSIX

PACKAGE STRUCTURE DETAILED

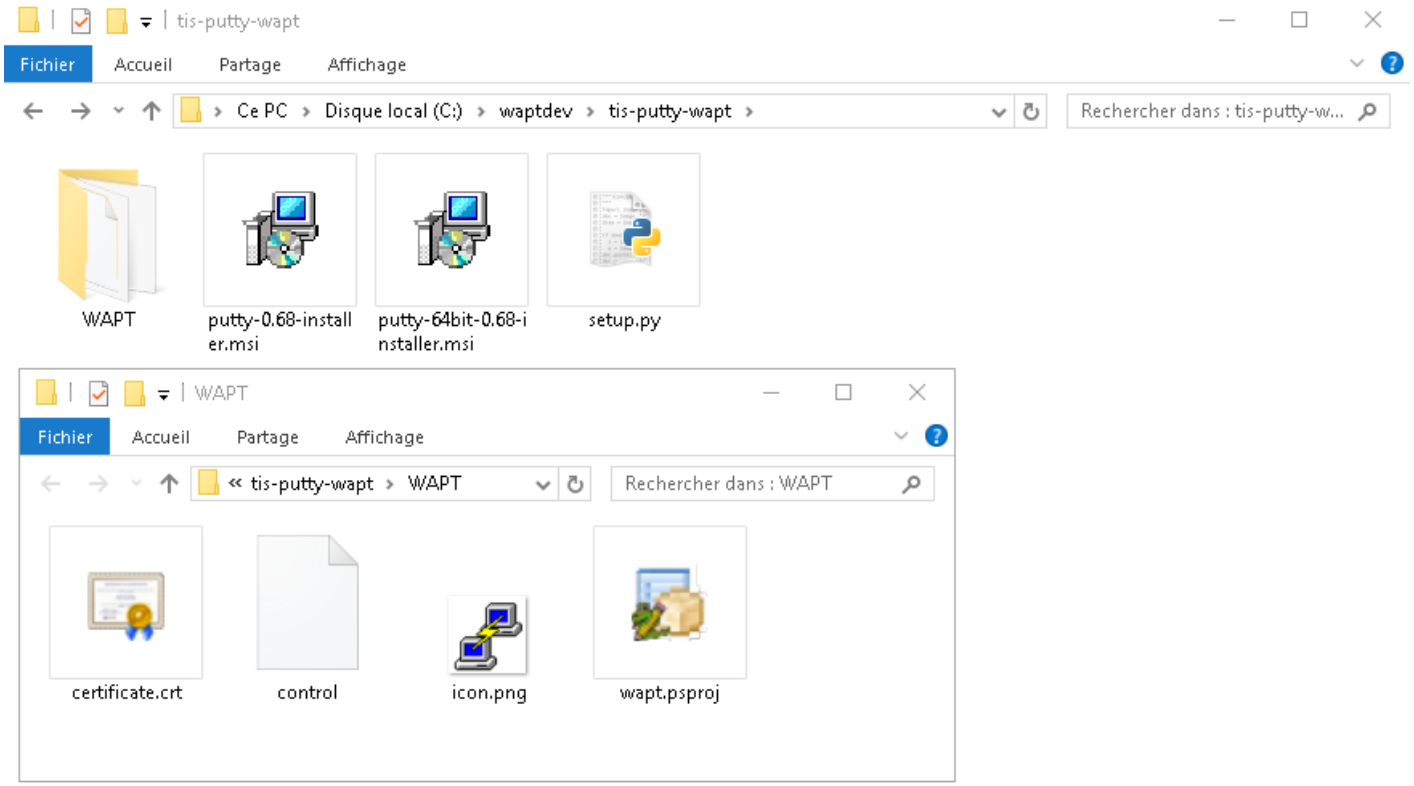


Fig. 1: WAPT package structure shown in Windows Explorer

A WAPT package is a *.zip* file containing several things:

- A `setup.py` file.
- 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 is used to store the **VScode** configuration data for the WAPT package.

36.1 The *control* file

The `control` file is the identity card of a WAPT 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/
package_uuid      : dc66ccd1-d987-482e-b792-04e89a3803f7
valid_from        : 2022-02-23T00:00:00
valid_until       : 2022-03-23T00:00:00
forced_install_on : 2022-03-23T00:00:00
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 1: Description of options of the control file

Settings	Description	Example value
`package`	Defines the package name, **without any accent, nor space, nor any special or uppercase character**.	tis-geogebra
`version`	Defines the package version (note: the version **MUST** not contain more than 5 delimiters, the last number being the version number of the packaging). The version **MUST** start with the packaged software version (**digits only**) split by points (`.`) and **MUST** finish with the WAPT packaging version separated by a dash (`-`) character.	5.0.309.0-1
`architecture`	Defines the processor architecture onto which the WAPT package will install. A x64 package will be invisible to a WAPT Agent installed on a x86 host. Allowed values are: • **x86**: the package is designed for 32bit computers; • **x64**: the package is designed for 64bit computers; • **all**: the package is designed for any architecture.	x64
`section`	Defines the WAPT package type (`host`, `group`, `base`). Allowed values are: • **host**: host package; • **group**: group package; • **base**: software package; • **unit**: OU package;	base
`priority`	Defines the WAPT package install priority (optional). 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.	Not used at this moment
`maintainer`	Defines the author of the WAPT package. To define the WAPT package maintainer's email address may be useful. Use Firstname LASTNAME <email@example.com> format.	Arnold SCHWARZENEGGER <terminator@mydomain.lan>
`description`	Defines the WAPT package description that will appear in the WAPT Console and in the self-service. 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.	The Graphing Calculator for Functions, Geometry, Algebra, Calculus, Statistics and 3D
`description_fr`	Localizes the description of the package.	Calculatrice graphique
`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 (,).	tis-java

continues on next page

Table 1 – continued from previous page

Settings	Description	Example value
conflicts	Defines WAPT packages that **MUST** be **removed** before installing the 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. It works the opposite way of depends. Several conflicts may be defined by splitting them with commas (,).	tis-graph
maturity	Defines the maturity level of the WAPT package (PREPROD, DEV, PROD, etc). By default, WAPT Agents will see packages flagged as *PROD* and packages with an empty maturity. For a computer to see WAPT packages with different maturity levels, the maturities attribute **MUST** be set in the wapt-get-ini configuration file of the WAPT Agent.	PROD
locale	Defines the language environment for the WAPT package. A WAPT Agent will see by default packages that are configured for its language environment(s) 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 of the WAPT Agent. Case and order do not not matter. If you want to respect an order for maturities, you will need to set the order in the wapt-get-ini configuration file of the WAPT Agent.	fr,en,es
target_os	Defines the accepted Operating System for the WAPT 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 2.3 the field target_os can have several data, it has to correspond to the host_capabilities tag. With debian, you can use < or >. The First three in the example are the most used.	windows, mac, linux, debian-bullseye, redhat_based, centos8, debian(>8), ubuntu, almalinux8
min_os_version	Defines the minimum version of Windows for the package to be seen by the WAPT Agent. For a target_os = windows, this field defines the minimal Windows Operating System Version. For example, this attribute may be used to avoid installing WAPT packages on WindowsXP that only work on Windows7 and above. Since version 1.8, it can also define the minimal macOS version. We advise not to use it with Linux since there are several different Linux distributions.	6.0

continues on next page

Table 1 – continued from previous page

Settings	Description	Example value
`max_os_version`	Defines the maximum version of Windows for the package to be seen by the WAPT Agent. For a `target_os` = `windows`, 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 minimal macOS version. We advise not to use it with Linux since there are several different distributions.	10.0
`min_wapt_version`	Defines the minimal version of the WAPT Agent for the WAPT package to work properly. With functionalities in WAPT evolving, some functions that you may have used in old packages may become obsolete with newer versions of WAPT Agents.	1.3.8
`sources`	Defines the path to the SVN location of the WAPT package (**`wapt-get source`**). Defines a repository for versionning WAPT packages, for example https://svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/. This method allows to version a package and collaboratively work on it. Package versionning 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.	https://srv-svn.mydomain.lan/sources/tis-geogebra-wapt/trunk/
`installed_size`	Defines the minimum required free disk space to install the WAPT package. 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. To convert storage values to bytes, visit bit-calculator.	254251008
`impacted_process`	Indicates a list of impacted processes when installing a WAPT package. `impacted_process` 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 WAPT package. This allows to close the application if the application is running before being uninstalled.	firefox.exe
`audit_schedule`	Defines the periodicity of execution of the audit function in the WAPT package. The periodicity may be indicated in two ways: • an integer (in minutes); • an integer followed by a letter (*m* = minutes, *h* = hours, *d* = days, *w* = weeks).	60

continues on next page

Table 1 – continued from previous page

Settings	Description	Example value
editor	Defines the editor of the software title embedded in the WAPT package. The values may be used as filters in the WAPT Console and with the WAPT Self-service.	Mozilla
license	Defines the licence of the software title embedded in the WAPT package. The values may be used as filters in the WAPT Console and with the WAPT Self-service.	GPLV3
keywords	Defines a set of keywords describing the WAPT package. The values may be used as filters in the WAPT Console and with the WAPT Self-service.	Productivity, Text Processor
homepage	Defines the official homepage of the software title embedded in the WAPT package. The values may be used as filters in the WAPT Console and with the WAPT Self-service.	https://www.tranquil.it/
package_uuid	Unique identifier of the package. It is automatically generated when building the package.	dc66ccd1-d987-482e-b792-04e89a3803f7
valid_from	Date / time after which the package may be installed. The WAPT Agent will refuse to install it before that date. The string is formated according to the ISO8601 standard: YYYY-MM-DDTHH:MM:SS. When the date has passed, WAPT will install the package when an update is triggered.	2022-02-23T00:00:00
valid_until	Date / time after which the package may not be installed. The WAPT Agent will refuse to install it after that date. The string is formated according to the ISO8601 standard: YYYY-MM-DDTHH:MM:SS.	2022-02-23T00:00:00
forced_install_on	Date / time after which the WAPT Agent will trigger a forced install of the package. The string is formated according to the ISO8601 standard: YYYY-MM-DDTHH:MM:SS.	2022-02-23T00:00:00
signer	Defines the CN of the WAPT package's signer. It is generally the name of the signer's full name. The value is automatically inserted when signing the WAPT package.	Tranquil IT
signer_fingerprint	Provides the fingerprint of the certificate holder's signature. The value is automatically inserted when signing the WAPT package.	2BAFAF007C174A3B00F12E9CA1E749
signature	Provides the SHA256 hash of the WAPT package. The value is automatically inserted when signing the WAPT package.	MLO-zLiz0qC(...)hsEjGRaMLBlod/p8w==
signature_date	Provides the date when the package was signed. The value is automatically inserted when signing the WAPT package.	20180307-230413

continues on next page

Table 1 – continued from previous page

Settings	Description	Example value
`signed_attributes`	Lists of attributes of the `control` file of the WAPT package that are signed. The value is automatically inserted when signing the WAPT package.	package, version, architecture, section, priority, maintainer, description, depends, conflicts, maturity, locale, min_wapt_version, sources, installed_size, signer, signer_fingerprint, signature_date, signed_attributes

Attention: If the `control` file contains special characters, the `control` file **MUST** be saved in **UTF-8 (No BOM)** format.

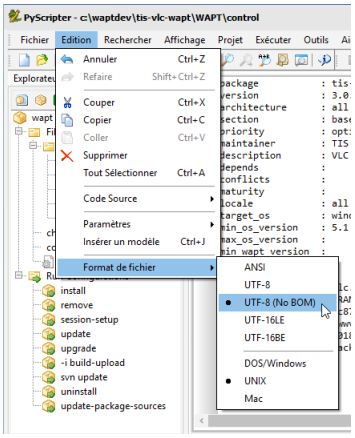


Fig. 2: PyScripter - UTF-8 (No BOM)

36.2 The *setup.py* file

- `import setuphelpers` is found at the beginning of every WAPT package that embeds a `setup.py`:

```
from setuphelpers import *
```

The WAPT package imports all *SetupHelpers* functions.

SetupHelpers is a WAPT library that offers many methods to easily develop highly functional WAPT packages.

- followed by a `uninstallkey` list to associate a list of *uninstall keys* to the WAPT package.

```
uninstallkey = ['tisnaps2','Mozilla Firefox 45.6.0 ESR (x86 fr)']
```

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 = []
```

- followed by functions such as `def_install()`, `def_uninstall()`, `def_session-setup()` and `def_audit()`

  These functions describe the recipes of the WAPT package, the set of instructions that will be executed to install, remove, configure and audit a WAPT package.

36.3 The *wapt.psproj* file

Package project file `wapt.psproj` is located in the WAPT folder.

It is the **PyScripter** project file for the WAPT package.

To edit a package with **PyScripter**, just open the file.

36.4 The *icon.png* file

The `icon.png` icon file is located in the WAPT folder.

It associates an icon to the WAPT package.

Hint:

- The icon is used in the Self-Service, it is downloaded with its MD5 sum for security; if the MD5 sum is not good then the icon is removed.
- The icon **MUST** be a 48px per 48px *.png* file.

36.5 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.

36.6 The *signature* file

The `signature` file is located in the WAPT folder.

It contains the signature of the `manifest.sha256` file.

On installing a WAPT package, **wapt-get** checks:

- That the signature of `manifest.sha256` matches the actual `manifest.sha256` file (the WAPT Agent will verify the public certificates in `C:\Program Files (x86)\wapt\ssl` on Windows and `/opt/wapt/ssl` on Linux and macOS).
- That the sha256 fingerprint of each file is identical to the fingerprint in the `manifest.sha256` file.

36.7 The *certificate.crt* file

The `certificate.crt` file is located in the WAPT folder.

It is the maintainer's certificate whom signed the package.

On installing a WAPT package, **`wapt-get`** checks that the `certificate.crt` or its parent matches with certificates in `C:\Program Files (x86)\wapt\ssl` on Windows and `/opt/wapt/ssl` on Linux and macOS. If the certificate does not match, the WAPT package will not be installed.

36.8 Other files

Other files may be embedded in the WAPT package, for example:

- An installer beside the `setup.py` to be called from the **`setup.py`**.
- An answer file to pass on to the software installer.
- A license file.
- Etc.

CHAPTER

THIRTYSEVEN

WAPT SETUPHELPERS APIDOC

37.1 Setuphelpers for Windows

Link for Windows

37.2 Setuphelpers for Linux

Link for Linux

37.3 Setuphelpers for MacOS

Link for MacOS

CHAPTER

THIRTYEIGHT

FREQUENT PROBLEMS AND QUESTIONS

38.1 Updating WAPT packages from Python 2 to Python 3

Attention: With WAPT 2.0, the WAPT internals have switched to python3. WAPT packages **MUST** follow the new python3 syntax.

Table 1: The principal syntax differences

Syntax	Python 2	Python 3
print	`print'Hello'`	`print('Hello')`
unicode string	`ur`	`r`
operators	`<> <=> !=`	`!=`
Windows registry access	`_winreg`	`winreg`

Hint: For more details, visit:

- https://python-future.org/compatible_idioms.html.
- https://blog.couchbase.com/tips-and-tricks-for-upgrading-from-python-2-to-python-3/.

38.2 Resetting the WAPT Linux Server 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:

- Connect to the WAPT Server with SSH.
- Connect with user root (or use sudo).
- Launch the post-configuration script.

38.3 I lost my WAPT private key

WAPT 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.

38.3.1 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, manually, using a GPO or using an Ansible role (not documented), the new certificate `.crt` on your clients in the **ssl** folder.
  - `C:\Program Files (x86)\ssl` on Windows;
  - `/opt/wapt/ssl` on Linux and macOS.

38.3.2 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:

- *Using the WAPT Console*, or
- *Using the command line*.

38.4 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 **ssl** folder:

- `C:\Program Files (x86)\ssl` on Windows;
- `/opt/wapt/ssl` on Linux and macOS.

That operation can be done using a GPO, manually, with a WAPT package or with an Ansible role (not documented).

Finally, you will have to follow the same steps as for *the loss of your private key*.

38.5 My BIOS UUID bugs

- Some problems happen sometimes with some BIOSes. WAPT uses the *UUID* of the host 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 host will register in the WAPT Console but it will replace an existing device, considering that the host has only changed its name.

38.5.1 Solving the BIOS UUID issue

WAPT allows to generate a random *UUID* to replace the one retrieved from the BIOS.

```
wapt-get generate-uuid
```

The WAPT Agent FQDN may be used instead of the UUID. In the `wapt-get.ini` configuration file, define in the *[global]* section:

```
use_fqdn_as_uuid = True
```

38.6 The WAPT Deploy utility does not work

The the WAPT Deployment utility does not succeed in installing the WAPT agent.

38.6.1 Launching the WAPT Deployment utility locally

Launching the WAPT Deployment utility locally can be a good method for showing errors explicitly.

> **Attention:** You **MUST** launch the command prompt using a *Local Administrator* account.

Example of command to launch:

```
C:\Program Files (x86)\wapt\waptdeploy.exe --
↪hash=2a9971aad083d6822b6e4d1ccfb9886be9429ec58bb13246810ff3d6a56ce887 --minversion=2.1.0.10550 --
↪wait=15 --waptsetupurl=https://srvwapt.mydomain.lan/wapt/waptagent.exe
```

In our case the hash for the WAPT Deploy utility is not correct.

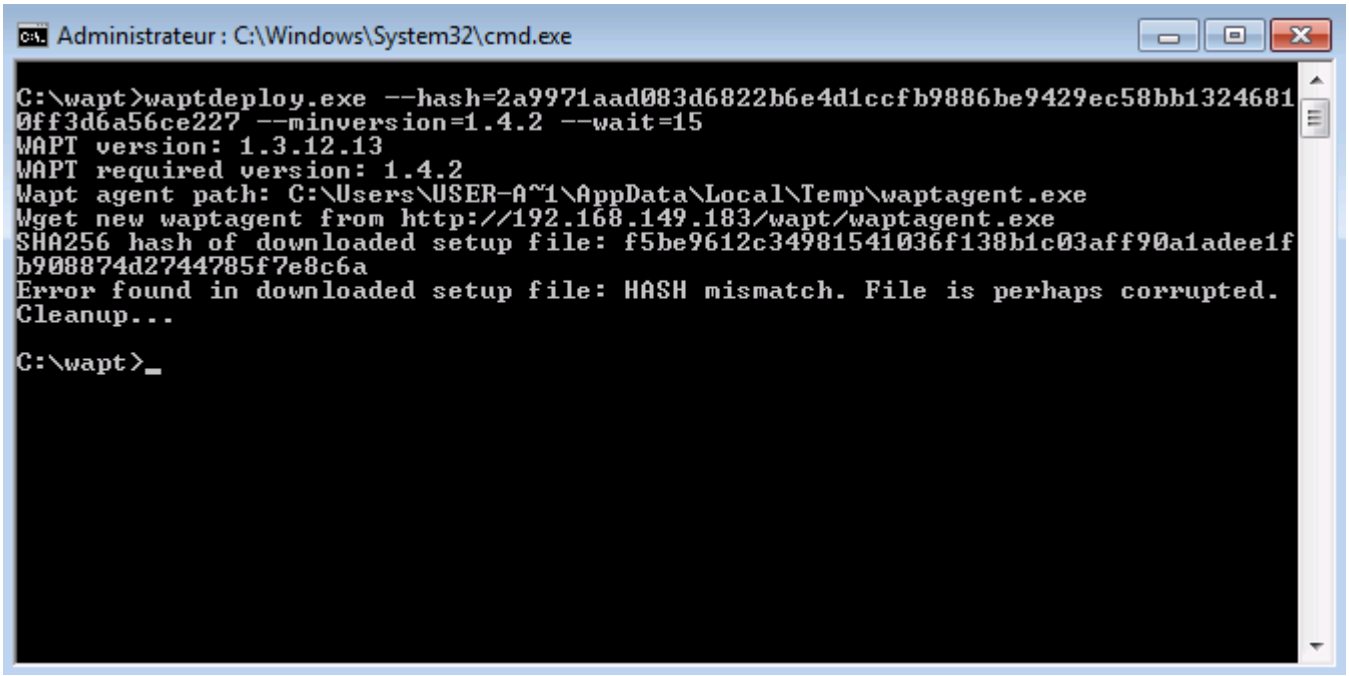


Fig. 1: Error with the WAPT Deploy utility hash in a text terminal window

The WAPT Deploy utility works manually but does not work with GPO

Check that port 8088 is listening correctly on the host:

```
gpresult /h gpo.html & gpo.html
```

To force the application of the GPO:

```
gpupdate /force
```

If the WAPT Deployment utility does not show up you will have to double check the GPO settings:

- You may be using an old WAPT Deployment utility version, then `download the latest version` of the WAPT Deployment utility from the WAPT Server web page.
- Thanks to Emmanuel EUGENE from French public research institution INSERM who submitted this possible cause for the WAPT Deployment utility 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`.

38.7 Windows does not wait for the network to be running on startup

By default Windows does not wait for the network to be up at computer startup.

This can cause problems during the WAPT Deployment utility execution because the WAPT Deployment utility requires network connectivity to retrieve the new WAPT Agent.

There are **2** solutions:

1. We recommend adding `waptdeploy.exe` to the startup and shutdown scripts *on the GPO*.
2. You can enable the GPO: **Always wait for the network at computer startup and logon** with *Computer Configuration → Administrative Templates → System → Logon → Always wait for the network at computer startup and logon*

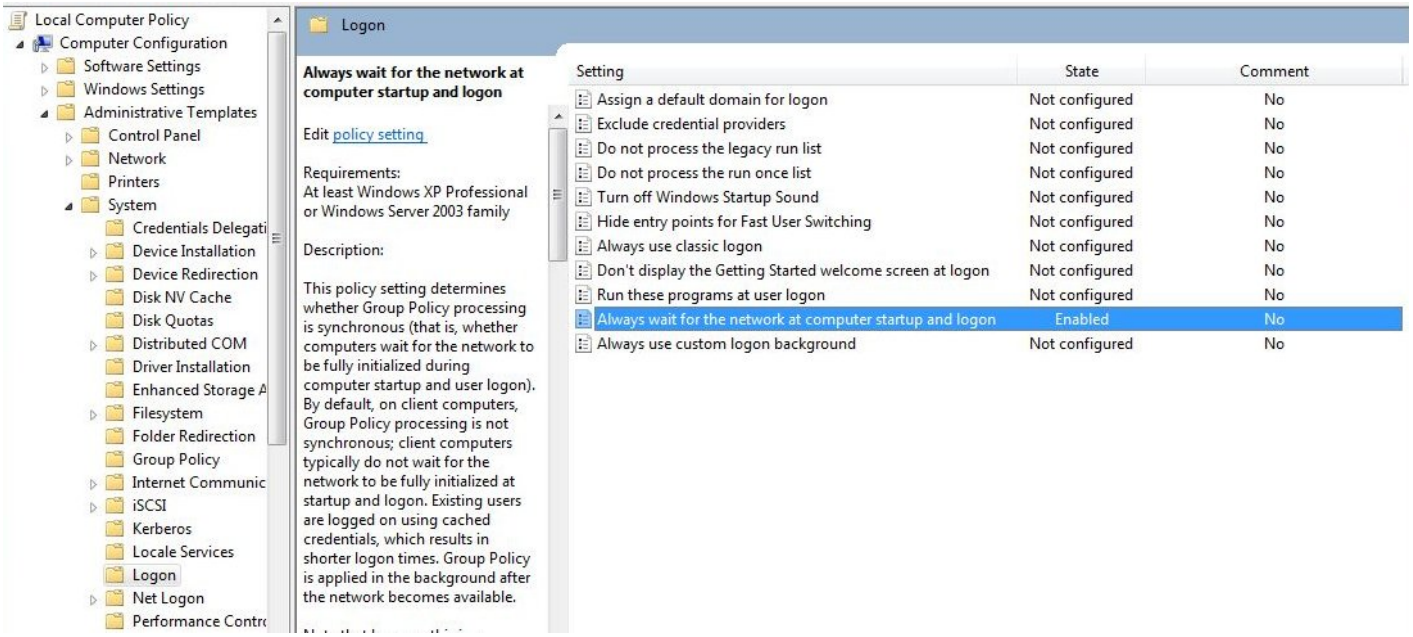


Fig. 2: GPO to wait network startup

38.8 The WAPT Exit utility will not launch

Despite the script actually being registered in the local security shutdown strategy, the **waptexit** script does not launch at computer shutdown.

38.8.1 Hybrid shutdown

Windows 10 hybrid shutdown **MUST** be disabled because it causes many problems and strange behaviors, disabling Hybrid Shutdown will restore the WAPT Exit script execution at shutdown.

Hybrid shutdown can be disabled by setting a value in `wapt-get.ini` file *of the WAPT Agent*.

It is possible to set this value when *creating the WAPT Agent*.

A WAPT package exists to solve the Hybrid Shutdown problem: tis-disable-hybrid-shutdown.

38.8.2 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.

The workaround consists in using a scheduled task that will launch `C:\Program Files (x86)\wapt\wapt-get.exe` with the argument `upgrade`.

38.8.3 Corrupted local GPO

It sometimes happens that local security policies on a computer are corrupted.

One of the possible solutions is to:

- Remove local security strategies by deleting the file `C:\Windows\System32\GroupPolicy\gpt.ini`;
- Restart the computer;
- Re-install the shutdown scheduled tasks with:

```
wapt-get add-upgrade-shutdown
```

If the problem occurs again, this may mean that another application also manipulates the local GPO.

38.9 The WAPT Exit utility halts after 15 minutes and does not finish the installing the WAPT packages

By default, Windows shutdown scripts are only allowed to run for 15 minutes.

If a script has not finished before that limit, Windows will interrupt the script.

To solve that problem, increase the `pre_shutdown_timeout` value and the `max_gpo_script_wait` value in the `wapt-get.ini` file of the WAPT Agent.

Define *these values* to change the default behavior.

```
max_gpo_script_wait = 360
pre_shutdown_timeout = 360
```

The WAPT package tis-wapt-conf-policy sets this configuration.

The other solution may be to use the GPO `File.ini`.

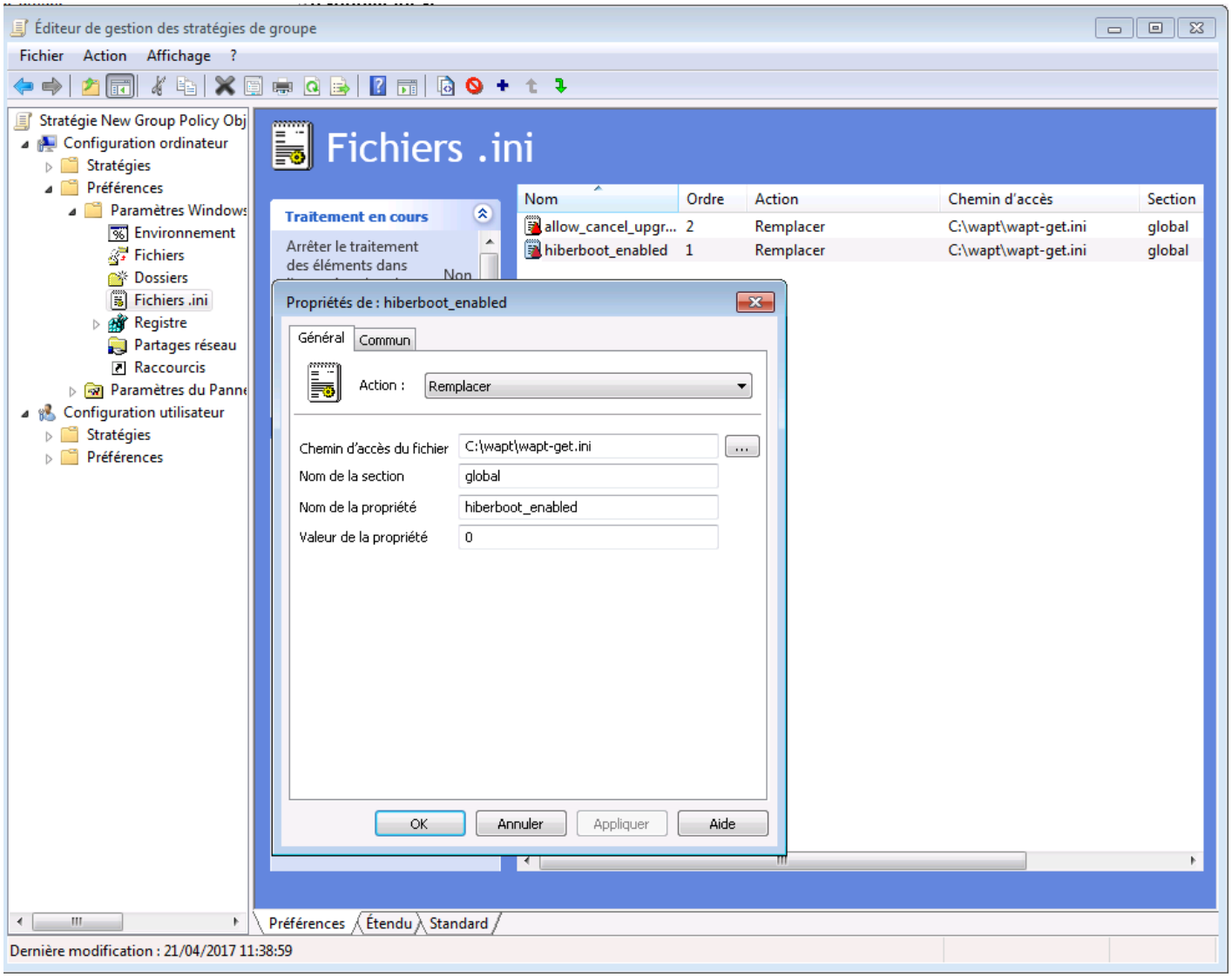
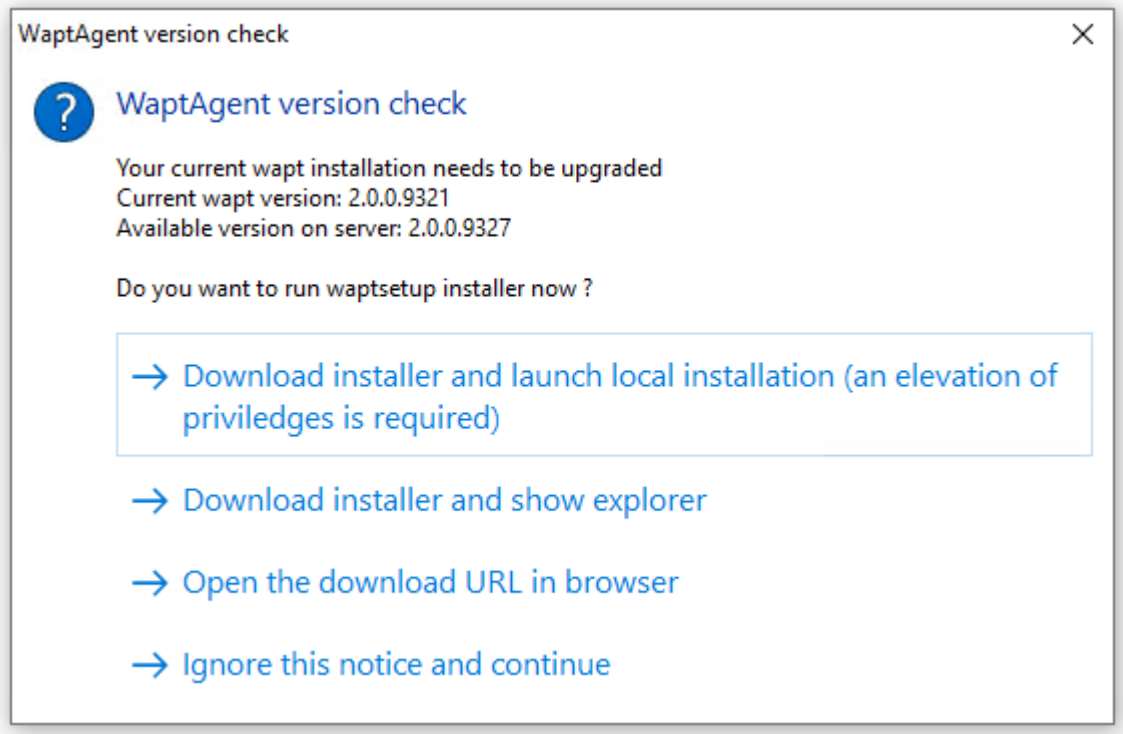


Fig. 3: Using a GPO ini File to configured the script execution delay

38.10 Error message when opening the WAPT Console

38.10.1 Version check



The WAPT Console version is not the same as the version of the WAPT Server. Upgrading the WAPT Console to the same version of the WAPT Server is the recommanded course of action.

38.10.2 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.

```
systemctl status nginx
```

- If **Nginx** is not running, restart the **Nginx** service.

```
systemctl restart nginx
```

- If **Nginx** still does not start, you will need to analyze the journal logs in:
  - `/var/log/nginx/` on Linux;
  - `C:\Program Files (x86)\wapt\waptserver\nginx\logs` on Windows.

38.10.3 Service unavailable

It is possible that the WAPT Server service is stopped:

- Check whether **waptserver** is running.

```
systemctl status waptserver
```

- If the command returns an error, then start the **waptserver**.

```
systemctl start waptserver
```

38.10.4 Error connecting with SSL ... verify failed

The WAPT Console seems not to be able to verify the WAPT 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 rebuilt 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 WAPT 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.

38.10.5 I can't do anything in the WAPT Console, everything is greyed

The WAPT Console seems locked, you can not execute any action, everything is greyed out.

If you are connected with another user than the *Superadmin*, your ACL rules may not be set properly.

To fix this, close the WAPT Console and open it with the *Superadmin* account. Then, go to *Tools* → *Manage Wapt users and rights*. Here, you will see the user in the list, give the user the appropriate permissions, then save and close the WAPT Console. Re-open the WAPT Console using your login.

38.11 Error message about package on the WAPT Console

38.11.1 Error when uploading package

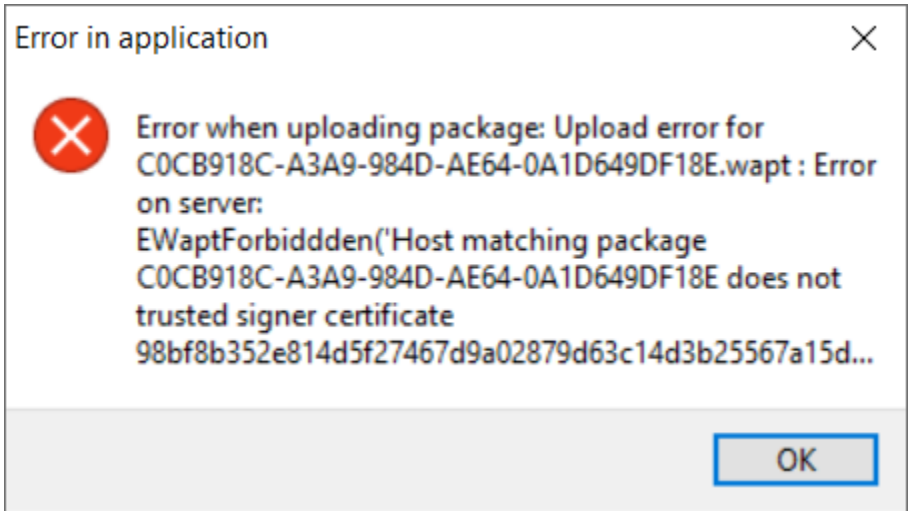


Fig. 4: Window showing that the uploaded package has signer certificate issue

The WAPT Console shows this error : `Error when uploading package : EWaptForbidden('Host matching package UUID_HOST does not trusted signer certificate)`.

You have this error when you try to upload a WAPT package but the used certificate which signed package is not present in your computer's ssl WAPT install location folder. Be reminded, if you have a WAPT Windows Server to **not lauch the WAPT Console on the server**. Add the WAPT certificate which signed the package in your computer's ssl WAPT install location folder then retry.

38.11.2 Error locale

Fig. 5: Window showing that the WAPT Console does not find a package

The WAPT Console shows this error in two situations.

Package does not exist in the repository anymore, yet a host need it

There are two possible solutions:

- Try to get package anew from Tranquil IT's store.
- Delete the package from the host dependencies.

When you try to install a package with a locale that is unknown to the host

There are two possible solutions:

- Download the WAPT package having the matching locale from Tranquil IT's store.
- Edit your WAPT package and set in the `control` file the option `locale` with the correct locale (`locale=en,fr`).

38.12 Problems with registering a host with WAPT

If you do a **`wapt-get register`** and it returns:

```
FATAL ERROR : ConnectionError: HTTPSConnectionPool(host='XXX.XXX.XXX.XXX', port=443): Max retries␣
↪exceeded with url: /add_host
```

You need to check that the 443 port is correctly forwarded to the WAPT Server and not blocked by a firewall.

38.13 Problems when enabling enable-check-certificate

38.13.1 Message "Certificate CN ### sent by the WAPT 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 the `wapt-get.ini` file of the WAPT Agent.

- There are **2** solutions:

1. Check the value of `wapt_server` in the `wapt-get.ini` file of the WAPT Agent.

   If the value is correct, this surely means that an error has happened during the generation of the self-signed certificate during the WAPT Server post-configuration (typing error, ...).

   You **MUST** then regenerate your self-signed certificates.

2. On the WAPT Server, delete the content of the `/opt/wapt/waptserver/apache/ssl/` folder.

   Then, relaunch the post-configuration script (the same as the one used during initial installation, with the same arguments and values).

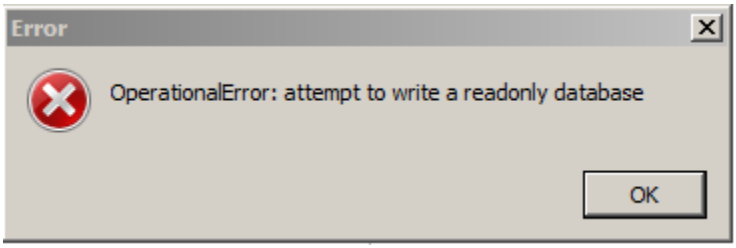
   Then, be sure that the value of *FQDN for the WAPT Server* is correct.

- You may now retry **`enable-check-certificate`**.

38.14 Problems when creating a WAPT package

38.14.1 Problems with access rights and PyScripter

When trying to install a package from **PyScripter**, if the following message appears:



Launch **PyScripter** using a *Local Administrator* account and redo the desired action.

38.14.2 The 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 the WAPT package locally then to upload it with **WinSCP** or an equivalent utility.

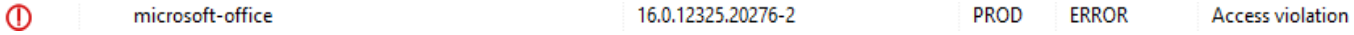
- Build the WAPT package with **PyScripter** or manually *build the package*.

Hint: The WAPT package in `C:\waptdev`.

- Download and install **WinSCP**
- Using **WinSCP**, upload the WAPT package in the correct *repository location* according to the version of the WAPT repository.
- Once the upload has finished, recreate the `Packages` index file on the WAPT repository using the following command and remplacing **repository** by the *repository location* according to the version of the WAPT repository.

```
wapt-scanpackages repository
```

38.14.3 Access violation error while re-signing a WAPT package



If the **Access violation** error appear, it may mean that the WAPT package is too big.

Manually edit the package and visit *this procedure for signing large WAPT packages*.

38.14.4 WAPT package in error

Problem installing a WAPT package

I have a WAPT 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 WAPT 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'"

Some packages return the following error in the WAPT Console:

```
"Error timed out after seconds with output '600.0'"
```

Explanation

By default, when installing a WAPT package embedding a **run** and a **install_msi_if_needed** command, 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

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"

Some WAPT 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 host.
- 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 host.

The function returns the WAPT package in error. The installation will be retried at every *upgrade* cycle until the WAPT 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"

Some WAPT packages return the following error in the WAPT Console:

```
XXX 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.

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 the 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*.

There are **2** solutions:

- Use the argument `get_version` to provide the path to another `uninstallkey`.

```
def install():

   def versnaps2(key):
       return key['name'].replace('NAPS2 ','')

   install_exe_if_needed('naps2-5.3.3-setup.exe',silentflags='/VERYSILENT',key='NAPS2 (Not Another␣
↪PDF Scanner 2)_is1',get_version=versnaps2)
```

- 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!**

38.15 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 (Anti-Virus) 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"
```

38.16 I have an issue with my proxy - THttpClientSocket.SockRecv(1) read = 0

If you have this issue:

The error comes from a `timeout` option in the `waptconsole.ini`.

Indeed, since WAPT 2.1 version, timeout is defined in milliseconds and not in seconds like before.

You will need remove the `timeout` option in the `waptconsole.ini` file located in `%localappdata%\waptconsole`.

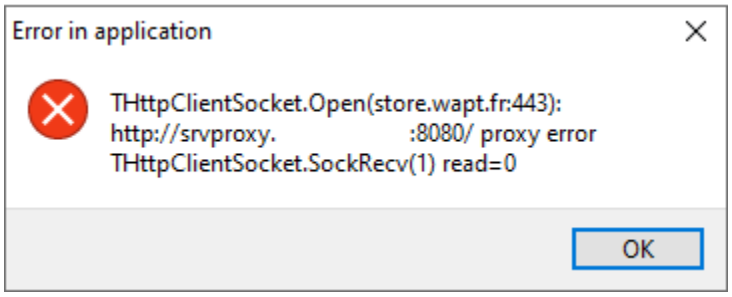


Fig. 6: Window showing a proxy timeout error in the WAPT Console

38.17 Common mistakes

38.17.1 How to move my repository to another partition

For any reason, you may need move the repository to another partition.

Your repository contains **3** folders which can be quite large:

- `wapt`;
- `wapt-host`;
- `waptwua`.

Linux

On Linux, create a mount point on `fstab`.

For this example, the second partition is named *part2*.

part2 is an **ext4 formated partition**.

Debian / Ubuntu

- Create a temporary folder.

```
mkdir /mnt/tmp
```

- Create a temporary mount point.

```
mount /dev/part2 /mnt/tmp
```

- Move the folders.

```
mv /var/www /mnt/tmp
```

- Unmont the partition.

```
umount /dev/part2
```

- Edit the `fstab` file.

```
vi /etc/fstab
```

- Add the following line to the `fstab` file.

```
# <file system> <mount point> <type>  <options> <dump>  <pass>
/dev/part2      /var/www      ext4    defaults  0       0
```

- Mount the partition.

```
mount -a
```

Hint: If there is no error, the partition is mounted.

- You can check by running.

```
df -h

#Result
Filesystem     1K-blocks     Used Available Use% Mounted on
dev/part2       15G          944M       14G   7% /var/www
```

- Remove the temporary folder.

```
rm -rf mnt/tmp
```

RedHat and derivatives

- Create a temporary folder for copying the folders.

```
mkdir /mnt/tmp
```

- Create a temporary mount point.

```
mount /dev/part2 /mnt/tmp
```

- Move the folders.

```
mv /var/www/html /mnt/tmp
```

- Unmont the partition.

```
umount /dev/part2
```

- Edit the `fstab` file.

```
vi /etc/fstab
```

- Add the following line to the `fstab` file.

```
# <file system> <mount point> <type>  <options> <dump>  <pass>
/dev/part2      /var/www/html      ext4    defaults  0       0
```

- Mount the partition.

```
mount -a
```

Hint: If there is no error, the partition is mounted.

- You can check by running.

```
df -h

#Result
Filesystem     1K-blocks     Used Available Use% Mounted on
dev/part2       15G          944M       14G   7% /var/www
```

- Remove the temporary folder.

```
rm -rf mnt/tmp
```

Windows

On Windows, the best method is to *backup* and *restore* the WAPT Server on the new partition.

Note: It is possible to install the WAPT Server on another partition than `C:`.

38.17.2 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**.

38.17.3 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 WAPT Server's performance because it forces the WAPT Server to parse a relatively large JSON (Java Script Object Notation) BLOB (Binary Large OBject) and to inject the result into the PostgreSQL database.

The function is by default triggered manually or when a new WAPT package upgrade is applied.

When you use the `register()` function in a WAPT audit script, it will run every time the audit script is triggered so it will load the WAPT Server with no apparent benefit.

Therefore, **we do not recommend the use of the :code:`register()` function in audit scripts**.

38.17.4 EWaptBadControl: 'utf8' codec can not 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)*.

38.17.5 I have a lot more hosts in the WAPT 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 WAPT Console than *host packages* on you WAPT sSrver. **This is normal**.

In fact, WAPT *unit* 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 WAPT Server if only *unit packages* are used for managing a fleet of devices.

CHAPTER

THIRTYNINE

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
- **Forum in French**: https://forum.tranquil.it/
- **Forum in English**: https://www.reddit.com/r/WAPT
- **Discord**: https://discord.gg/hFdrqs2C5g

CHAPTER

FORTY

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 **Discovery** version of WAPT, he is the unique *Administrator* of WAPT.

User
Users
A **User** is an individual who uses a host that is equipped with a WAPT Agent (WAPT **Enterprise** and **Discovery**).

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 with securing their IT systems.

Website : https://www.ssi.gouv.fr/

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.organization.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. Its 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.

waptagent
: **waptagent** is the WAPT resident program installed on each computer.

waptexit
: **waptexit** is a WAPT command launched by Windows shutdown script to update packages that have a *PENDING* status (on Windows Professional versions).

waptsetup
: **waptsetup** is a setup script to install the WAPT Console.

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
CNAME fields
: A **CNAME** DNS field is an alias name for another A *DNS* field.

A field
A fields
: A **DNS A field** matches a name (generally the name of a host) with an IP address.

Certificate Authority
An CA 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.

CHAPTER

FORTYONE

PRESENTATION OF THE SECURITY PRINCIPLES IN WAPT

Date	Sep 20, 2024
Written by	Hubert TOUVET, Vincent Cardon
Applicable for WAPT	>= 2.3.0.13180
Version of the Document	2.3.0.0-0
Git hash	2e8409fd6f4e09ae76569c47197267f2c1d67d29

  - *Hypothesis H2: the operating systems underlying WAPT are sane*
  - *Hypothesis H3: the binaries necessary for WAPT to operate are sane*
  - *Hypothesis H4: the WAPT packages are built in a safe manner*
  - *Hypothesis H5: the Users of the end-devices are not Local Administrators*
  - *Hypothesis H6: the Local Administrators are trained*
- *Description of threats on WAPT's sensitive assets*
  - *Threat T1: installation of an unsafe software by an unauthorized entity*
  - *Threat T2: modification of configuration values by an unauthorized entity*
  - *Threat T3: illegitimate access by an unauthorized entity*
  - *Threat T4: network listening by an unauthorized entity*
  - *Threat T5: modification of network traffic by an unauthorized entity (type* Man In The Middle*)*
- *Description of WAPT's security functions*
  - *Security function F1: access authentication*
    - *Security function F1A: authentication of a device on initial registration in the WAPT database*
      - *Solution implemented*
    - *Security function F1B: verification of WAPT Server HTTPS certificates by the WAPT Agents*
      - *Solution implemented*
    - *Security function F1C: no listening port on the WAPT Agents*
      - *Solution implemented*
    - *Security function F1D: signature of inventory return states*
      - *Solution implemented*
    - *Security function F1E: verification of authorizations before launching of WAPT commands*
      - *Solution implemented*
  - *Security function F2: protecting the integrity of the installation process of WAPT packages*
    - *Security function F2A: signature of WAPT packages*
      - *Solution implemented*
    - *Security function F2B: signature of the attributes in the control files*
      - *Solution implemented*
    - *Security function F2C: access restriction to the installation folder of the WAPT Agent*
    - *Security function F2D: total access restriction to the folder storing the key / certificate for inventory signing*
  - *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*
      - *Solution implemented*
- *Presentation of server authentication processes*

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.

41.1 Preamble and definitions

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

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

41.2 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 WAPT 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 `setup.py` 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 Authorities 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.

41.3 Description of typical user roles in WAPT

The following roles **MUST** be understood to evaluate the security principles incorporated into WAPT:

- **User**

  A *User* is an individual/ user of a WAPT equipped end-device (**Enterprise** and **Discovery**).

- **Package Deployer**

  A *Package Deployer* is an individual with the ability to sign packages that **DO NOT** contain python code (generally *group*, *host* and *unit* packages) and with the ability to upload the package to the main repository (**Enterprise**).

- **Package Developer**

  A *Package Developer* is an 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**);

Note: The distinction between *Package Deployer* and *Package Developer* only exists in the **Enterprise** version of WAPT.

- **SuperAdmin**

  The *SuperAdmin* is an individual with all rights within WAPT (**Enterprise** and **Discovery**).

- **Local Administrator**

  A *Local Administrator* is an individual with local administration right of the WAPT equipped end-devices (**Enterprise** and **Discovery**).

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

Note: The *Users* that are members of the Active Directory security group **waptselfservice** have access to all the packages of the wapt repository without any filtering.

41.4 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:

41.4.1 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: Security needs for the communications:

- integrity;
- confidentiality;
- authenticity.

41.4.2 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 needs for the inventory data:

- integrity;
- confidentiality.

41.4.3 Sensitive asset A3: log journals

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

Note: Security needs for historical logs:

- availability.

41.4.4 Sensitive asset A4: configuration values

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

Note: Security needs for configuration values:

- integrity;
- confidentiality.

41.4.5 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 for configuration values:

- integrity.

41.4.6 Sensitive asset A6: authentication

Authentication to the WAPT Management 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 for the authentication

- integrity;
- confidentiality.

41.5 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:

41.5.1 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.

41.5.2 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.

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

All libraries and tools necessary to install WAPT are considered to be sane.

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

The *Administrator* and *Package Developer* are 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.

Administrators are also responsible for writing and incorporating safe `setup.py` scripts into the WAPT packages.

41.5.5 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.

41.5.6 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.

41.6 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 an entity that interacts with WAPT without legitimately having access to it.

Note: *Administrators*, *Local Administrator*, *Package Deployers*, *Package Developers* are not considered to be attackers.

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

41.6.1 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.

41.6.2 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*.

41.6.3 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 WAPT Server. It also corresponds to an attacker being able to impersonate a WAPT Agent.

41.6.4 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 WAPT Server hosting WAPT.

41.6.5 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 WAPT Agents and the WAPT Server hosting WAPT, or between the WAPT Console and the WAPT Server.

41.7 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.

41.7.1 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: The risks avoided are:

- 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 WAPT 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 host account that is correctly registered in the *Organization*'s Active Directory domain.

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

When registering, the device generates a RSA keypair in local private directory, and sends to the server a certificate signing request along with the kerberos ticket.

If the Kerberos ticket is valid, the WAPT Server registers the client in the database, signs the CSR, stores the certificate in its database and returns it to the client.

In case a device is already registered, it can re-register itself without a kerberos ticket using its current server's signed certificate.

If the kerberos authentication fails or is not available, the server sends an unauthorized status, and the client can ask the Local Administrator to enter credentials of a WAPT Server account having either *admin* privilege or *register_host* priviledge.

The authentication of the account is performed server side using either *admin*, *passwd*, or *ldap* mechanisms.

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 WAPT Server HTTPS certificates by the WAPT Agents

New in version 1.5.

Note: The risks avoided (notably MITM) are:

- 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 WAPT 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 WAPT Server HTTPS certificate is introduced in `C:\Program Files (x86)\waptwapt-get.ini` on the WAPT Agents that will **force the verification of the WAPT Server certificate by the WAPT Agents**.
- An option for verifying the WAPT Server HTTPS certificate is introduced in `C:\Program Files (x86)\waptwapt-get.ini` on the WAPT Agents that will force the verification of the WAPT 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: The risks avoided are:

- 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 WAPT Server.

Security function F1D: signature of inventory return states

New in version 1.3.12.13.

Note: The risks avoided are:

- 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 deciphered with the public key stored in the database.
- The WAPT Server will refuse any inventory that is signed with a wrong key.

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

Note: The risks avoided are:

- 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 WAPT 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.

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

Security function F2A: signature of WAPT packages

Note: The risks avoided are:

- 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 WAPT 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 the *`.crt`* file 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 *`.crt`* file) 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 WAPT Agent.

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

New in version 1.4.

Note: The risks avoided are:

- 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 WAPT 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 WAPT package.

The WAPT Server extracts from each WAPT package the certificate of the signer and adds it to 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: The risks avoided are:

- 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.

The access restrictions to the `wapt` folder rely on the standard ACLs mechanism of the Operating system and are enforced during installation of the WAPT Agent.

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

Note: The risks avoided are:

- 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.

41.7.3 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: The risks avoided are:

- An unauthorized entity sending falsified requests to the WAPT Agents.

Solution implemented

The following commands amongst others are signed by the WAPT Console before being sent to the targeted WAPT Agents via the WAPT server and Websockets:

- **`trigger package install`**: requests the WAPT Agent to install a WAPT package that is marked as **MISSING**.
- **`trigger package remove`**: requests the WAPT Agent to remove a WAPT package.
- **`trigger package forget`**: requests the WAPT Agent to forget the existence of a previously installed WAPT package without removing the software or the configuration.
- **`trigger host update-status`**: requests the WAPT Agent to send its current inventory status to the WAPT Server.
- **`trigger host upgrade`**: requests the WAPT Agent to execute a package flagged as **NEED UPGRADE**.
- **`trigger host update`**: requests the WAPT Agent to update the list of available packages, and check whether the client should install, upgrade or remove WAPT packages, depending on the WAPT packages dependencies tree.

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

- the device's *UUID*;
- the action (ex: **`wapt-get 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 WAPT Agent will the verify that the timestamp is within a one minute delay window.
- Ultimately, the WAPT Agent will verify that the certificate is authorized to launch actions.

41.8 Presentation of server authentication processes

Kerberos authentication is:

- *admin*: the user name and pbkdf2-sha256 hash derivation of the password are stored in the server's configuration. This is the primary WAPT Administrator authentication mechanism.
- *passwd*: additional secondary user names and passwords hash are stored in `htpasswd` style file server side.
- *ldap*: if the passwd hash is not defined in `htpasswd` file, a ldap server can de declared and used as authentication mechanism for wapt server secondary accounts. Valid ldap accounts are defined by a ldap base DN and a ldap list of groups.
- *session*: when initially authenticated using one of *admin*, *passwd* or *ldap* mechanism, a session cookie can be used as a subsitute for subsequent server authentication. The session cookie has a default lifetime of 12h.

41.9 Presentation of server authorization processes

- After proper authentication, the server stores the associated priviledges in client sessions.
- In the context of the current TOE, only the *admin* priviledge is evaluated.
- *admin* priviledge is an alias for all priviledges. Once authenticated, the User gains access to all the available endpoints of the WAPT Server.
- If not authenticated, the User has still access to the packages repository which is public by design.

41.10 Coverage matrices

41.10.1 Threats and sensitive assets

The following matrix shows the coverage of threats to sensitive assets (the letters **D**, **I**, **C** and **A** represent **Availability**, **Integrity**, **Confidentiality** and **Authenticity** requirements respectively):

Table 1: Threat coverage of sensitive assets

	A1.Communications	A2.Inventory data	A3.Logs	A4.Configuration	A5.Client computers	A6.Authentication
T1.Installation of malware	I,C				I	
T2.Configuration alteration				I		
T3.Illegitimate access		I,C	D	I,C	I	I,C
T4.Network listening	C	C				
T5.Network alteration	I,A	I,A				

41.10.2 Threats and security features

The following matrix shows the coverage of threats by security functions:

Table 2: Threat coverage by security functions

	F1.Authentication access control	F2.Data protection	F3.Secured communications	F4.Package signature
T1.Installation of malware	✓	✓		✓
T2.Configuration alteration	✓	✓		
T3.Illegitimate access	✓			
T4.Network listening			✓	
T5.Network alteration			✓	✓

CHAPTER

FORTYTWO

PRESENTATION OF CRYPTOGRAPHIC PROCESSES

Date	Sep 20, 2024
Written by	Hubert TOUVET, Vincent Cardon
Applicable for WAPT	>= 2.3.0.13180
Version of the Document	2.3.0.0-0
Git hash	2e8409fd6f4e09ae76569c47197267f2c1d67d29

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 Agents.
- Signature of inventories and **status of WAPT Agents**.
- Authentication of the WAPT Agents Websocket connections on the WAPT Server.
- HTTPS communication between the WAPT Agents and the WAPT Server.
- HTTPS communication between the WAPT Console and the WAPT Server.
- HTTPS communication between the WAPT Agents and the WAPT repositories.

42.1 Folders and files referenced in this document

- `<WAPT>`: WAPT installation folder. By default `%ProgramFiles(x86)%\WAPT`.
- `<WAPT>\wapt-get.ini`: WAPT Agent configuration file (**`wapt-get`** and **`waptservice`**).
- `<WAPT>\ssl`: default directory of trusted certificates for signed packages and actions.
- `<WAPT>\ssl\server`: default directory for storing the WAPT HTTPS Server certificates (pinning).
- `<WAPT>\private`: default host key and certificate directory for signing the inventory and the Websocket connections.
- `%LOCALAPPDATA%\waptconsole\waptconsole.ini`: configuration file for the WAPT Console and package development actions for the **`wapt-get`** tool.
- `%APPDATA%\waptconsole\ssl`: default trusted certificate directory for importing packages from an external repository (i.e. *package templates*).

42.2 Definitions of Actors

- **Organization**

  An Organization is the realm of responsibility within which WAPT is used.

- **Certificate Authority**

  A Certificate Authority is the entity that keeps the keys that have been used to sign certificates for the *Package Developers*, the *Package Deployers* and the WAPT HTTPS Servers.

- **Administrators**

  Administrators 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 Agents**

  WAPT clients are the realm of devices that the *Organization* has allowed the *Administrators* to manage with WAPT. The clients **may or may not be a member** of the *Organization*'s Active Directory domain.

- **WAPT Server**

  The WAPT Server is the Linux / Nginx/ PostgreSQL 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 host account in the *Organization*'s Active Directory.

- **Internal WAPT repositories**

  Internal WAPT repositories are one or several Linux/ Nginx servers that deliver signed WAPT packages to WAPT clients using the HTTPS protocol.

- **External WAPT repositories**

  External WAPT repositories are 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**

  The Active Directory Server manages the *Organization*'s AD domain;

42.3 Summary of crypto modules present in WAPT

The WAPT Agent, the WAPT Server and the WAPT Console make use of both Python interpreter code and Lazarus / FPC compiled code.

The Lazarus / FPC code make use of the mORMot framework (>= 2.0.4383) for the https protocol handling, client kerberos handling, X509 certificates operations, and cryptographic operations (Hash, RSA).

The mORMot framework is itself configured and linked to make use of **`OpenSSL 1.1.1s 1 Nov 2022`** library for TLS sockets, asymetric RSA operations (key generation, encrypt, decrypt, sign, verify), and X509 certificates operations.

Python code (WAPT Agent, WAPT Server, WAPT Packages) is linked against the very same **`OpenSSL 1.1.1s 1 Nov 2022`** libraries and makes use of following modules:

WAPT Agent Side:

- **cryptography==3.3.2** and **pyOpenSSL==20.0.1** python modules linked on **`openssl 1.1.1s`**: used for all RSA crypto operations, X509 certificate generations and signature verifications in the python WAPT Agent.
- **winkerberos==0.8.0** (Windows agent) / **kerberos==1.3.1** (Linux agent) and **requests-kerberos==0.12.0**: used for authenticating the WAPT client on its first registrationg on the WAPT Server.
- **certifi==2021.5.30**: used as base for the Root Authorities certificates.

WAPT Server Side:

- **Python** 3.8.16 ssl module linked on **`openssl 1.1.1s`**.
- **cryptography==3.3.2** and **pyOpenSSL==20.0.1** linked on **`openssl 1.1.1s`**: used for all RSA crypto operations such as key generations, X509 certificate generations and signature verifications.

On the WAPT Server, the **`nginx/1.18.0`** service is configured to serve WAPT Packages over https, it handles https API requests to the WAPT Server, client Kerberos authentications, and client certificate check. The nginx server is configured for TLS1.2, cipher 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH'.

42.4 Types of PKI / CA infrastructures in a standard WAPT Setup

There are three types of PKI / CA in a standard WAPT setup

Table 1: PKI / CA types in a WAPT setup

Type	Usage	Origin and registration	Renewal	Revocation
HTTPS transport certificate	The purpose of this certificate is to secure the communications between the WAPT Agent and the WAPT Server and to authenticate the WAPT Server.	The certificate is issued by a *Certificate Authority* trusted by the client computer (the *Certificate Authority* must be integrated into the local Certificate Store), and the private https key must be stored on the **nginx** http server.	The certificate must be renewed as any https certificates handled by the *Organization*.	The certificate must be revoked as any https certiticates handled by the *Organization*.
The Administrator / Packager certificate	The purpose of this certificate is to sign Packages and Messages	This certificate is issued by the *Certificate Authority* of the *Organization*, and must be stored in the `<WAPT>\ssl` directory of each WAPT Agent that needs to trust that certificate.	The certificate must be renewed using the standard process of the *Certificate Authority* of the *Organization*.	CRL must be published through http and reachable by the WAPT Server, and the CRL distribution point attribute must be set in the certificate. The WAPT Server redistributes the CRL to the WAPT Agent.
The WAPT Agent client certificate	The purpose of this certificate is to authenticate the WAPT Agent with the WAPT server, to sign data and finally potentialy encrypt data that the WAPT Agent sends to the WAPT Server. This certificate is purely technical to identify a WAPT client and is created through a CSR process initiated by the client during registration of the WAPT Agent with the WAPT Server. The private key is stored on the client in `<WAPT>\ssl\private`, and the public key is stored on the WAPT Server.	The *Certificate Authority* is handled internaly by the WAPT Server and is used only for the authentication of clients on the WAPT Server.	The renewal is done during intial registration or when the client certification fails.	The revocation is handled internally by the WAPT Server which configures a CRL for the **Nginx** service.

42.5 Key and certificate management for the Administrators

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.

42.5.1 Validity of the Administrator's certificate

The WAPT Agent trusts all unexpired X509 certificates located in the <wapt>ssl WAPT Agent directory.

If a PEM encoded *.crt* file in <wapt>ssl WAPT Agent directory contains more than one certificates, only the first one is trusted.

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

42.5.2 Authorizing an 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%\waptconsole\waptconsole.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 *Code-Signing* type.

42.6 Managing the WAPT Agent's key and certificate

The WAPT Agent (**`waptservice`**) uses RSA keys and X509 certificates to authenticate itself with the WAPT Server.

The WAPT client certificate is used in the following situations:

- To control access to hosts packages and other packages repositories when client certificate authentication is enabled in NGINX server configuration.
- When updating the WAPT client status on the WAPT Server (**`update_server_status`**) **signing informations**.
- When the WAPT Agent establishes a Websocket with the WAPT Server (**`waptservice`**) **signing the WAPT client UUID**.

42.6.1 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 host 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 UID>.pem`.
- The generated certificate has the following attributes:
  - `Subject.COMMON_NAME` = `<device UID>`.
  - `Subject.ORGANIZATIONAL_UNIT_NAME = name of the Organization registered in the WAPT client's Windows registry`.
  - `SubjectAlternativeName.DNSName` = `<device UID>`.
  - `BasicConstraint.CA` = `True`.
  - `Validity` = `10 years`.
  - `Serialnumber` = `random`.
- The temporary self signed client certificate is saved in the `<WAPT>\private\<device UID>.crt`.

  Note: Only host 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 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* of the WAPT Client.
- After having initially authenticated the WAPT client with kerberos, the WAPT Server receives the certificate signing request sent by the Client, sign a proper client certificate and stores it in the table *hosts* of its inventory in `PEM` format (column *host_certificate*).
- The newly signed client certificate is sent to the client.
- The client then saves its certificate in the `<WAPT>\private\<device UID>.crt`.

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 client certificate that has been signed by the WAPT Server will be saved in the database.

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

PEM formatted certificates are stored in files with *.crt* or *.pem* extensions in the directory defined with the `public_certs_dir` parameter in the `<WAPT>\wapt-get.ini` file. 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 WAPT Console, the *Administrator* builds 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.

42.6.3 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` = `True` or `1`

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

- `verify_cert` = `<path to .pem>`

  This method checks 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` = `False` or `0`

  This method does not verify the HTTPS certificate of the WAPT Server.

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

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

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

During the build 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>\ssl\server` and the `verify_cert` parameter in section `[global]` in `<WAPT>\wapt-get.ini` is filled out to indicate the path to the certificate bundle.

42.7 HTTPS communication between the WAPT clients and the WAPT repositories

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

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

42.7.2 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.

42.8 Communications between the WAPT Console and the WAPT Server

42.8.1 Deploying certificates of Authorities of Certification

The `verify_cert` parameter in section `[global]` in file `%LOCALAPPDATA%\waptconsole\waptconsole.ini` can have different values:

- `verify_cert` = `True` or `1`

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

- `verify_cert` = <path to *`.pem`*>

  This method checks 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` = `False` or `0`

  This method does not verify the HTTPS certificate of the WAPT Server.

Conventionally, the approved bundle of certificates from the *Certificate Authority* is stored in the <WAPT>\ssl\server directory.

The WAPT Console includes a function that facilitates the initial recovery of the WAPT Server certificate chain and that stores it in *.pem* format in the <WAPT>\ssl\server\<server FQDN>.

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

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

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

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%\waptconsole\waptconsole.ini.

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

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

If the parameter is not explicitly mentioned, it is initialized at %APPDATA%\waptconsole\ssl.

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 authority_key_identifier or issuer_subject_hash attribute of the certificate. The signature of the certificate is verified using the **x509.verification.CertificateVerificationContext** class;

42.9 Process for signing a WAPT package

The process for signing a WAPT 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>**.
- Saving a *host* package in the WAPT Console.
- Editing or saving any package in the WAPT Console.
- Importing a package from an external repository.
- Creating a package with the setup wizard.

42.9.1 Initial parameters

- ZIP file of the WAPT 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)*;

42.9.2 Signing the attributes in the control file

The `control` file defines the metadata of a WAPT 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 taken in account in the signature;
  - `signer`: CommonName of the certificate holder for information;
  - `signature_date`: current date and time (UTC) in '%Y-%m-%dT%H:%M:%S format;
  - `signer_fingerprint`: hexadecimal encoded *sha256* fingerprint of the signer's certificate obtained with the **`fingerprint`** function included in the **`cryptography.x509.Certificate`** class.
- The signed attributes of the control structure are JSON encoded without space and line feed, and sorted in alphabetical order.
- 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.

42.9.3 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.

42.10 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.

42.11 Verifying the signature of a WAPT package

The verification takes place:

- When installing a WAPT package on a WAPT client.
- When editing an existing WAPT package.
- When importing a WAPT package from an external repository (if the checkbox is checked in the WAPT Console).

The verification consists of:

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

42.12 Signing immediate actions

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`.
- `start_waptexit`.
- `show_message`.
- `trigger_host_update_server_status`.
- `trigger_change_description`.
- `trigger_waptservicerestart`.
- `unregister_computer`.
- `trigger_gpupdate`.
- `trigger_waptwua_scan`.
- `trigger_waptwua_download`.
- `trigger_waptwua_install`.
- `trigger_waptwua_uninstall`.
- `trigger_host_audit`.
- `trigger_audit_packages`.
- `trigger_cleanmgr`.
- `trigger_host_reboot`.
- `trigger_host_shutdown`.
- `trigger_session_setup`.
- `run_wol`.
- `get_tasks_status`.

42.12.1 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.

42.12.2 Verifying the signature of an immediate action

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.

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

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.

42.13 Verifying the complete download of a WAPT package

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

When installing a WAPT package, the WAPT client checks whether a local version of the WAPT 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 WAPT package is deleted.

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

The checking of the signature of the WAPT package will be used instead of the *md5* sum to fully insure the integrity and the authenticity of the WAPT package.

CHAPTER

FORTYTHREE

(FOR SOFTWARE EDITORS) 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.

43.1 Environment variables

- Environmental variables have been around since DOS. They can make your (and my) life easier.

43.2 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 do not 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 do not need to actually know this;
  - `%SystemRoot%` is where the Windows directory is located. You hopefully do not care about this. Leave the Windows directory alone;
  - `%ProgramFiles%` is where you should place your program files, preferable in a `CompanyProgram` 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 is not user specific, but still needs to be written to by users (Users do not have write access to this folder either).

    Your program should not require administrator rights to run as you should not have us writing to the `%ProgramFiles%` directory. Also, do not 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 do not 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 host 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 host 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 do not need to sync up a thumbnail cache. They will not be transferred when a user migrates to a new host, 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 proper *`.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 will 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 do not / can not use environmental variables.

43.3 Logs

- Use the Windows Event Log for logging. It will 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.

43.4 Error codes

- Use documented Error Codes when exiting your program.

43.5 Printing

- Use the Windows printing API and do not use direct printing in your program.

43.6 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 is how we know it is 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**.

43.7 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 administration 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 do not allow users to run software updates, which is the policy often chosen in large security conscious enterprises.

43.8 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**!!

43.9 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 Organizational Units, so your WAPT package will behave in production exactly as a GPO would, **just much easier**.

43.10 License dongles

- USB license dongles are a sin. Use a regular software or network license. I am sure there are off the shelf ones so you don ot have to reinvent the wheel;

Note: You can make your software accept a license key as a parameter in your `.msi` executable.

WAPT can be used to assign license keys to individual workstations at install using a *method that ensures that the license key can not be read during transport*.

Then, if you want your software to call home to check on the validity of the license, make the routine work with *proxies*.

43.11 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 (I can not 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 will not even need an outgoing. Set those up at install time, not launch time. Use Firewall Groups so it is easy to filter. Do not use Any rules if you can help it. The goal isnot to make it work, it is to make it work securely. If you do not 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!

43.12 PDFs

- Do not 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` is the file format people use by default to exchange documents. PDF readers are meant to display documents, not execute unsigned programs.

CHAPTER

FORTYFOUR

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.

44.1 Release delay between the Enterprise and the Discovery versions

A new major version will be released as **Enterprise** and that same version will be released as an **Discovery** RC1 (Release Candidate #1). Before releasing, the Enterprise version will have undergone thorough internal testing and testing with valued **Insider Program** customers to insure no regression has slipped into the core of WAPT.

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 first release to our **Insider Program** customers.

This delay will provide several benefits to the General Availability release process:

- It allows additional time to perform in depth testing of the new Enterprise features while avoiding major regressions.
- It allows Tranquil IT to work with a small set of selected Enterprise customers to insure upgrade procedures work smoothly. As a reward, this selected set of Enterprise customers has direct access to Tranquil IT's developers and support team, allowing them to practice and learn new WAPT features before they become available to anyone else.
- It gives sufficient time to the forum and the mailing list to index questions and answers to eventually include into the official documentation.
- It gives Tranquil IT's documentation team a fixed functional target to document the new or improved features.
- It gives the translation team the necessary delay to update translations.
- It gives the communication and marketing team a fixed functional target and a capacity to backward schedule announcements, video podcasts and overall promotion.

CHAPTER

FORTYFIVE

SECURITY BULLETIN

45.1 WAPT-2021-01 : CVE-2021-38608

- Brief: Insecure permission allows a user running as guest to escalate privileges.
- Announced: August 13, 2021.
- Impact: **High**.
- Products: WAPT Enterprise & Community.
- Impacted versions: WAPT Enterprise < 2.0.0.9450, WAPT Enterprise < 1.8.2.7373 and WAPT Community < 1.8.2.7373.
- Description: Insecure permission allows guest OS users to escalate privileges via WAPT Agent.
- Reporter: Anass ANNOUR from the ORM/ITT&AC Risk Assessment Team, BNPParibas.
- Published CVE: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-38608.

CHAPTER

FORTYSIX

CHANGELOG

46.1 WAPT-2.4 Serie

46.1.1 WAPT-2.4.0.14143 (2023-08-08)

hash : 9847ee8b

This is a bugfix release for WAPT 2.4.0. Notable fixes are fixes are :

- better handling of scrolling in SelfService on macOS
- fix network error on macOS m1
- better support for authentication on WAPT Store Enterprise when downloading packages in the WAPT Console

WAPT Console

- [IMP] waptconsole import packages: avoid flickering when clicking on rows.
- [FIX] waptconsole / external repositories settings: renamed user and password fields to mention explicitly Store and token.
- [IMP] waptconsole import packages from store: handle 401 and 403 proactively to suggest user to authenticate to WAPT Store Enterprise and validate licences for proprietary software
- [IMP] better handling of icons list WAPT Self-service
- [FIX] fix waptconsole download waptagent for linux and mac (symlink for waptagent gui not properly handled)

WAPT core

- [FIX] better handling of current path when starting wapt: determine default_waptservice_ini with waptutils__file__, not from sys.argv[0] to handle
- [FIX] add use random uuid in json agent configurations

WAPT on Linux and macOS

- [FIX] Fix running_on_ac setuphelpers function on Linux
- [FIX] fix older macOS support specify `--platform macosx_10_9_x86_64` and `--platform macosx_11_0_arm64` when run pip compilation for backward compatibility
- [FIX] macOS : fix app startup icon not working on macos ventura and above
- [FIX] Debian : add dependency on rsyslog OR syslog-ng in server and service deb package
- [FIX] fixed socket ioctl() on some POSIX targets (e.g. macOS on M1 architecture)
- [FIX] fix scrolling WAPT Self-service under MacOS with magic mouse or macbook trackpad

WAPT Server

- [FIX] edit order check_auh for get_wads_config
- [FIX] fix db upgrade bug when upgrading from WAPT 1.8.2

46.1.2 WAPT-2.4.0.14080 (2023-06-22)

hash : 25f00c3f

This is a bugfix release for WAPT 2.4. Notable fixes are fix a for issues when building and uploading package from PyScripter due to `__pycache__` and `.pyc` files, and a fix for the broken WakeOnLan feature.

WAPT Console

- [FIX] waptconsole: show main_ip of pre wapt 2.4 host before upgrade
- [FIX] waptconsole gui: splitter position in softwares inventory
- [FIX] waptconsole : missing data in softwares inventory (host_capabilities)
- [FIX] waptconsole: label showing KBs usage space
- [FIX] waptconsole / sendMessage: don't autosize form as it creates endless layout loop on linux
- [FIX] waptconsole: MS remote assist is on port 135, not 3389

WAPT Core

- [FIX] wapt dynamic configuration: hiberboot_enabled is a boolean in json config, but must be set as a dword in registry
- [FIX] wapt-get build-upload: excluded files are not properly excluded when building the zip file due to `__pycache__` and `.pyc`
- [FIX] waptagent macox: using launchctl kickstart instead of launchctl unload && load for wapt service under MacOS

WAPT Server

- [FIX] server: reintroduce hosts.gateways extraction from host_networking
- [FIX] server / trigger wakeonlan: fix for compatibility with old host data.

WAPT WADS

- [IMP] send a human readable message to ipxe when WADS is disabled while trying to deploy through WADS
- [IMP] ensure WADS deployment and ipxe still works when djoin is empty

46.1.3 WAPT-2.4.0.14058 (2023-06-09)

hash : ae548d8ab

This is a bugfix release for WAPT 2.4.

Notable changes :

- Added support for Debian 12 amd64 on client and server
- Upgrade openssl from 3.0.8 to 3.0.9
- Upgrade python from 3.8.16 to 3.8.17

WAPT Server

- [NEW] add debian12 for amd64
- [UPD] no filter by default for importing WUA updates
- [UPD] adding more update file extension
- [FIX] handle server side Hosts dataset ordering (when a hosts count limit is given in waptconsole, we expect to get the first n hosts in the grid order)
- [FIX] waptserver : upload linux waptagent ensure symlink is secure filename
- [FIX] waptserver model: missing extraction of dnsdomain and mac from host_networking json into plain Hosts columns

WAPT macOS

- [FIX] direct waptservice restart on MacOS

WAPT Linux

- [NEW] add debian12 for amd64
- [NEW] Add new systemd function to setuphelpers for Linux

WAPT Console

- [FIX] fix waptpython.exe and waptpythonw.exe upgrade through innosetup when version id does not change
- [FIX] fix waptsetup install when setup file is located in directory with non ascii chars
- [FIX] Add escape_filter_chars for ldap3 (allow parenthesis and other special char in group names)
- [FIX] DJoin: fetch ldap search result until no more pages left
- [FIX] DJoin: Limit ldap search page to 500 results
- [FIX] showing pending WUA updates

WAPT Core

- [SEC] sign all dll and exe that are compiled by Tranquil IT during build process
- [SEC] switch to openssl 3.0.9
- [SEC] switch to python 3.8.17

46.1.4 WAPT-2.4.0.14031 (2023-05-26)

hash : 1420892a

This is the release of WAPT 2.4. WAPT 2.4 version brings a ton of small improvements and bugfixes along with the following main features:

- better co-existence with antivirus due to removal of NSSM service manager which was often wrongly flagged as suspicious. WAPT Agent now uses mORMot Angelize for service management
- due to OpenSSL 1.1.1 being eol'ed next september, WAPT has switch to embedded OpenSSL 3.0.8
- re-implemented Active Directory offline join in WADS (djoin.exe) to work around many bug and limitation in the Microsoft version of djoin.exe, now with support of Active Direcotry Forrest and subdomains
- it is now possible to have a use a user/password credentials when importing packages from the store. Authentification will be required for the WAPT Enterprise Store that provides educational softwares
- add support for Debian 10 and Debian 11 support on ARM 64 bit platform
- new WADS graphical interface
- remove usage of Microsoft Windows RestartManager during upgrade to avoid unecessary killing of services

CAVEAT:

- the new OpenSSL 3.0 has a huge performance issue when loading large certificate bundle. If you have verify_cert and want to use the Operating System bundle, please set **`verify_cert=1`**
- WADS WinPE format has changed and it needs to be recreated . Please refer to https://www.wapt.fr/en/doc-2.4/wapt-wads.html#adding-the-winpe-files

WAPT Server

- [NEW] waptserver: when login with ssl auth, check that the sha1 of the client certificate matches the sha1 of the user account in database for client cert auth
- [NEW] waptserver: accept empty username when using ssl auth. if username is provide, it must match the CN part of the certificate DN
- [NEW] use http status 403 instead of 401 when client side auth does not succeed to avoid a user/password popup in console.
- [NEW] waptserver: add login_auth_methods configuration parameter in waptserver.ini defaults to admin,ldap,passwd,token,kerb (format : csv)
- [NEW] waptserver licences: be tolerant if no server_uuid yet
- [NEW] wapserversession: share waptserveruser across all waptserver connection * to make it easer to relogin after token expiration. * retry to get a token if http 401 status
- [NEW] waptserver, waptservice on Windows: removed nssm service manager, replaced by waptsvc * waptsvc service supervisor is based on mormot agl. * waptservice.exe is a symlink to waptsvc.exe and manages "waptpython -I waptservice/service.py" * waptserver is a symlink to waptsvc and manages server.py, wapttasks huey queue, and nginx
- [NEW] waptserversetup: don't set repo_url and wapt_server url during setup as this done now later when building waptagent
- [ADD] WAPTWUA missing allow url allow mp.microsoft.com
- [RM] removed endpoint /api/v2/download_wuredist
- [IMP] lower case for test rules secondary repo in case of mixed case scenario
- [IMP] waptservice and wapttftpserver: don't wait for enter key on error
- [IMP] waptserver nginx: add api/v3/login specific section to forward client SSL auth
- [IMP] waptserver: add signer_fingerprint db field to Wads models
- [IMP] adding generic symlink when uploading waptagent to have standard http url for agent download
- [IMP] waptrepo: hardened handling of multiple concurrent repo cache updates
- [IMP] server add_configurations : return json config filenames in result.
- [IMP] waptserver: get_ad_ou_split : be tolerant to malformed OU sent by client
- [IMP] waptserver crls updates for nginx: * merge all known crls into file if "ssl_crls" waptserver.ini is defined
- [IMP] waptserver model: update Packages table description_localized dict from package entry.
- [IMP] add psycogreen patching for eventlet / postgresql
- [IMP] Be sure to fill executable version infos when initializing logger
- [IMP] cache CASigners in waptrepo
- [UPD] upgrade to 14.7 postgresql for windows
- [UPD] waptserver autocreate console ldap authenticated users if default_ldap_users_acls config is not empty
- [FIX] waptserver: fix startup issue when calling waptlicences.CheckValidLicencesCount
- [FIX] waptserversetup: missing dir=in in firewall rules for wapttftpserver on Windows Server
- [FIX] waptserver nginx: add "proxy_request_buffering off;" to the top server nginx config to workaround issues with big iso uploads.
- [FIX] fix username in log history of actions on waptserver

- [FIX] newest_only in api/v3/packages api does not compare versions properly.
- [FIX] fixed regexp in nginx location for `conf.d/*.json` files (and others).
- [FIX] waptserver: login initialization of user typo
- [FIX] configurations repositiories repo wapt/conf.d should not be protected by client side certificates
- [FIX] config url on server index landing page.
- [FIX] twaptserver auth callbacks. use OnHttpClientAuthorize if password in session, then OnAuthorize if defined and no password is available session
- [FIX] StripCertificateComments endless loop is Pem bundle ends with 2 CR NextPem does't not set input pointer P to nil if end of file.
- [REF] waptserver: add a config parameter to change globally the default enabled auth methods default_auth_methods defaults to session,admin,passwd,ldap this can be overriden on per endpoint basis
- [REF] server: removed legacy url style login

WAPT Agent

- [NEW] waptsetup: removed the option to trust tranquilit certificates.
- [NEW] don't set wapt-templates by default in agent config file wapt-get.ini
- [IMP] waptsetup: don't configure URL in waptsetup by default as it it proposed later on in waptconsole.
- [IMP] waptsetup: don't ask innosetup to close applications using RestartManager as sometimes, it kills vital services (network) when launched as silently
- [IMP] logo in WAPT SelfService
- [IMP] waptself: improve auth error message
- [IMP] waptself: removed shadows to lower redraw workload removed some visual overrides to panels
- [IMP] waptdeploy: useWaptServer task does not exist anymore. Enable installService task by default
- [IMP] WAPT Message adaptive form size to content if no size is set
- [IMP] waptstarter: fix some waptstarter default settings removed kerberos checkbox
- [IMP] wapt-get fpc: use agent key/cert client auth if none is defined in config inifile.
- [IMP] add double quotes around waptservice executable filename for ImagePath in services windows registry. If not quoted, and there are spaces in file path, service can not start in certain case
- [IMP] waptsetup: add logs of service install exec shell commands.
- [UPD] wapt-get: add restart-waptservice action. fix add-licence authentication
- [FIX] waptself: after hitting task panel hide button, packages flowpanel is hidden too
- [FIX] Self Service : DownloadAllPackageIcons after getting a token
- [FIX] restarting waptservice by scheduler under MacOS
- [FIX] taking care of display_time in WAPT Service
- [FIX] fix again regression on waptmessage impersonification from Agl waptservice. child processes are launched inside a job to control their termination. so for impersonification, we need CREATE_BREAKAWAY_FROM_JOB creation flag
- [FIX] waptsetup: add waptconsole start shortcut only if not running a stuffed waptsetup.exe

- [FIX] fix waptsetup trusted_external_certs

WAPT Linux

- [NEW] add json config url in waptserver homepage to help linux agent config
- [IMP] waptupgrade : improve command line install for deb base distro
- [IMP] Debian: add reboot_needed and reboot-required.pkgs info in host info
- [IMP] force locale C for strptime installed_softwares
- [FIX] fix datetime.datetime.strptime for installed_softwares in rhel9

WAPT macOS

- [NEW] WAPT Tray compilation config. for macosx
- [FIX] fix out of range error when importing waptlicences python module on macosx

WAPT Console

- [NEW] waptconsole acls form: fix the check signature action. add some icons to show when a certificate or password is assigned to a user
- [NEW] add HttpGet and HttpPost helpers for mustache templates to create custom html display in console
- [NEW] button export pending required WUA KB as curl string list
- [NEW] import CAB WUA updates
- [NEW] Showing pending WUA updates to download
- [NEW] audit info Add asus support button to asus support site with computer ref
- [NEW] WaptHttpGetString and WaptHttpPostData: add a default referer with root of URL to pass some basic access API authentication * applied as example for HP support access
- [NEW] add lenovo got to support button as an example of HttpGet mustache helper. * note the leading "," in the list of arguments because of a bug in mormot helpers arg handling.
- [NEW] add display time for WAPT Message when sending from WAPT Console
- [NEW] waptconsole: Enable audit data tab by default
- [ADD] message user friendly for '.exe' signature
- [ADD] Message to confirm hosts deletion
- [IMP] package maturity action
- [IMP] adding url for wsusscn2.cab to download
- [IMP] fix double click not able to show certificate using shell.
- [IMP] adding possibility to cancel configuration package creation
- [IMP] Add Tasks Status for better security and messages
- [IMP] waptconsole edit package form: show always files tab. add a message for user if package does not exist anymore.

- [IMP] WaptConsole: Discover domain controllers from domain dns name
- [IMP] WaptConsole: Load available OU from AD in TVisPrepareDjoin
- [IMP] User can add username / password for repositories while importing packages for Internet
- [IMP] better grid status if restart pending
- [IMP] external repositories settings: removed the checkbox for signature certificates directory. Check is enforced if cert is defined
- [IMP] waptconsole configuration: set verify_cert to 1 instead of path to certifi bundle when checking "Check https certificate".
- [IMP] waptconsole: on first login, when no server is defined in waptconsole.ini, show the configuration dialog first
- [IMP] waptconsole: manage reloading of ini config if file is updated externally add public_certs_dir setting.
- [IMP] waptconsole: trust always own waptconsole's user certificate when processing / resiggning packages
- [IMP] missing changes for waptconsole build waptsetup: don't include ssl dir in waptupgrade package.
- [IMP] waptconsole: try to get a new session cookie if 401 and there is cached password for user instead of switching to basic auth
- [IMP] waptconsole: Add update package tab in package editor
- [IMP] waptconsole: Display min/max os version in target_os column if defined.
- [IMP] waptconsole waptgent: allow to double click on certificates to open them with os shell.
- [IMP] waptconsole: add architectures arm and arm64 to the filters
- [IMP] new dark view mode for console
- [UPD] waptconsole: show login dialog if the server session cookies expires
- [UPD] add support for pkcs#12 file for private key and certificate in waptconsole and wapt-get.
- [UPD] waptconsole private key password change : try to change P12 file password too if same base filename and same old password.
- [UPD] icon on error status in host WUA
- [UPD] filter out packages having a untrusted signer certificate when loading Packages index note that this is only to avoid processing or listing packages which will not be trusted anyway. But we dont check the signature at this point, so package control signature must still be checked later.
- [FIX] waptconsole: fix potential AV when getting isEnterprise status if no waptserver is defined yet.
- [FIX] adding a password in Acls raise an exception about missing arg. fix decoding of utf8 when building SO and SA from Array of const (valid for lazatus only where String=Utf8String)
- [FIX] waptconsole reporting : no column displayed when running query outside of query editor
- [FIX] waptconsole acls: small fix console acls signature display when deleting a certificate in console
- [FIX] waptconsole: propagate licences count to background threads
- [FIX] TVisPrepareDjoin: Handle properly subdomains in AD Forrest
- [FIX] waptconsole PrepareDJoin: allow direct input of Host OU
- [FIX] give modal status to driver download windows when creating winPE to avoid other conflicting actions
- [FIX] splitter placement on audit data when showing history
- [FIX] Better Design for Import from Internet Basket

- [FIX] FrmLdapSearch: Fallback on OS DNS nameservers if no domain controller found using domain as nameserver
- [FIX] fix basic auth (issue when concatenating user+':'+password), prevent recursive call to login dialog, clear private key password if password is not OK on login.
- [FIX] waptconsole: fix local agent configuration based on built agent config
- [FIX] waptconsole : image showed as inactive on action forget package
- [FIX] waptconsole: empty server side message when upload error.
- [FIX] waptconsole import package: restore last used repository
- [FIX] waptconsole create waptsetup: handle the host_profiles config attribute * removed unused organisation.
- [FIX] waptconsole server login: be sure to not loop if basic auth fails
- [FIX] waptconsole import packages newer than mine when there are dots in names
- [FIX] deleting rows from audit data history
- [FIX] waptconsole regression decrypting old python rsa encrypted data
- [FIX] waptconsole decrypt of client side encrypted data
- [FIX] Clearing audit data history view if no data

WAPT Core

- [SEC] waptcrypto: don't try to guess signed_attributes. this attribute in mandatory. signer is mandatory for python waptcrypto verify_claim check
- [NEW] add wapt-get dmiinfo
- [NEW] showing countdown on WAPT Message + stopping countdown when entering in message viewer
- [NEW] GetStrippedDownServerCABundlePath : stores only issuer CA cert chain, not server chain. keep file cache for 1 hour.
- [NEW] improve handling of external repo user/password authentication.
- [IMP] waptsetup: don't change server and repo config by default if repo is already defined in wapt-get.ini.
- [IMP] wakeonlan: be tolerant if no interface or no macs on a host
- [IMP] fix get_net_ips() if not address on an interface (eg. CAN bus)
- [IMP] store networking infos as a separate field in hosts table. removed list_services and listening_sockets from host's status data moved audit_status into wapt_status
- [IMP] waptcrypto python: add arguments for certificates's not_before and not_after constraints add option to specify date of claim's signature for testing purpose.
- [IMP] waptrepo: Protect repo cache packages directory when updating. In case several process or threds are updating the same repo cache.
- [IMP] wapt-get waptdeploy waptlicences lpi wads wgetwads waptsvc: disable -Wg win32 app mode for win32 and win64 target to force stdout open.
- [IMP] waptcrypto: be sure to not create an empty stripped down CA file return full bundle path if function fails.
- [IMP] use mormot instead of tsmbios for get_biosinfos
- [IMP] mormot2 fix Samba LDAP expectations in its "strong auth = yes" default mode - i.e. allow "signing sealing" of the frames if TLS is not used

- [IMP] when checking for changed file over http, use a 2s tolerance before or after.
- [IMP] waptutils copytree2 : don't follow symlinks to avoid copying entire disks.
- [IMP] waptpackage get_stripped_package: include 'update_package.py' in payload for the console.
- [IMP] Add –only-priorities and –only-if-not-process-running to wapt-get upgrade, install, remove
- [IMP] logo for WAPT Message
- [IMP] waptcypto: TRSAPrivateKey: allow loading unencrypted PEM RSA key
- [IMP] fixed OpenSSL UTF-8 encoding flags for certificates closes
- [IMP] be sure to get only public cert from TX509Certificate mormot unit
- [IMP] add pfx and p12 file filter for personal cert file browser
- [IMP] waptdeploy: retry up to 30s to be able to get version on waptsetup
- [IMP] waptsetup/waptstarter: install /StartPackages=xx if runningSilently
- [IMP] create waptsetup: set verify_cert to '1' instead of path to cabundle if verify cert is checked.
- [UPD] update vc_redist to version 14.36.32532
- [UPD] avoid untrapped exception when password can not decrypt key
- [UPD] Strip comments in pem encoded certificates to reduce size and try to fit into the 32kb limit of stuffed exe.
- [UPD] manage multivalued "architecture" in wapt packages control.architecture attribuet is now a csv of x64, x86, arm, arm64, armhf
- [UPD] separate networking information from host_info to lower pressure on database when hosts update their status put host's audit_status in last_update_status key.
- [UPD] python waptpackage make_package_filename include os version in package filename for waptupgrade packages.
- [FIX] missing makepath import and syntax fix
- [FIX] waptpackage: remove references to old signature and manifest.sha1 files. delete them when unzipping package so that they are not considered as corruption.
- [FIX] fix python WaptRepo packages_matching when condition is a PackageRequest (this is actually unused. The method packages_matching of Wapt class is used instead)
- [FIX] allow empty folders in package
- [FIX] TWaptSignatureChecker.VerifyJsonSignature in case 'signed_attributes' is not supplied in the json.
- [FIX] DNS fallback to TCP on truncated UDP response - and also allow direct TCP query by using 'tcp@1.2.3.4' name server
- [FIX] waptutils python fileutcmtime and httpdatetime2time. Convert all dates to UTC
- [FIX] python wget not setting properly the file last-modified date from http header.
- [FIX] wapt-get / commandline : user RawReadKey from keyboard unit to avoid crt unit whicj breaks console.
- [FIX] wapt-get.py import waptservice is optionnal
- [FIX] fix Machine without main_ip are ignored
- [FIX] bad TTL for CACert bundle on disk cache
- [FIX] old bug causing removes to fail when software is already uninstalled
- [FIX] use '1' for system CA in external repositories to force use of stripped down CA bundles due to openssl 3.0 perf bug

- [REF] breaking change: removed import of PackageEntry from setupdevhelpers.py
- [REF] refactor the http client to handle all requests the same way. handle user:password embedded in Urls renamed proc InitTlsContext to func InitHttpTlsContext. Returns a PTlsContext moved GetServerCertificate to waptcrypto GetPeerCertChainFromServerPath
- [REF] move get_host_architecture from common to setuphelpers, move unzip_with_7zip from setuphelpers to setupdevhelpers

WAPT WADS

- [SEC] add iso hash in ipxescript
- [NEW] IP address and details of DISKPART info (volumes and disks) on wads_register_host
- [NEW] Wads with Graphical Display and Info
- [NEW] add update driver bundle option
- [NEW] reset drivers on hosts OSDeploy
- [NEW] drag and drop .iso on console for upload
- [NEW] drag and drop of drivers folder on drivers in WADS part
- [NEW] drag and drop from Host to deploy to drivers or configuration
- [IMP] Verify WADS hostname on WADS Winpe / Console / Server
- [IMP] Better login for login_on_wads
- [IMP] Wapt downloads are now in Graphical WADS
- [IMP] waptserver: calc sha256 of iso during upload rather than after upload
- [IMP] TVisPrepareDjoin: Add domain discovery
- [IMP] TVisPrepareDjoin: sort DC by response time using cldap
- [IMP] Save prepare djoin form fields in session (domain, username and password)
- [IMP] Add ubuntu and rhel9 wads template
- [IMP] Upload iso. Deleting file if wrong hash after upload
- [IMP] ipxe add keymap
- [IMP] sending file to api/v3/upload_deploy_files only if needed
- [IMP] Default prepare djoin window credentials to current domain's
- [IMP] Prepare Djoin: Retrieve domain controller using mormot dns resolver
- [IMP] On WADS conf, a password for superadmin is defined
- [IMP] Prepare DJoin: Connect through kerberos if possible
- [IMP] waptconsole PrepareDJoin: allow direct input of Host OU
- [UPD] wads: wait 30s for an ip address.
- [UPD] limiting uploading iso files only on WADS part
- [FIX] Wads fix default dir for iso upload
- [FIX] osdeploy data signature. signer_fingerprint is not saved into db, so must not be included in signed attributes

- [FIX] getting ipv4 addresses excluding APIPA
- [FIX] wads: break loop if 401 login fails.
- [FIX] Fix VisPrepareDJoin: Reset ldap kerberos SPN before connecting to the domain
- [FIX] Stop Graphical if WADS is only used to send status
- [FIX] Retry Wads now reset the status
- [FIX] avoiding loop showing message if ISO name already exits
- [FIX] empty error message on refreshing ISO file list
- [FIX] waptdeploy unable to read setup exe version same potential issue in wads missing call to RetrieveInformationFromFile-Name
- [FIX] fix copy cert in winpe for wads
- [FIX] empty error message on refreshing drivers file hashes and bundle names
- [FIX] Warning Removal and reset wads32 binary
- [FIX] Fix TVisPrepareDjoin GetDJoinBlob method - Fix verification of computer existence in the domain - Set computer password in AD even if we're not creating it - Parse the created djoin blob after creation and set an error if the format is invalid
- [FIX] TVisPrepareDjoin: Call to CldapSortHosts missing a parameter
- [FIX] TVisPrepareDjoin: Handle sub-domain within forest
- [FIX] waptconsole wads osdeploy grid: popupmenu clears multiselect
- [REF] Prepare djoin fixes and form rework - Allow to configure ldap port - Don't load OU on show - Split DC load and ldap connect buttons - Forbid to modify existing machine password (force to overwrite)

46.1.5 WAPT-2.4.0.14001-rc3 (2023-05-25)

hash : 1420892a

This is the third release candidate of WAPT 2.4. WAPT 2.4 version brings a ton of small improvements and bugfixes along with the following main features:

- better co-existence with antivirus due to removal of NSSM service manager which was often wrongly flagged as suspicious. WAPT Agent now uses mORMot Angelize for service management
- due to OpenSSL 1.1.1 being eol'ed next september, WAPT has switch to embedded OpenSSL 3.0.8
- re-implemented Active Directory offline join in WADS (djoin.exe) to work around many bug and limitation in the Microsoft version of djoin.exe, now with support of Active Direcotry Forrest and subdomains
- it is now possible to have a use a user/password credentials when importing packages from the store. Authentification will be required for the WAPT Enterprise Store that provides educational softwares
- add support for Debian 10 and Debian 11 support on ARM 64 bit platform
- new WADS graphical interface
- remove usage of Microsoft Windows RestartManager during upgrade to avoid unecessary killing of services

CAVEAT:

- the new OpenSSL 3.0 has a huge performance issue when loading large certificate bundle. If you have verify_cert and want to use the Operating System bundle, please set **`verify_cert=1`**

WAPT Server

- [FIX] waptserversetup: missing dir=in in firewall rules for wapttftpserver on Windows Server
- [FIX] waptserver nginx: add "proxy_request_buffering off;" to the top server nginx config to workaround issues with big iso uploads.
- [FIX] fix username in log history of actions on waptserver
- [FIX] newest_only in api/v3/packages api does not compare versions properly.
- [IMP] lower case for test rules secondary repo in case of mixed case scenario
- [IMP] waptservice and wapttftpserver: don't wait for enter key on error

WAPT Agent

- [FIX] Self Service : DownloadAllPackageIcons after getting a token
- [IMP] waptsetup: don't configure URL in waptsetup by default as it it proposed later on in waptconsole.
- [UPD] wapt-get: add restart-waptservice action. fix add-licence authentication
- [IMP] wapt-get fpc: use agent key/cert client auth if none is defined in config inifile.
- [FIX] restarting waptservice by scheduler under MacOS
- [IMP] add double quotes around waptservice executable filename for ImagePath in services windows registry. If not quoted, and there are spaces in file path, service can not start in certain case
- [IMP] waptsetup: add logs of service install exec shell commands.
- [FIX] waptself: after hitting task panel hide button, packages flowpanel is hidden too
- [IMP] waptdeploy: useWaptServer task does not exist anymore. Enable installService task by default

WAPT Console

- [FIX] waptconsole: fix potential AV when getting isEnterprise status if no waptserver is defined yet.
- [IMP] waptconsole configuration: set verify_cert to 1 instead of path to certifi bundle when checking "Check https certificate".
- [IMP] waptconsole: on first login, when no server is defined in waptconsole.ini, show the configuration dialog first
- [FIX] adding a password in Acls raise an exception about missing arg. fix decoding of utf8 when building SO and SA from Array of const (valid for lazatus only where String=Utf8String)

WAPT Core

- [FIX] missing makepath import and syntax fix
- [FIX] waptpackage: remove references to old signature and manifest.sha1 files. delete them when unzipping package so that they are not considered as corruption.

WAPT WADS

- [FIX] Wads fix default dir for iso upload

46.1.6 WAPT-2.4.0.14001-rc2 (2023-05-17)

hash : 13e724ad

This is the second release candidate of WAPT 2.4. WAPT 2.4 version brings a ton of small improvements and bugfixes along with the following main features:

- better co-existence with antivirus due to removal of NSSM service manager which was often wrongly flagged as suspicious. WAPT Agent now uses mORMot Angelize for service management
- due to OpenSSL 1.1.1 being eol'ed next september, WAPT has switch to embedded OpenSSL 3.0.8
- re-implemented Active Directory offline join in WADS (djoin.exe) to work around many bug and limitation in the Microsoft version of djoin.exe, now with support of Active Direcotry Forrest and subdomains
- it is now possible to have a use a user/password credentials when importing packages from the store. Authentification will be required for the WAPT Enterprise Store that provides educational softwares
- add support for Debian 10 and Debian 11 support on ARM 64 bit platform
- new WADS graphical interface
- remove usage of Microsoft Windows RestartManager during upgrade to avoid unecessary killing of services

CAVEAT:

- the new OpenSSL 3.0 has a huge performance issue when loading large certificate bundle. If you have verify_cert and want to use the Operating System bundle, please set **`verify_cert=1`**

WAPT Console

- [FIX] waptconsole reporting : no column displayed when running query outside of query editor
- [FIX] waptconsole acls: small fix console acls signature display when deleting a certificate in console
- [FIX] waptconsole: propagate licences count to background threads
- [FIX] TVisPrepareDjoin: Handle properly subdomains in AD Forrest
- [FIX] waptconsole PrepareDJoin: allow direct input of Host OU
- [FIX] give modal status to driver download windows when creating winPE to avoid other conflicting actions
- [FIX] splitter placement on audit data when showing history

WAPT Server

- [FIX] waptserver: fix startup issue when calling waptlicences.CheckValidLicencesCount
- [IMP] adding generic symlink when uploading waptagent to have standard http url for agent download
- [UPD] upgrade to 14.7 postgresql for windows
- [FIX] fixed regexp in nginx location for `conf.d/*.json` files (and others).

WAPT Core

- [FIX] fix python WaptRepo packages_matching when condition is a PackageRequest (this is actually unused. The method packages_matching of Wapt class is used instead)
- [IMP] wapt-get waptdeploy waptlicences lpi wads wgetwads waptsvc: disable -Wg win32 app mode for win32 and win64 target to force stdout open.
- [UPD] update vc_redist to version 14.36.32532
- [FIX] allow empty folders in package

WAPT Linux

- [IMP] waptupgrade : improve command line install for deb base distro

WAPT macOS

- [FIX] fix out of range error when importing waptlicences python module on macosx

46.1.7 WAPT-2.4.0.13958 RC1 (2023-04-17)

hash : 2cb08262

This is the first release candidate of WAPT 2.4. This new version brings a ton of small improvements and bugfixes along with the following main features:

- better co-existence with antivirus due to removal of NSSM service manager which was often wrongly flagged as suspicious. WAPT Agent now uses mORMot Angelize for service management
- due to OpenSSL 1.1.1 being eol'ed next september, WAPT has switch to embedded OpenSSL 3.0.8
- re-implemented Active Directory offline join in WADS (djoin.exe) to work around many bug and limitation in the Microsoft version of djoin.exe
- it is now possible to have a use a user/password credentials when importing packages from the store. Authentification will be required for the WAPT Enterprise Store that provides educational softwares
- add support for Debian 10 and Debian 11 support on ARM 64 bit platform
- new WADS graphical interface
- remove usage of Microsoft Windows RestartManager during upgrade to avoid unecessary killing of services

CAVEAT:

- the new OpenSSL 3.0 has a huge performance issue when loading large certificate bundle. If you have verify_cert and want to use the Operating System bundle, please set **`verify_cert=1`**

WAPT Console

- [FIX] Better Design for Import from Internet Basket
- [FIX] FrmLdapSearch: Fallback on OS DNS nameservers if no domain controller found using domain as nameserver
- [NEW] waptconsole acls form: fix the check signature action. add some icons to show when a certificate or password is assigned to a user
- [IMP] waptconsole: manage reloading of ini config if file is updated externally add public_certs_dir setting.
- [IMP] waptconsole: trust always own waptconsole's user certificate when processing / resiggning packages
- [IMP] missing changes for waptconsole build waptsetup: don't include ssl dir in waptupgrade package.
- [IMP] waptconsole: try to get a new session cookie if 401 and there is cached password for user instead of switching to basic auth
- [FIX] fix basic auth (issue when concatenating user+':'+password), prevent recursive call to login dialog, clear private key password if password is not OK on login.
- [UPD] waptconsole: show login dialog if the server session cookies expires
- [FIX] waptconsole: fix local agent configuration based on built agent config
- [NEW] add HttpGet and HttpPost helpers for mustache templates to create custom html display in console
- [IMP] waptconsole: Display min/max os version in target_os column if defined.
- [FIX] waptconsole : image showed as inactive on action forget package
- [FIX] waptconsole: empty server side message when upload error.
- [IMP] waptconsole: Add update package tab in package editor
- [FIX] waptconsole import package: restore last used repository
- [IMP] waptconsole waptgent: allow to double click on certificates to open them with os shell.
- [IMP] waptconsole: add architectures arm and arm64 to the filters
- [IMP] new dark view mode for console
- [NEW] button export pending required WUA KB as curl string list
- [NEW] import CAB WUA updates
- [IMP] adding url for wsusscn2.cab to download
- [IMP] fix double click not able to show certificate using shell.
- [NEW] Showing pending WUA updates to download
- [UPD] add support for pkcs#12 file for private key and certificate in waptconsole and wapt-get.
- [UPD] waptconsole private key password change : try to change P12 file password too if same base filename and same old password.
- [IMP] package maturity action
- [IMP] adding possibility to cancel configuration package creation

- [IMP] Add Tasks Status for better security and messages
- [IMP] waptconsole edit package form: show always files tab. add a message for user if package does not exist anymore.
- [FIX] waptconsole create waptsetup: handle the host_profiles config attribute * removed unused organisation.
- [IMP] WaptConsole: Discover domain controllers from domain dns name
- [IMP] WaptConsole: Load available OU from AD in TVisPrepareDjoin
- [NEW] audit info Add asus support button to asus support site with computer ref
- [NEW] WaptHttpGetString and WaptHttpPostData: add a default referer with root of URL to pass some basic access API authentication * applied as example for HP support access
- [NEW] add lenovo got to support button as an example of HttpGet mustache helper. * note the leading "," in the list of arguments because of a bug in mormot helpers arg handling.
- [UPD] icon on error status in host WUA
- [IMP] User can add username / password for repositories while importing packages for Internet
- [NEW] add display time for WAPT Message when sending from WAPT Console
- [FIX] waptconsole server login: be sure to not loop if basic auth fails
- [FIX] waptconsole import packages newer than mine when there are dots in names
- [UPD] filter out packages having a untrusted signer certificate when loading Packages index note that this is only to avoid processing or listing packages which will not be trusted anyway. But we dont check the signature at this point, so package control signature must still be checked later.
- [IMP] better grid status if restart pending
- [FIX] deleting rows from audit data history
- [FIX] waptconsole regression decrypting old python rsa encrypted data
- [NEW] waptconsole: Enable audit data tab by default
- [IMP] external repositories settings: removed the checkbox for signature certificates directory. Check is enforced if cert is defined
- [FIX] waptconsole decrypt of client side encrypted data
- [ADD] message user friendly for '.exe' signature
- [ADD] Message to confirm hosts deletion
- [FIX] Clearing audit data history view if no data

WAPT Agent

- [FIX] waptsetup: add waptconsole start shortcut only if not running a stuffed waptsetup.exe
- [FIX] fix waptsetup trusted_external_certs
- [IMP] WAPT Message adaptive form size to content if no size is set
- [NEW] waptsetup: removed the option to trust tranquilit certificates.
- [IMP] waptstarter: fix some waptstarter default settings removed kerberos checkbox
- [FIX] taking care of display_time in WAPT Service

- [NEW] don't set wapt-templates by default in agent config file wapt-get.ini
- [FIX] fix again regression on waptmessage impersonification from Agl waptservice. child processes are launched inside a job to control their termination. so for impersonification, we need CREATE_BREAKAWAY_FROM_JOB creation flag
- [IMP] waptsetup: don't ask innosetup to close applications using RestartManager as sometimes, it kills vital services (network) when launched as silently
- [IMP] logo in WAPT SelfService
- [IMP] waptself: improve auth error message
- [IMP] waptself: removed shadows to lower redraw workload removed some visual overrides to panels

WAPT Core

- [SEC] waptcrypto: don't try to guess signed_attributes. this attribute in mandatory. signer is mandatory for python waptcrypto verify_claim check
- [FIX] DNS fallback to TCP on truncated UDP response - and also allow direct TCP query by using 'tcp@1.2.3.4' name server
- [NEW] add wapt-get dmiinfo
- [IMP] waptcrypto: be sure to not create an empty stripped down CA file return full bundle path if function fails.
- [IMP] use mormot instead of tsmbios for get_biosinfos
- [FIX] TWaptSignatureChecker.VerifyJsonSignature in case 'signed_attributes' is not supplied in the json.
- [IMP] mormot2 fix Samba LDAP expectations in its "strong auth = yes" default mode - i.e. allow "signing sealing" of the frames if TLS is not used
- [FIX] waptutils python fileutcmtime and httpdatetime2time. Convert all dates to UTC
- [UPD] python waptpackage make_package_filename include os version in package filename for waptupgrade packages.
- [REF] breaking change: removed import of PackageEntry from setupdevhelpers.py
- [IMP] when checking for changed file over http, use a 2s tolerance before or after.
- [FIX] python wget not setting properly the file last-modified date from http header.
- [IMP] waptutils copytree2 : don't follow symlinks to avoid copying entire disks.
- [IMP] waptpackage get_stripped_package: include 'update_package.py' in payload for the console.
- [IMP] Add –only-priorities and –only-if-not-process-running to wapt-get upgrade, install, remove
- [IMP] logo for WAPT Message
- [IMP] waptcypto: TRSAPrivateKey: allow loading unencrypted PEM RSA key
- [IMP] fixed OpenSSL UTF-8 encoding flags for certificates closes
- [IMP] be sure to get only public cert from TX509Certificate mormot unit
- [IMP] add pfx and p12 file filter for personal cert file browser
- [UPD] avoid untrapped exception when password can not decrypt key
- [UPD] Strip comments in pem encoded certificates to reduce size and try to fit into the 32kb limit of stuffed exe.
- [IMP] waptdeploy: retry up to 30s to be able to get version on waptsetup
- [IMP] waptsetup/waptstarter: install /StartPackages=xx if runningSilently

- [FIX] wapt-get / commandline : user RawReadKey from keyboard unit to avoid crt unit whicj breaks console.
- [UPD] manage multivalued "architecture" in wapt packages control.architecture attribuet is now a csv of x64, x86, arm, arm64, armhf
- [FIX] wapt-get.py import waptservice is optionnal
- [IMP] waptsetup: don't change server and repo config by default if repo is already defined in wapt-get.ini.
- [IMP] wakeonlan: be tolerant if no interface or no macs on a host
- [IMP] fix get_net_ips() if not address on an interface (eg. CAN bus)
- [FIX] fix Machine without main_ip are ignored
- [FIX] bad TTL for CACert bundle on disk cache
- [IMP] create waptsetup: set verify_cert to '1' instead of path to cabundle if verify cert is checked.
- [FIX] old bug causing removes to fail when software is already uninstalled
- [NEW] showing countdown on WAPT Message + stopping countdown when entering in message viewer
- [NEW] GetStrippedDownServerCABundlePath : stores only issuer CA cert chain, not server chain. keep file cache for 1 hour.
- [FIX] use '1' for system CA in external repositories to force use of stripped down CA bundles due to openssl 3.0 perf bug
- [REF] refactor the http client to handle all requests the same way. handle user:password embedded in Urls renamed proc InitTlsContext to func InitHttpTlsContext. Returns a PTlsContext moved GetServerCertificate to waptcrypto GetPeerCertChainFromServerPath
- [UPD] separate networking information from host_info to lower pressure on database when hosts update their status put host's audit_status in last_update_status key.
- [IMP] store networking infos as a separate field in hosts table. removed list_services and listening_sockets from host's status data moved audit_status into wapt_status
- [NEW] improve handling of external repo user/password authentication.
- [IMP] waptcrypto python: add arguments for certificates's not_before and not_after constraints add option to specify date of claim's signature for testing purpose.
- [IMP] waptrepo: Protect repo cache packages directory when updating. In case several process or threds are updating the same repo cache.
- [REF] move get_host_architecture from common to setuphelpers, move unzip_with_7zip from setuphelpers to setupdevhelpers

WAPT Server

- [IMP] waptserver nginx: add api/v3/login specific section to forward client SSL auth
- [NEW] waptserver: when login with ssl auth, check that the sha1 of the client certificate matches the sha1 of the user account in database for client cert auth
- [IMP] waptserver: add signer_fingerprint db field to Wads models
- [NEW] waptserver: accept empty username when using ssl auth. if username is provide, it must match the CN part of the certificate DN
- [ADD] WAPTWUA missing allow url allow mp.microsoft.com
- [NEW] use http status 403 instead of 401 when client side auth does not succeed to avoid a user/password popup in console.

- [REF] waptserver: add a config parameter to change globally the default enabled auth methods default_auth_methods defaults to session,admin,passwd,ldap this can be overriden on per endpoint basis
- [FIX] waptserver: login initialization of user typo
- [REF] server: removed legacy url style login
- [NEW] waptserver: add login_auth_methods configuration parameter in waptserver.ini defaults to admin,ldap,passwd,token,kerb (format : csv)
- [FIX] configurations repositiories repo wapt/conf.d should not be protected by client side certificates
- [NEW] waptserver licences: be tolerant if no server_uuid yet
- [IMP] waptrepo: hardened handling of multiple concurrent repo cache updates
- [FIX] config url on server index landing page.
- [FIX] twaptserver auth callbacks. use OnHttpClientAuthorize if password in session, then OnAuthorize if defined and no password is available session
- [UPD] waptserver autocreate console ldap authenticated users if default_ldap_users_acls config is not empty
- [IMP] server add_configurations : return json config filenames in result.
- [IMP] waptserver: get_ad_ou_split : be tolerant to malformed OU sent by client
- [IMP] waptserver crls updates for nginx: * merge all known crls into file if "ssl_crls" waptserver.ini is defined
- [NEW] wapserversession: share waptserveruser across all waptserver connection * to make it easer to relogin after token expiration. * retry to get a token if http 401 status
- [NEW] waptserver, waptservice on Windows: removed nssm service manager, replaced by waptsvc * waptsvc service supervisor is based on mormot agl. * waptservice.exe is a symlink to waptsvc.exe and manages "waptpython -I waptservice/service.py" * waptserver is a symlink to waptsvc and manages server.py, wapttasks huey queue, and nginx
- [RM] removed endpoint /api/v2/download_wuredist
- [NEW] waptserversetup: don't set repo_url and wapt_server url during setup as this done now later when building waptagent
- [IMP] waptserver model: update Packages table description_localized dict from package entry.
- [FIX] StripCertificateComments endless loop is Pem bundle ends with 2 CR NextPem does't not set input pointer P to nil if end of file.
- [IMP] add psycogreen patching for eventlet / postgresql
- [IMP] Be sure to fill executable version infos when initializing logger
- [IMP] cache CASigners in waptrepo

WAPT WADS

- [FIX] osdeploy data signature. signer_fingerprint is not saved into db, so must not be included in signed attributes
- [IMP] waptserver: calc sha256 of iso during upload rather than after upload
- [FIX] getting ipv4 addresses excluding APIPA
- [IMP] TVisPrepareDjoin: Add domain discovery
- [IMP] TVisPrepareDjoin: sort DC by response time using cldap
- [IMP] Save prepare djoin form fields in session (domain, username and password)

- [FIX] wads: break loop if 401 login fails.
- [FIX] Fix VisPrepareDJoin: Reset ldap kerberos SPN before connecting to the domain
- [NEW] reset drivers on hosts OSDeploy
- [FIX] Stop Graphical if WADS is only used to send status
- [FIX] Retry Wads now reset the status
- [IMP] Verify WADS hostname on WADS Winpe / Console / Server
- [NEW] IP address and details of DISKPART info (volumes and disks) on wads_register_host
- [IMP] Better login for login_on_wads
- [IMP] Wapt downloads are now in Graphical WADS
- [NEW] Wads with Graphical Display and Info
- [FIX] avoiding loop showing message if ISO name already exits
- [FIX] empty error message on refreshing ISO file list
- [SEC] add iso hash in ipxescript
- [IMP] Add ubuntu and rhel9 wads template
- [UPD] wads: wait 30s for an ip address.
- [IMP] Upload iso. Deleting file if wrong hash after upload
- [IMP] ipxe add keymap
- [FIX] waptdeploy unable to read setup exe version same potential issue in wads missing call to RetrieveInformationFromFile-Name
- [FIX] fix copy cert in winpe for wads
- [FIX] empty error message on refreshing drivers file hashes and bundle names
- [NEW] add update driver bundle option
- [UPD] limiting uploading iso files only on WADS part
- [IMP] sending file to api/v3/upload_deploy_files only if needed
- [FIX] Warning Removal and reset wads32 binary
- [NEW] drag and drop .iso on console for upload
- [NEW] drag and drop of drivers folder on drivers in WADS part
- [NEW] drag and drop from Host to deploy to drivers or configuration
- [IMP] Default prepare djoin window credentials to current domain's
- [IMP] Prepare Djoin: Retrieve domain controller using mormot dns resolver
- [IMP] On WADS conf, a password for superadmin is defined
- [REF] Prepare djoin fixes and form rework - Allow to configure ldap port - Don't load OU on show - Split DC load and ldap connect buttons - Forbid to modify existing machine password (force to overwrite)
- [IMP] Prepare DJoin: Connect through kerberos if possible
- [FIX] Fix TVisPrepareDjoin GetDJoinBlob method - Fix verification of computer existence in the domain - Set computer password in AD even if we're not creating it - Parse the created djoin blob after creation and set an error if the format is invalid

- [IMP] waptconsole PrepareDJoin: allow direct input of Host OU
- [FIX] TVisPrepareDjoin: Call to CldapSortHosts missing a parameter
- [FIX] TVisPrepareDjoin: Handle sub-domain within forest
- [FIX] waptconsole wads osdeploy grid: popupmenu clears multiselect

WAPT Linux

- [IMP] Debian : add reboot_needed and reboot-required.pkgs info in host info
- [IMP] force locale C for strptime installed_softwares
- [FIX] fix datetime.datetime.strptime for installed_softwares in rhel9
- [NEW] add json config url in waptserver homepage to help linux agent config

WAPT macOS

- [NEW] WAPT Tray compilation config. for macosx

46.2 WAPT-2.3 Serie

46.2.1 WAPT-2.3.0.13516 (2023-02-23)

hash : 69968974

This is a bugfix release for WAPT 2.3.

Attention: When upgrading from WAPT 2.2.3 to WAPT 2.3, when installing the new **waptsetup.exe** 2.3, if the **waptagent.exe** 2.2.3 had previously been installed ON the management machine ABOVE the **waptsetup.exe** 2.2.3, then the org certificate located in `wapt\ssl` directory of the agent belonged to the **waptagent.exe** 2.2.3 InnoSetup installation instead of being a local file, and was removed during upgrade to **waptsetup.exe** 2.3, which handles certificate deployment differently.

Now, in the case a **waptagent.exe** has been installed above a waptsetup.exe install, the certificates in `wapt\ssl` will be preserved during upgrade. This should happen only on the managemnent machine that is used to rebuild the agent if the agent has been re-installed above the **waptsetup.exe** install.

Note: The RHEL9 repository are how signed with a sha256 key/digest

WAPT Agent

- [IMP] waptsetup.exe : backup <wapt>\ssl*.crt before upgrading and restore after install
- [UPD] when building waptagent, check that there is at least one trusted cert for packages and actions
- [UPD] be more relax on waptagent setup naming: if setup exename "starts" with waptagent, assume we can safely use the configuration inside when running silently
- [IMP] waptsetup: don't ask innososetup to close applications using Microsoft Windows RestartManager. Use specific process name instead.

WAPT Console

- [FIX] fix zip64 for big packages (>2GB) not handled properly in waptconsole
- [FIX] waptconsole build waptagent certificate issue when both CA and personal cert+CA files are selected

WAPT Server

- [FIX] Debian : fix logrotate on wapt server

46.2.2 WAPT-2.3.0.13505 (2023-02-13)

hash : c7fcb3a7

This is a bugfix release for WAPT 2.3, and has been signed with a new code signing certificate to replace the expired one.

Attention: All the previous version of the 2.3 branch have an issue with the creation of the waptagent.exe due to a expiring code signing certificate. If you need to create a new WAPT Agent, please upgrade to this version.

The error message that you will get is "Error while creating waptagent.exe: Checking hashes of executables on server against Tranquil IT certificate has failed. Please check if waptbinaries.sha256 has not been altered."

Message in French : "Erreur lors de la création du waptagent.exe : La vérification de la signature Tranquil IT des hashs de contrôle sur le serveur a échoué. Vérifier que le waptbinaries.sha256 n'a pas été altéré sur le serveur."

WAPT core

- [FIX] better handling of filename with '..' and '~' in zip filenames. No need to be paranoid if '..' and '~' are in the middle of the name
- [FIX] waptservice only_if_no_process_running not taken in account when auto upgrade with waptupdate_task_period is enabled.
- [UPD] waptservice / core: include packages with install status == error when checking for conflicting packages to remove.
- [FIX] remote user waptmessage encoding issue
- [FIX] waptconsole waptpackage manifest add support for file with non ascii chars.
- [IMP] read Packages index from disk: use mormot function to potentially avoid lock conflicts
- [FIX] remove or forget packages with spaces in package name. fix RemoveDuplicates when there are spaces in data items.

- [FIX] closing WAPT Self for Linux/MacOSX
- [FIX] waptdeploy : update certificate pinning with new code signing certificate
- [FIX] waptcrypto : takes into account signature_date when checking certificate expiration date vs timestamping time.
- [SEC] update openssl binaries to 1.1.1t

WAPT Server

- [FIX] waptdeploy on server location: <repodir>/waptagent/waptdeploy.exe
- [SEC] add server_tokens off to avoid giving nginx server version to non authenticated clients
- [SEC] delete waptversion in /ping to avoid giving waptserver version to non authenticated clients
- [IMP] add view acl for get_storage_used_by_kbs

WAPT WADS

- [FIX] check volume letters before diskpart closes
- [IMP] waiting network for wgetwads Closes
- [IMP] install waptagent at end pressed debian
- [IMP] not force login in ipxescript if login already in ipxescript (for leave the possibility of forcing the language before)
- [IMP] add keymap on menu register
- [IMP] add login in pxe for linux deploy
- [IMP] delete double login wads

46.2.3 WAPT-2.3.0.13470 (2023-01-26)

hash : 4cc5fc06

This is a bugfix release for WAPT 2.3, and add support for Red Hat Enteprise Linux 9 and derivatives (both as server and agent)

WAPT Core

- [FIX] fix waptdeploy.exe unable to read setup exe version, requiring the use of force flag in GPO

WAPT Agent

- [FIX] fix datetime display for software inventory on Redhat and derivatives
- [IMP] better support for Red Hat os version numbering in inventory and tags
- [NEW] add el9 waptagent and waptserver support

WAPT Server

- [IMP] simplify web interface displayed version values to avoid misunderstanding
- [UPD] waptserver autocreate console ldap authenticated users if default_ldap_users_acls config is not empty
- [FIX] fix update_hosts_sid_table connexion leaks (to update the reachable column before calling query in reporting tab)
- [NEW] add el9 waptagent and waptserver support

WAPT Console

- [FIX] fix package maturity action default value if none chosen
- [FIX] fix grayed out host packages actions in Discovery mode
- [UPD] Strip comments in pem encoded certificates to reduce size and try to fit into the 32kb limit of stuffed exe.
- [IMP] adding possibility to cancel configuration package creation

WAPT WADS

- [IMP] add support for keyboard selection in ipxe
- [FIX] fix template windows 11 wads
- [UPD] wads: wait 30s for an ip address if dhcp is slow to respond or waiting for 802.1x vlan switch
- [FIX] fix wads regression where a computer could connect to waptserver instead of local secondary repo
- [IMP] Upload iso. Deleting file if wrong hash after upload
- [FIX] fix copy cert in winpe for wads
- [FIX] fix waptdeploy unable to read setup exe version, requiring the use of force flag

46.2.4 WAPT-2.3.0.13438 (2023-01-19)

hash : 8e580896

This is a bugfix release for WAPT 2.3. Those are mainly fixes and improvements to smooth the upgrade process from older WAPT versions.

WAPT Core

- [FIX] waptcore: keep install status of previous package if new package upgrade status is ERROR
- [FIX] Don't forced install packages which could't not be installed properly last time (to avoid install loop) a better approach could be to define a maximum retries count and an increasing delay between retries.

WAPT Console

- [FIX] fix verify **waptsetup.exe** and **waptdeploy.exe** hash when creating waptupgrade
- [UPD] set all search timer to default (300ms)
- [FIX] waptconsole display correct icon on Linux
- [UPD] waptconsole: propose to add a licence right after login if none on server.
- [FIX] waptconsole: fix some tab orders in forms
- [FIX] waptconsole package wizard: change layout for compatibility with linux.
- [FIX] waptconsole: quick fix for external repos settings if none is currently defined in waptconsole ini settings. Autoregister `wapt-templates`.
- [FIX] waptsetup: don't create a shortcut for the waptconsole to replicate behavior from older waptsetup…
- [NEW] waptagent for Windows can be generated on Linux waptconsole
- [REF] Improved djoin support
- [NEW] waptconsole: better support for dark mode on Linux / MacOS

WAPT Agent

- [IMP] macOS: use **sw_vers -productVersion** for mac os version
- [FIX] windows: waptwua client: fix issue when main repo url ends with a slash
- [FIX] fix **wapt-signpackage** compatibility error : removes mds argument
- [FIX] fix waptupgrade package for centos
- [FIX] fix application version on MacOsx
- [FIX] switch `DisableSkipWindowsUpdates` to `waptwua` section
- [NEW] Add ForceUnsigned for add drivers in winpe
- [FIX] add defaultInterpreterPath for vscode support
- [FIX] waptexit self-kill if machine has been started for too much time

WAPT WADS

- [IMP] wads: removing mounted drive letters before diskpart for better support of machine without any installed OS
- [NEW] Add script compile_ipxe.py to integrate waptserver url directly in ipxe binary
- [FIX] fix acl wads_admin on upload_winpe
- [FIX] wads: fix wads skip_login_wads and acl

WAPT Server

- [FIX] waptserver: don't try to convert jsonb boolean to raw boolean as it fails for postgresql <= 10
- [FIX] better support for postgres upgrade for Debian / Ubuntu in **postconf.py**
- [FIX] waptserver: path to **waptdeploy** on windows server to fix link
- [FIX] during upgrade, run **/opt/wapt/wapt-scanpackages.sh** when run **postconf.py**
- [NEW] waptserver: new option to set nginx port from waptserver.ini

46.2.5 WAPT-2.3.0.13356 (2023-01-10)

hash : fd590589

This is the first release of WAPT 2.3. This release does not have any new big feature, but brings a ton of little bugfixes and improvements to make WAPT usage more lean and smooth.

What's New?

- 1000+ bugfixes
- Less issues with false positive with antivirus software when deploying a new agent: WAPT Agents do not need to be rebuilt. The WAPT Agent is based on **waptsetup.exe** with certificate and configuration stored in the certificate signature of the file. The signature of the file is not altered.
- WAPT Agent on Linux and macOS: improved workflow for installing and updating the WAPT Agent.
- Improved Websocket connexion. Disconnects and reconnects have be made more robust.
- Improved support on macOS.
- Improved support on Linux.
- Update of WAPT external components.
- Tech Preview : WAPT Console support on Linux (Debian and derivatives, Redhat and derivatives)
- Tech Preview : WAPT Console support on macOS (Mojave and above).

Upgrade details

WAPT Server 2.3 needs PostgreSQL 10 or above. Please be sur to have the correct version running, especially if your server is running Debian and has been upgraded from Stretch:

- If the WAPT Server is running on Debian or Ubuntu, if you have upgrade from Debian Stretch to Buster to Bullseye, please check that the running instance of PostgreSQL has been upgraded when the OS has been upgraded;
- If you are on Redhat 7, upgrade is taken care of in the postconf script, and it should upgrade from 9.6 to 14;
- If the WAPT Server is running on Redhat 8 or derivative, then the DB is already in a good version;
- If the WAPT Server is running on Windows the DB upgrade is done during the upgrade from 9.6 to 14.

WAPT Core

- [SEC] When checking exe certificate, first check that the signature is OK.
- [SEC] when stuffing waptsetup.exe, check that waptsetup.exe downloaded from wapt server is properly signed by Tranquil IT.
- [FIX] Fixed handling properly utf8 chars in certificate subject.
- [FIX] Small improvement for wapt-get build-waptagent. Do not ask the server password twice.
- [FIX] Fixed stuffed legacy waptagent build: be sure to have a deterministic binary result when stuffing in waptconsole or server side.
- [IMP] remove client library dependency for command line progress bar.
- [SEC] waptpython 3.8.16 is now compiled with the isolated mode flag at true by default (Python -I)
- [REF] Removed unused functions.
- [REF] Removed unused headers.
- [IMP] waptservice: fix setting loglevel for specific components do not log WS listening too often. Fixed some action's "created_by" attributes which were not not set.
- [FIX] Windows setuphelpers: missing service_list in _all__.
- [FIX] **`wapt-get`**: moved *LoadOpenSSLFromPythonLib* to get proper path for *RegWaptBaseDir* on Linux.
- [NEW] Added armhf as a valid package architecture.
- [FIX] Fixed `scan_package` issue when there are packages without `package_uuid`. Packages table was growing at each `scan_packages`.
- [IMP] **`wapt-get`**: Added some help for `build-waptagent` and `add-config` / `reset-config`/ `set-config` *-from-url*.
- [IMP] `wapt-get reset-config-from-url`: removes dynamic configs from `conf.d` too.
- [IMP] Re-include empty folders in zipped WAPT packages.
- [FIX] Update for zip empty folder entries.
- [FIX] When checking files and directories from package manifest, create empty directories from the `manifest` file if thet do not exist yet.
- [UPD] `wapt-get update-package-sources`: Implicit transparent import of all functions from `packagesdevhelpers.py`.
- [FIX] Do not audit packages with `install_status` <> 'OK'.
- [SEC] waptpackage: Cleanup removed multiple MD type. We use only sha256 now.
- [NEW] waptconsole: Stuff `waptsetup` with *`json`* config for embedding into *waptupgrade* package.
- [FIX] waptpackage signature issue if the WAPT package is created from scratch with null attributes (ex. max_os_version). If signed, these null attributes are written to control file as sempty string, this breaks the signature control. So we initialize all default signed attributes to empty string instead of null.
- [UPD] `wapt-get create-waptagent`: Use *`json`* embedded config stuffed into certificate zone of executable signature.
- [FIX] Fixed regression in python _sign_control (encoding issue).
- [UPG] Upgraded python to 3.8.16.
- [IMP] `waptutils.py` cleanup and small fix in `user_is_member_of`.
- [REF] waptserver: Cleanup code with **`pyflakes`**.

- [IMP] Allow *none* loglevel.
- [NEW] Introduced `wapt-get reset-config-from-url`.
- [FIX] Fixed json_load_file() by adding encoding option. Default is "utf-8".
- [IMP] waptguihelper: Introduced StayOnTop argument for input_dialog() and grid_dialog()
- [FIX] Fixed `wapt-get add-config-from-url` in pure Pascal. The hash is retrieved from the filename if present, or as second parameter of command line.
- [REF] wapt python core: Removed sha1 compatibility with wapt 1.3 packages signatures.
- [FIX] Shows the proper logged user after login.
- [IMP] Fallback other method for get domain in get_hostname.
- [REF] `jsonconfig` data embedded in setup exe.
- [FIX] Default value for check verify cert.
- [UPD] Introduced uwaptjsonconfig (port of json config from python to FPC (FreePascal Compiler)).
- [UPD] **`wapt-get`**: Added a command to list the initial configs available on server (in `wapt/conf.d`).
- [UPD] waptsetuputil: Added UnzipConfigFromExe.
- [FIX] Removed global variable for PopupEnterprise, check Licensing after closing the window.
- [IMP] buildlib: Do not remove unittest from python lib when creating the build environment.
- [FIX] `remove_file()` was unable to remove symlinks.
- [FIX] wapt core: Regression on uuid retrieval from WMI. 'System_Information' key is an array.
- [NEW] wapt core: added "wapt_temp_dir" `wapt-get.ini` parameter to specify the directory wher packages are unzipped at installation (for wyse terminal).
- [REF] Introduced packagesdevhelpers python module to remove helpers useful only for "packages source update" and reduce import time of setuphelpers.
- [IMP] windows_version() now getting the correct UBR (Update Build Revision) shown with "winver" command, adding windows_version_full in hardware inventory
- [IMP] waptguihelper: help improved for grid_dialog - also, introduced an (optional) Text parameter.
- [FIX] waptpackage: trim attributes value in `control` data. ('all' was retrieved as 'all ').
- [IMP] twaptpackage: Always set architecture and priority default.
- [UPD] Refactored SSLCABundle usage.
- [FIX] Fixed waptpackage build issue when sourceroot includes the ending path separator. Fixed self service package building. Fixed version incbuild result.
- [FIX] Fixed issue with in path in zipped files created with `CreateRecursiveZip`.
- [FIX] Fixed file not found when calling `GetServerCertificate`.
- [FIX] Fixed editing zipped package inplace (hosts packages).
- [FIX] Added call to mormot2 `RegisterOpenssl` for Access violation in **`waptlicences`**.
- [IMP] Grid editor: Show which column is currently focused even if grid has not the focus.
- [IMP] Use UTC (Coordinated Universal Time) time for expiration check of ACLs.

- [UPD] wapt core: use datetime in UTC for `audit_data`.
- [IMP] wapt core: allow usage of an environment variable *waptbasedir* to specify the location of root *waptbasedir*.
- [IMP] Default grid order set to descending signature date.
- [FIX] Allow ~ character in WAPT package names (for spaces in Organizational Units packages).
- [FIX] waptcrypto: Fixed certificate filename attribute not set when loading a certificate chain.
- [UPD] Refactored `SSLCABundle` usage.
- [FIX] Fixed using particular characters in passwords.
- [FIX] Fixed waptcore: Fixed the type for dynamic configuration.
- [FIX] copytree2 `replace_at_next_reboot`.
- [REF] Moved all the dynamic json config functions into the WAPT class to take in account the actual agent settings (specially directories).
- [UPD] Created a full version 1.2.3.rev-hash into file `wapt/version-full`.

WAPT Agent

- [FIX] force create random uuid if bios uuid is not correct.
- [FIX] Do not check `wsusscn2.cab` if not Enterprise.
- [IMP] add host_capabilities inventory.
- [IMP] Better JSON format (Human Readable) for Audit Data.
- [FIX] Use parameter `IncludeCA` on `ListSOCertificatesFromFolder`.
- [FIX] Fixed translation issues in graphical components.
- [FIX] Fixed last version, checks the minimal OS version
- [FIX] edit waptwua if `install_delay` has value.
- [IMP] When uninstalling the WAPT Agent, stop the **`waptservice`** only if the service exists.
- [FIX] Popping wrong license message on new installation.
- [FIX] waptservice socketio: Force get new ws params in case of connection error like when config is updated.
- [FIX] Fixed add new rule missing import for `isenterprise`.
- [NEW] Added disk drives to host overview template.
- [IMP] Reduced size of host *`json`* inventory data. Do not send host configurations data if not changed. Do not send audit_data headers if no data. Fixed last audit data that was always sent.
- [IMP] Improved local waptservice auth feedback.
- [REF] Refactored waptservice code.
- [FIX] Enable custom CA file for websockets certificate checking.
- [FIX] Fixed WAPT Agent `websockets_verify_cert`: error reading setting from *`ini`* file. Reset socketioclient to None when connection error to force recreating the object with new TLS (Transport Layer Security) settings.
- [IMP] waptdeploy: Use only registry wapt_is1 install location to get the WAPT base directory.
- [IMP] waptdeploy: Do not check **`wapt-get`** working condition.

- [FIX] Fixed waptdeloy argument parsing.
- [UPD] waptsetup: Removed distribution of **innosetup** as it is no longer needed.
- [NEW] waptdeploy: Check that the WAPT Agent installer and **wapt-get.exe** are digitally signed by Tranquil IT.
- [FIX] waptdeploy wapt basedir guessing. Hardened waptdeploy RunTask.
- [FIX] Fixed `wapt-get build-waptagent`: empty configuration name.
- [ADD] Check all rules signatures before doing anything else.
- [IMP] The agent version is obtained from the *exe*, not the server.
- [FIX] waptsetup auto json config: should accept `waptsetup-1.2.3_<confname>_<confhash>.exe`.
- [FIX] Fixed remote WakeOnLAN.
- [IMP] waptservice: Do not include *PrinterPaperNames*, *PaperSizesSupported* and self_service_rules in inventory sent to the WAPT Server.
- [FIX] waptexit: If unable to get licences from waptservice, assume *is_enterprise* is False.
- [FIX] wapt-get: Set password callbacks after reloading config.
- [FIX] Shortened the upgrade scheduled task argument, as it is limited to 256 chars.
- [FIX] Stuffed waptsetup: Append waptwua settings to *json*.
- [FIX] waptserver socketio: Host does not register / reconnect by itself when deleted from the WAPT Server.
- [NEW] waptsetup.exe : If waptagent.exe is named, and only one config is embedded, take the first available config for the name of the configuartion to install instead of hardcoded "default".
- [IMP] waptservice: Can start right after install even if no `wapt-get.ini`.
- [NEW] Added *nopassword* to config wizard for `service_auth_type`.
- [UPD] Added `wapt-get reset-config-from-url` and `set-config-from-url` json configuration.
- [FIX] Do not delete the files if the signature has failed.
- [IMP] waptsetup: Display a summary of embedded stuffed json configurations. Removed *use dynamic configuration* task.
- [FIX] waptserver: Fixed WakeOnLAN issue when no broadcast address exists in inventory.
- [FIX] `remove_user_appx` was not initialized from setuphelpers.
- [UPD] waptsetup: ApplyJsonConfigToIniFile when a *json* configuration is stuffed instead of `conf.d` dynamic configuration.
- [IMP] waptsetup: Do not update `wapt-get.ini` when using dynamic *json* configuration.
- [UPD] waptservice socketio: Do not require connection params update / reconnection try if there is no authorization token. When `allow_unauthenticated_connect` = **True** on the WAPT Server, the WAPT agents should be able to connect without getting a token.
- [FIX] waptself: Fixed next page button unavailable on last page.
- [UPD] waptexit: Add `waptexit_disable_skip_windows_updates` parameter in `wapt-get.ini` file and commandline argument to disable the checkbox for skipping Windows Updates.
- [UPD] wapt-get: Return ExitCode <> 0 when an exception is raised Added **ping --service** command to check waptservice accessibility from waptsetup.
- [UPD] waptself: Display details of WAPT package on top of packages list to avoid reframes.
- [UPD] Enable `waptservice_allow_all_packages` only for *nopassword* `service_auth_type`.

- [NEW] Added a waptservice parameter `waptservice_allow_all_packages` which allow all user to install / remove all packages as if they were part of the waptselfservice group.
- [NEW] If a *json* configuration is provided in waptsetup as stuffed data in certicode certificate area, use it for initial configuration.
- [FIX] Improved error message and wait cursor when waptselfservice is starting.
- [FIX] Fixed selfservice missing common module for `self_service_rules` when using the *nopassword* argument with the WAPT Enterprise version.
- [FIX] Changed Icon for *Add Dependencies* → *Trashcan* to *Plus*.
- [IMP] User is now informed when self service can not get a token (service not started).
- [FIX] Remove double slahs in url *//Packages*.
- [NEW] Added Ubuntu22 in waptsetup package.
- [FIX] Fixed waptmessage ambiguous '-s' option (use stdout and set window size), replaced by -c for init console.
- [FIX] Fixed tasks list on host.
- [FIX] Normalized view (lowercase) in grid for *target_os* from control part.
- [FIX] Fixed execution of waptmessage in file instead of base64 (to avoid too long command line).
- [FIX] Use cached trusted signer certificates store instead of recreating it each time.
- [FIX] Fixed signed_attributes written as string list (instead of python form) and signer is the signer certificate *Common Name*.
- [IMP] use **`--not-interactive`** with register if the installation runs in silent mode.
- [FIX] waptservice: Do not ignore broadcast for `WaptUpdateServerStatus` to avoid the WAPT Tray sticking upon sending data to the WAPT Server.
- [FIX] Fixed unable to synchronize remote repository.
- [IMP] waptmessage: No autosize if a size is specified on the command line.
- [FIX] Fixed no hash in clipboard, added missing helper for `add-config-from-url` in wapt-get.
- [IMP] Limit access right to Administrators to log directory (in case non public stuff gets written to logs).
- [IMP] `install_scheduling` work if not in PENDING_UPDATES status.
- [FIX] Fixed waptexit compilation: Removed specific `WaptIniFilename` function.
- [FIX] Fixed waptmessage unable to load sqlite.
- [IMP] Updated waptwua status to 'NEED-SCAN' on hosts when `download_wsusscan` is triggered and `wsusscn2.cab` file is downloaded.
- [NEW] wapt core: Added `as_dict` and descending parameters to `Wapt.read_audit_data_set`.
- [IMP] Do not take care anymore of maturity for version when it is compared to WAPT store version.
- [FIX] Fixed configuration package template `setup_package_template_conf.py`.
- [FIX] Fixed waptservice configuration: Set the `configs_dir` relative to `wapt-get.ini` full path.
- [FIX] Fixed waptservice 'start_waptexit' with arguments Faulty arguments boolean value decoding.
- [FIX] Fixed bad arguments sent to waptservice triggering upgrades with `only_priorities` and `only_if_not_process_running`.
- [FIX] Fixed `Wapt.write_audit_data_if_changed`: Write data if previous data has expired.

- [FIX] Updated the template of dynamic *json* configuration packages to match new location and naming of *json* configuration related functions.
- [NEW] Option `include_potentially_superseded_updates` in configuration wizard.
- [FIX] Fixed waptservice: Be sure to dynamically revert to default setting when a key is removed from `wapt-get.ini`.
- [FIX] Fixed waptservice: Make sure we have a random `secret_key` for local waptservice session.
- [NEW] WAPTWUA superseded support.
- [IMP] **`wapt-get edit`** now opens `update_package.py` too.
- [UPD] Added a *NEED-SCAN* waptwua.status, updated when `Wapt.update()` is called.
- [FIX] Fixed waptself: Set focus on search when opening.
- [IMP] Ignore history for waptwua status.
- [FIX] Fixed **`wapt-get update-package-sources`**: Handle properly relative path to package sources.
- [FIX] Fixed **`wapt-get update-package-sources`**: use `devdirupdate_package.py` to call `update_package()` hook if this file exists. Else use `setup.py`.
- [IMP] wapttray: Launch external **`waptself`** and **`waptconsole`** with OpenDocument instead of windows only ShellExecuteW.
- [FIX] Workaround fix when **`pyscripter`** is put as editor for packages. `params_vscod_list` fixed when space in parameters. Reupdated description.
- [IMP] **`wapt-get edit`** now opens `changelog.txt`, VSCod* now opens `control` file too. **`wapt-get edit`** can now be run as user with VSCod* updating `wapt_sources_edit()` description.
- [UPD] Changed default log path to `wapt/log` if writable.
- [UPD] Same logging initialization code for all UI executables with `waptcommon.InitLoggingFromCommandLine`.
- [IMP] waptservice waptself: localauth with file token (ie. nopassword). Handles local groups.

WAPT Console

- [FIX] display an explicit error message if a new host package can not be saved on the WAPT Server because of acl.
- [IMP] Process application messages when performing file hash/zip actions.
- [FIX] Fixed waptconsole copy cert to `wapt/ssl`: handle properly spaces in target directory name.
- [FIX] Place cursor at end of line instead of point of click in textareas.
- [ADD] Popup Menu with Copy and Copy as JSon for Audit TreeView.
- [FIX] Fixed proper images on actions buttons.
- [FIX] Fixed OU icon when OU name contains an empty character.
- [FIX] Fixed Out of bound error : add verification on condition check in specific cases.
- [FIX] Fixed missing error message on secondary repositories.
- [IMP] Improve usability of copying new certificate in `<WAPT>\ssl` directory
- [FIX] Fixed icon on action `ActWUAGetUnusedKB`.
- [FIX] Fixed actions caption on toolbar in Windows Update.
- [FIX] Fixed removing ability to personalize toolbuttons on ISO, configs, and drivers in *OS Deployment*.

- [FIX] Fixed popup menus on toolbar in *OS Deployment*.
- [FIX] Fixed actions on toolbar in *Software Inventory*.
- [NEW] waptconsole / waptserver: Added a specific ACL for `update_audit_data`.
- [UPD] Increasing softwares max count limit in *Software Inventory* from 5000 to 20000.
- [FIX] Fixed locking some actions on non Enterprise versions.
- [FIX] Fixed waptconsole package zip build: `CreateRecursiveZip`.
- [IMP] cleanup of HTML templates on waptservice. Removed unused js.
- [IMP] Showing icons for *target_os*.
- [FIX] Fixed waptconsole TX509Store: when intermediate certificates are provided in user *.pem* certificate file, only trust the first one.
- [FIX] Fixed waptconsole waptcrypto: implement `TX509Store.GetCertificatesChainFromFingerprint`. Fixed self signed certificates are always trusted when checking the WAPT package.
- [FIX] Fixed waptconsole: when signing packages, make sure we end with LF only (n unix style) `control` files.
- [IMP] Basic POC (Proof of Concept) for Auto Completion on Reporting Queries.
- [FIX] Fixed viewing TechPreview Features does not take care of display preferences.
- [FIX] Fixed the downloaded packages have now the chosen maturity.
- [IMP] Show `*.cmd` files in post install script selector.
- [NEW] Upload a default *json* configuration on the WAPT Server when building **waptagent.exe**. Fixed **waptsetup.exe** stuffing on the WAPT Server when uploading a *json* configuration.
- [FIX] Fixed the button Type for update package warning.
- [ADD] Confirm button before Update from the WAPT store.
- [FIX] Fixed waptconsole update from the WAPT store Introduced `StripPrefix` in `TPackageRequest` to allow searching in the repository on package name without prefix.
- [FIX] Include `min_os_version` and `max_os_version` in WAPT package identification to check which WAPT package is newest.
- [FIX] When building customized waptsetup, sometimes missing trusted certificate.
- [FIX] Fixed the copy of `wapt-get.ini` if there is no `waptconsole.ini`.
- [NEW] Menu item for restoring toolbars to default.
- [FIX] Fixed actions on toolbar in *WAPT Development* and *OS Deployment* forms.
- [FIX] Fixed removing certificates in create waptsetup [NEW] function for listing certificates from folder.
- [FIX] Fixed buttons links with actions on WSUS.
- [FIX] Fixed encoding problem for WSUS.
- [IMP] Removed GUI interface for the Update from the store action.
- [ADD] Added a warning message before updating a WAPT package.
- [ADD] Updated from the store button in private repository done.
- [IMP] Added Updated part for the Store Update Action.

- [IMP] Update from the store button (visual part).
- [FIX] Fixed regression on creating new *wuagroup* package.
- [UPD] waptconsole *build agent -> named with version*, config and hash instead of **waptagent.exe/**.
- [FIX] Fixed __pycache__ included in zipped package when built from waptconsole.
- [ADD] reporting: Added Unique save for each query.
- [FIX] Fixed SQL query editor: any query can be edited at any time, without erasing the others.
- [FIX] Fixed SQL query editor: if queries are already created and registered and have the same name, you can edit both without overwriting the other one.
- [IMP] Use system font for html viewers.
- [IMP] Allow package wizard without installer path.
- [NEW] Added "keys" mustache helper for html templates.
- [IMP] waptconsole: Do not try to ping servers before login dialog.
- [FIX] Fixed enabling build and upload if all information are set / pre configuration in case of portable app if an executable is found.
- [UPD] waptconsole Cyberwatch integration. Added Values mustache helper to format dict as list for Cyberwatch html report template. Added styled Cyberwatch example audit template.
- [IMP] Addied listening to ipv6 only if ipv6 is available.
- [FIX] Fixed waptconsole crash if custom column with empty size cell.
- [IMP] Added a warning when no DNS record is found (Remote repository).
- [FIX] Fixed call if app is currently closing (login cancelled).
- [IMP] Opening configuration by double-clicking on grid.
- [IMP] Package wizard for portable apps.
- [IMP] waptconsole, display bytes size in human readable format in grid.
- [FIX] Fixed OU options that are now available if the user is currently focusing the *OU* grid.
- [IMP] Improved asking credentials on http error 401.
- [FIX] Fixed waptconsole: random timeout error when running commands from the WAPT Console.
- [FIX] Fixed WAPT package creation for OUs (Organizational Units).
- [ADD] Link to the official documentation for the Config Package Wizard.
- [IMP] Proper restore of GUI when WindowState is maximized. Prevent flickering if starting maximized.
- [IMP] Improved warning before deleting a valid licence.
- [FIX] Fixed waptconsole regression: import packages. Check the signature even if it is disabled in remote repository settings.
- [FIX] Fixed waptconsole regression on additional private repositories listed in the repositories tab, even if not defined in `repositories` setting in `waptconsole.ini`.
- [FIX] Fixed waptconsole: private key password is not asked again if a matching key can not be found or decrypted.
- [REF] waptserver model upgrade: removed unused database migration steps.
- [UPD] waptserversetup: avoid automatic restart when installing MSVC (Microsoft Visual C++) 2022.

- [FIX] Fixed error editing same OU package in one session.
- [ADD] ACL Edit Repo on Index for secondary repos
- [FIX] Fixed missing editing ACL *Edit Repo*.
- [FIX] Fixed waptconsole access violation when checking unzipped package signature.
- [FIX] Fixed waptonsole multiple update of hosts corrupt packages depends grid display.
- [IMP] **waptself**, **wapt-get**, **waptexit**, **wapttray**: kill check threads on close, even on linux to speed up application shutdown.
- [UPD] waptconsole: lazy loading of DMPython. Removed python source scripter tab on main form. Moved to (inactive) uvispysources. Removed debug panel on main form removed unused `uvissearchpackage`. Added some euristic icons on audit and reporting grids depending on well known values (OK, ERROR etc…).
- [IMP] Improved the interpretation of checkbox states due to label description.
- [IMP] Improved search when importing queries.
- [FIX] Fixed host configuration package that are not editable right after creating them.
- [FIX] Fixed waptconsole pkcs12 export and email in X509 certificates.
- [IMP] Removed Python dependency in the WAPT Console.
- [UPD] waptconsole: Added popup menu to Json hardware treeview.
- [IMP] Improved reporting import, now select all queries by default + some code improvement
- [IMP] Improved enabling or disabling ACL by double click.
- [FIX] Fixed waptconsole: html audit templates. Bad search order.
- [FIX] Fixed waptconsole: actions categories fixes and updates. Hide unused categories from toolbars customization.
- [FIX] Fixed waptconsole: empty success message for some actions. Updated translations.
- [FIX] Fixed waptconsole get agents installers: fixed MISSING -> OK status.
- [UPD] Fixed waptconsole: Added Edit html template popup menu action.
- [FIX] Fixed no logo resizing if smaller size.
- [UPD] Load html templates for `host_overview` and `host_audit` from user's `appdata` directory if it exists, else from `wapt`.
- [REF] waptconsole: Refactored `TFrmHtmlViewer` to lookup templates either in user templates directory (`%APPDATA%waptconsoletemplates`) or in default `wapt` installation directory (`%WAPTBASEDIR%templates`).
- [UPD] waptconsole: Improved drag & drop of columns into GridHosts.
- [FIX] Fixed blocking action editing WSUS package if no Enterprise licence is active.
- [FIX] Fixed waptconsole drag & drop audit values.
- [FIX] Fixed waptconsole regression when signing *unit* package or modyfing stripped down WAPT packages.
- [IMP] waptconsole: Load AD Groups in thread.
- [FIX] Fixed waptconsole compilation without *USE_WAPTPACKAGE* flag.
- [REF] waptconsole: Introduced an interface for uwaptpackage TWaptPackage WIP: fix compilation when *USE_WAPTPACKAGE* is defined TODO: implement IX509Store
- [FIX] Fixed waptconsole: fixed host overview layout if no html template.

- [UPD] waptconsole: host details layout changes: introduced html templates based overview if `templateshost_overview.html` file exists (mustache template).
- [FIX] Fixed waptconsole *sendmessage* gui splitter.
- [IMP] waptconsole: check that downloaded waptsetup version is the same or newer than that of the WAPT Server.
- [FIX] Fixed autosearch in `ttissearch` component.
- [NEW] waptconsole: Added a popumenu *copy to clipboard as json for audit data*.
- [IMP] waptconsole: allow drag & drop of a audit *json* value subkey from value tree explorer.
- [NEW] waptconsole: displays audit history and WIP audit data explorer (treeview + html template).
- [FIX] Fixed reporting queries grid layout not saved properly.
- [UPD] GUI Vis ACL: zebra colored lines and added possibility to change user password from one button (same action like in right click on user).
- [FIX] Fixed avoiding exception if no user was selected before adding ACL rights.
- [FIX] Fixed trigger downloads when triggering updates from the WAPT Console (missing import).
- [UPD] Updated icons on windows update status for WUA.
- [FIX] Fixed waptconsole check external repository version timeout exception raised in frontend.
- [FIX] Fixed waptconsole multiserver: fixed can't trigger action on servers other than main WAPT Server.
- [FIX] Fixed waptconsole: Avoid error message of no `repo_url` for last used package template section.
- [FIX] Fixed modifying a password if old password was empty.
- [ADD] Hide / show all columns in grids.
- [NEW] new option `check_package_version` in `waptconsole.ini`.
- [UPD] waptconsole reporting: Added a quick search filtering zone for the query result.
- [FIX] Fixed wrong message when no admin rights and the WAPT Agent needs to be upgraded or is not present.
- [UPD] Host menu for upgrading hosts part.
- [REF] waptconsole multiserver: Refactored `TriggerActionOnHosts` to send multiples actions to the right servers.
- [FIX] Fixed waptconsole: use ROOT in addition to CA windows system certificates stores when building **winpe** with `verify_cert` = **True**.
- [UPD] Deleted host popup.
- [NEW] Possibility to download WAPT packages when asking hosts for updates.
- [UPD] `trigger_host_update` adding possibility to download the WAPT package after update.
- [FIX] Fixed waptconsole: The WAPT Console crashed when checking newest packages if wapt-templates repository is protected with an encrypted client key.
- [FIX] Fixed saving configuration when new configuration was created.
- [FIX] Fixed saving language parameter.
- [FIX] Fixed waptconsole: access violation when access to external repository is blocked or needs a proxy.
- [FIX] Fixed waptconsole multiserver regression: unable to edit a WAPT package which was just edited.
- [FIX] Fixed waptconsole edit conf package: Do not close if error when uploading to the WAPT Server.

- [FIX] patched `setup_package_template_cert.py.tmpl`.
- [FIX] Fixed not adding "cn" in OU.
- [FIX] Fixed layout on Windows Update part.
- [FIX] Fixed the flow layout.
- [IMP] waptconsole: WIP multiserver. Mostly works for hosts, but not for packages management.
- [FIX] Fixed waptconsole: re-enable dataexport to *.csv* for grids.
- [NEW] Explicit hint on number version when the WAPT package is not up to date (GridPackages).
- [REF] Refactored private key password handling. Added a callback to clear cached key password in case of decrypt error in http client. Stores client https authentication key password in same storage as package private keys.
- [REF] WIP for multiserver console. WaptCookieManager takes in accounts the domain. TODO: send allowed session cookies for cross domain auth. Lazy loading of waptserver instance. Loads list of servers in `DMWaptConsole.ReloadConfigFile`. All sections with a `wapt_server` key are taken in account. Shares the WaptServerSession across all waptserver connections.
- [FIX] Fixed bad port for **veyon**.

WAPT Server

- [SEC] Windows: waptserversetup.exe windows: do not reenable acl inheritance on wapt root folder.
- [SEC] Send minimal information on /ping api call.
- [IMP] Set session cookie to 3 days
- [IMP] waptserversetup: Check if there is a CRITICAL log entry during `winsetup.py` and exit with an exitcode 1000 if it is the case.
- [IMP] waptserver: Do not automatically create users in wapt database when user logs in with kerberos (self-service case).
- [FIX] Fixed waptserverinstall windows: regression unable to install on new windows machine if wapt was not already installed.
- [REF] Server python code cleanup.
- [IMP] wapttasks: use environment variable on linux to pass `config` file path.
- [NEW] waptserver: reduced lifetime of session cookie to default 12h. `session_lifetime` can be changed in `waptserver.ini` using `session_lifetime` seconds parameter.
- [UPD] Updated to python 3.8.16 for all supported operating systems.
- [FIX] Fixed stuffed setup exe naming on the WAPT Server.
- [NEW] new parameter `list_subnet_skip_login_wads`.
- [FIX] Fixed waptserver: shorten SQL columns aliases for long `get_hosts` *json* queries.
- [SEC] Upgraded **werkzeug** 2.0.2 -> 2.1.1 for PYSEC-2022-203.
- [NEW] waptservice websocket: Enabled certificate checking on websockets.
- [IMP] waptserver: Added index on `computer_ad_ou`.
- [FIX] Fixed waptserver: by default, do not create stuffed `waptsetup` when a dynamic config is uploaded.
- [FIX] Fixed waptserversetup: if installService, configure the local service to reach newly installed server. Propose to start the WAPT Console right after for demo mode.
- [NEW] `model.py`: Added `upgrade-db` action and **`--overwrite-version=1.2.3`** option to force the replay of upgrade db.

- [FIX] Fixed waptserver **nginx** config, there can be spaces in path. quotes include.
- [NEW] Be sure to not start the WAPT Server if the database structure can not be upgraded properly.
- [NEW] If licences *json* data is empty, assume an empty list.
- [IMP] Getting storage used by KBs.
- [NEW] 22H2 build numbers in WindowsVersions class.
- [NEW] Added `hosts_sid` endpoint routing to uwsgi in nginx configuration templates.
- [FIX] Fixed **wapt-get build-waptagent**: create **waptagent.exe** link on the WAPT Server.
- [FIX] Fixed waptserver: ignore null bytes in audit data string values.
- [FIX] Fixed waptserver: allow access to agent download without client certificate auth.
- [FIX] Fixed waptserver model: remove references to unused HostExtData table.
- [FIX] Fixed waptserver multiinstance with uwsgi: takes in account `application_root` for interprocess `get_ws_connections` /api/v3/hosts_sid calls.
- [UPD] Added waptserver /api/v3/update_hosts_sid_table endpoint to fill the HostWebsocket table with the in memory ws_connections for reporting purpose.
- [UPD] Changed the path of the untouched **waptsetup.exe** on the WAPT Server: moved to the `wapt/waptagent` folder to be consistent with other agents location Same for **waptdeploy.exe**.
- [DEL] waptserver: Removed "enable_store" setting.
- [UPD] waptconsole multiserver: display unreachable servers.
- [FIX] Fixed waptserversetup: Reinclude waptwua even if service is not installed to allow **wapt-get** usage.
- [FIX] Fixed waptconsole multiserver dynamic config: bad server url for checking https certificate.
- [FIX] Fixed waptconsole multiserver: Do not include a server at startup if it is not pingable.
- [UPD] waptserversetup windows: Removed some additional unused files when waptservice is not installed.
- [UPD] waptconsole multi servers: Do not try to update / merge repo if `repo_url` is empty.
- [IMP] waptserver / waptservice websockets: When registering host, return an authentication token in response, so that websockets can connect without additional roundtrip.
- [IMP] `allow_unauthenticated_registration` is now like `use_kerberos`.
- [FIX] Postconf, current config is now autoselected.
- [UPD] waptsetup waptserversetup: Sign the installers and uninstallers using embedded iscc logic.
- [UPD] waptserver db: Changed the primary key of tables *HostPackagesStatus*, *HostExtData*, *Packages*, *HostSoftwares*, *HostGroups*, *HostWebsocket*, *HostAuditData*, *ReportingSnapshots*, *HostWsus*, *LogsAPI* to bigint.
- [UPD] waptserversetup: Check that the user is a LOCAL computer user and not a domain user.
- [FIX] Fixed waptserversetup: postgresql upgrade. Try to fix ACLs on data directory.
- [FIX] Added a conflict on apache2 in the Linux WAPT Server package to avoid old install leftovers.
- [REF] Removed `enterprise_common.py`.
- [UPD] Upgrade **nginx** on Windows.
- [UPD] Upgraded DB to **postgresql v14** for windows.

- [UPG] upgraded **postgresql** 9.6 to **v14** on CentOS7.
- [FIX] Fixed waptserver: Fixed sid map sharing in uwsgi mode (missing imports).
- [FIX] Fixed waptserver websocket: Be sure to not clear a SID which would be newer than current disconnect event. Not sure if disconnect / reconnect are always synchronous.
- [FIX] Fixed waptserver: Improved message when triggering action.
- [IMP] Added HTST (HTTP Strict Transport Security) header to nginx template.
- [FIX] Fixed waptserver `update_hosts_audit_data`: Updated values with same global key (host_id,value_id).
- [FIX] Added **trigger_host_action** ACL on /api/v3/connected_wol_relays (used by /api/v3/trigger_wakeonlan).
- [IMP] waptserver websocket auth: Put host certificates in cache.
- [UPD] waptserver websocket: Do not cache UUID twice.
- [REF] waptserver websockets: use a global in memory dictionary to hold the host uuid -> SID of connected host to avoid Database insert or updates.
- [FIX] Fixed server regression for custom *json* fields ValueError: too many values to unpack (expected 3).
- [IMP] waptserver: WIP endpoint `update_hosts_audit_data` to bulk insert hosts related data.
- [IMP] waptserver: update api/v3/get_agents_info to match the online `wapt_agent_list.json`.
- [FIX] Fixed glpi sync: simplified `glpi_upload_hosts.py` script.
- [FIX] Fixed waptserver huey tasks: `licences_list` not properly initialized when not using default `waptserver.ini`.
- [FIX] Fixed waptserver audit table structure upgrade: typo
- [FIX] Fixed avoiding GET method limits on `hosts_for_wua`.
- [FIX] Fixed waptserver unable to delete some hosts when CRL is enabled be tolerant if the host certificate is not issued by this server's CA.
- [FIX] Fixed waptconsole multiserver: Computers identified by fqdn uuid are not displayed properly in the grid.
- [UPD] Remove references to **waptsetup-tis.exe** -> renamed to **waptsetup.exe**.
- [FIX] Fixed `update_server_status` with dynamic configuration.
- [IMP] Include **waptsetup.exe** in **waptserversetup.exe**.

WADS

- [FIX] Clear WADS stdout before and after **diskpart** to avoid broken stdout.
- [IMP] Check whether `winpe.wim` and `7z.exe` files exist when creating the WADS WinPE.
- [FIX] Added missing '/' in `wgetwads` error messages.
- [IMP] WADS: Added session login type and acl.
- [IMP] WADS: Login to server only one time instead of for each request.
- [IMP] WADS: Added flags: unchecked for wads login on Windows Server.
- [IMP] Use of latest mormot function for `WgetWads` to fix DNS check.
- [IMP] Improved error messages for WADS and WGETWADS.
- [IMP] Added option *wads* in Windows Server installer.

- [IMP] `get_wads_secondary_repo` –> follow protocol of the server connection.
- [FIX] Fixed `list_subnet_skip_login_wads` read configuration.
- [IMP] WinPE creation key
- [REF] Remove useless code on `get_wads_config` (Login WADS).
- [IMP] WgetWads does not require python to work.
- [FIX] Be more indulgent on *`json`* rules for WADS.
- [FIX] Fixed WADS working when no logging required.
- [ADD] Login in IPXE, more tests needed.
- [IMP] Proper way to secure `wads_get_config`.
- [ADD] Login on WADS register host and get wads configuration.
- [NEW] include hostname in `debian.ipxe` for OS deployment.
- [FIX] Fixed `djoin` with given domainuser parameter.
- [IMP] Added back support GET method on /api/v3/get_wads_config.
- [NEW] Added asset tag in HostOSDeploy.
- [IMP] Ask for a new hostname when starting to deploy if hostname equals to 'autoregister'.
- [IMP] Improved filtering keyboard faster + french translation in *Make WinPE*.
- [FIX] Fixed missing glob import in WADS `get_iso_config`.
- [NEW] Adding drivers in WinPE from WADS drivers.
- [IMP] Improved feedback when the `djoin` fails (already existing machine).
- [WADS] <Value> format in *`XML`* was incorrect and not complete for password definition.
- [IMP] Last error message added for failed `djoin`.
- [FIX] Fixed uninstall **`wapttftpserver`** when uninstalling **`waptserver`**.
- [IMP] Improved file upload with hash check wads *`iso`* files listed from the WAPT Server even if not saved in the WAPT Console.
- [NEW] Added customized WinPE export to zip file.
- [IMP] Improved showing the error message on upload failure.
- [IMP] Improved applying default configuration on wads host if no configuration has been set.
- [IMP] ISO download dialog box.
- [IMP] WADS: WinPE now pinging WAPT Server. Selected language keyboard layout will be available directly in a new cmd.
- [IMP] WADS: XML no longer disable UAC by default.
- [FIX] Fixed `mac_address` not returned with iPXE.
- [ADD] Added `ipxe_script_jinja_path` and two templates.
- [UPD] Added file type filters for loading the post-install script.
- [FIX] Restored a progression bar when uploading the *`ISO`* and the *`winpe`* files.
- [IMP] kill **`wapttftpserver`** and uninstall the service before installing it.
- [ADD] Added Windows 11 unattend *`XML`* template files.

- [IMP] Improved searching WADS data (hosts, isos, driver bundles, configurations).
- [FIX] Added tftp **firewalld** port opening.
- [IMP] Avoid creating WinPE on Windows partition + some ACL added.
- [UPD] Renamed drivers bundle filenames to sha256(filename).
- [ADD] Added a template for Debian.
- [UPD] *GridConfigDeploy* showing the platform now.
- [FIX] Fixed saving bundle names.
- [NEW] Injecting a:abbr:*OEM (Original Equipment Manufacturer)* key by **slmgr** command.
- [FIX] Fixed SELinux rules for wads.
- [FIX] Potential fix for (over 10 joins for djoin by a standard user on MSAD).
- [UPD] WADS grayed when windows update repository is selected.
- [UPD] Possibility to select an *iso* file even if not Windows.
- [FIX] Fixed waptconsole `uploadWinPE`: regression in upload progress bar and incomplete zip.
- [FIX] Fixed wads to include non CA certificates for WinPE build.
- [IMP] Added `ipxe_script` in DeployConfig table.

WAPT Agent MacOS

- [UPD] Delete old *pkg* if available in *pkg* list.
- [NEW] Added fake menu for macOS for letting user to quit the app from the *MainMenu*.
- [FIX] Improved support for macOS *MenuBar*.
- [FIX] Added WAPT Console *.app* plist file for macOS X.
- [FIX] Fixed some macOS X model and serial number reports.
- [FIX] Fixed macOS X `local_groups` key in `host_info`.
- [FIX] Updated mormot2 for **gssapi** on macOS X.
- [NEW] support WADS security, Network masks.
- [FIX] Fixed `installed_softwares` on MacOS.
- [NEW] Added timestamping to *pkg* signing.
- [FIX] Fixed getting agent version in `get_wads_config`.
- [NEW] Added entitlements file for macOS signing.
- [IMP] Force light UI when DarkMode is active on macOS.
- [FIX] Fixed opening maximized self service on macOS
- [FIX] Fixed loading hosts on macOS when *more options* in inventory is checked.
- [IMP] Better handle on input (utf8 convertion).
- [IMP] macOS: Updated build script to handle binary file signing and better debugging.
- [IMP] Patched `dmidecode` info for macOS.

- [FIX] Fixed macOS core `get_hostname` return binary string instead of str -> update_status loop.
- [IMP] Use `system_profiler_info` for `dmi_info` on macOS X.
- [REF] `plistlib.readPlistFromBytes` deprecation fix.
- [FIX] Fixed core macOS: use UUID from `system_profiler_info` instead of `dmidecode`.
- [FIX] Fixed duplicated macOS code in setuphelpers for `get_hostname()`.
- [IMP] Improved mounting content for *.pkg*, *.mpkg*, *.app* only if file is not symbolic.
- [NEW] Added the WAPT Console to Linux and macOS gui distribution.
- [IMP] Fixed keyword and name with `installed_softwares` in macOS and Linux.
- [FIX] Fixed register for macOS.
- [FIX] Fixed custom waptmessage logo linux.
- [FIX] Fixed **harakiri** on non Windows kills all running processes.
- [FIX] Fixed restart waptservice for macOS.
- [IMP] Silently attach *dmg* file.
- [FIX] Fixed `get_file_type` in macOS.

WAPT Agent Linux

- [FIX] Fixed logrotate on RedHat8 for waptserver and wapttasks.
- [IMP] **wapt-get.bin**: Improved python traceback format with proper line endings on non Windows.
- [IMP] Improve support for dark mode on WAPT Console on Linux
- [IMP] Replaced in `/usr/bin/` **wapt-get.sh** by **wapt-get.bin**.
- [IMP] Added Ubuntu and CentOS icons.
- [IMP] Added icons in *ImportPackages* window.
- [FIX] Fixed `user_local_appdata` for Linux.
- [IMP] waptagent Debian package: removed system python3 dependency.
- [IMP] Avoid loop in checkbox events on search inventory especially on operating systems other than Windows.
- [IMP] Added `PYTHONNOUSERSITE` = **True** to all *.sh* scripts to avoid spoiling PYTHONPATH with locally installed libraries in user home directory.
- [UPD] Disable compression on unix WAPT agent bundle (each package is itself already compressed).
- [NEW] Added the WAPT Console to Linux and MacOS gui distribution.
- [FIX] Fixed *configpackage* wizard and main form layouts for Linux.
- [UPD] Updated virtualtreeview for Linux visual grid lines improvements.
- [IMP] Fixed keyword and name with `installed_softwares` in macOS and Linux.
- [FIX] Fixed **harakiri** on non Windows kills all running processes.
- [ADD] Added snap software inventory.
- [FIX] Fixed waptservice linux restart Linux: AttributeError: `WaptServiceRestart` object has no attribute *logger*.

- [NEW] Linux OS deployment.
- [FIX] Added **firewalld** rule on RedHat based server for **wapttftpserver**.

46.2.6 WAPT-2.3.0.13334 RC3 (2023-01-06)

hash : a06031bd

This is the third release candidate of WAPT 2.3.

This is a release candidate for testing that is not intended for production.

This changelog lists the fixes sinces WAPT 2.3 RC2.

WAPT Core

- [SEC] When checking exe certificate, first check that the signature is OK.
- [SEC] when stuffing waptsetup.exe, check that waptsetup.exe downloaded from wapt server is properly signed by Tranquil IT.
- [FIX] Fixed handling properly utf8 chars in certificate subject.
- [FIX] Small improvement for wapt-get build-waptagent. Do not ask the server password twice.
- [FIX] Fixed stuffed legacy waptagent build: be sure to have a deterministic binary result when stuffing in waptconsole or server side.
- [IMP] remove client library dependency for command line progress bar.

WAPT Agent

- [FIX] force create random uuid if bios uuid is not correct.
- [FIX] Do not check `wsusscn2.cab` if not Enterprise.

WAPT Server

- [SEC] Windows: waptserversetup.exe windows: do not reenable acl inheritance on wapt root folder.
- [SEC] Send minimal information on /ping api call.
- [IMP] Set session cookie to 3 days

WAPT Console

- [FIX] display an explicit error message if a new host package can not be saved on the WAPT Server because of acl.
- [IMP] Process application messages when performing file hash/zip actions.
- [FIX] Fixed waptconsole copy cert to `wapt/ssl`: handle properly spaces in target directory name.
- [FIX] Place cursor at end of line instead of point of click in textareas.

WADS

- [FIX] Clear WADS stdout before and after **diskpart** to avoid broken stdout.
- [IMP] Check whether `winpe.wim` and `7z.exe` files exist when creating the WADS WinPE.
- [FIX] Added missing '/' in `wgetwads` error messages.

WAPT Linux

- [FIX] Fixed logrotate on RedHat8 for waptserver and wapttasks.
- [IMP] **wapt-get.bin**: Improved python traceback format with proper line endings on non Windows.
- [IMP] Improve support for dark mode on WAPT Console on Linux

46.2.7 WAPT-2.3.0.13301 RC2 (2023-01-04)

hash: a2af0e8d

What's New?

This is second release candidate of WAPT 2.3. This is second release candidate of WAPT 2.3.

This is a release candidate for testing that is not intended for production.

This changelog lists the fixes sinces WAPT 2.3 RC1.

Note : for security reasons in waptpython, Python isolated mode is now enabled by default (Python -I). If you are using the waptpython Python environment outside of WAPT, please be sure to check for the corresponding Python documentation.

WAPT Core

- [SEC] waptpython 3.8.16 is now compiled with the isolated mode flag at true by default (Python -I)

WAPT Console

- [ADD] Popup Menu with Copy and Copy as JSon for Audit TreeView.
- [FIX] Fixed proper images on actions buttons.
- [FIX] Fixed OU icon when OU name contains an empty character.
- [FIX] Fixed Out of bound error : add verification on condition check in specific cases.
- [FIX] Fixed missing error message on secondary repositories.
- [IMP] Improve usability of copying new certificate in `<WAPT>\ssl` directory

WAPT Agent

- [IMP] add host_capabilities inventory.
- [IMP] Better JSON format (Human Readable) for Audit Data.
- [FIX] Use parameter `IncludeCA` on `ListSOCertificatesFromFolder`.
- [FIX] Fixed translation issues in graphical components.
- [FIX] Fixed last version, checks the minimal OS version
- [FIX] edit waptwua if `install_delay` has value.

WADS

- [IMP] WADS: Added session login type and acl.
- [IMP] WADS: Login to server only one time instead of for each request.
- [IMP] WADS: Added flags: unchecked for wads login on Windows Server.
- [IMP] Use of latest mormot function for `WgetWads` to fix DNS check.
- [IMP] Improved error messages for WADS and WGETWADS.
- [IMP] Added option *wads* in Windows Server installer.
- [IMP] `get_wads_secondary_repo` –> follow protocol of the server connection.
- [FIX] Fixed `list_subnet_skip_login_wads` read configuration.
- [IMP] WinPE creation key

WAPT Linux Agent

- [IMP] Replaced in `/usr/bin/` **`wapt-get.sh`** by **`wapt-get.bin`**.
- [IMP] Added Ubuntu and CentOS icons.
- [IMP] Added icons in *ImportPackages* window.

46.2.8 WAPT-2.3.0.13239 RC1 (2022-12-21)

hash: 675d861e

What's New?

- 1000+ bugfixes
- Less issues with false positive with antivirus software when deploying a new agent: WAPT Agents do not need to be rebuilt. The WAPT Agent is based on **`waptsetup.exe`** with certificate and configuration stored in the certificate signature of the file. The signature of the file is not altered.
- WAPT Agent on Linux and macOS: improved workflow for installing and updating the WAPT Agent.
- Improved Websocket connexion. Disconnects and reconnects have be made more robust.

- Improved support on macOS.
- Improved support on Linux.
- Update of WAPT external components.

Tech Preview

- WAPT Console support on Linux (Debian and derivatives, Redhat and derivatives).
- WAPT Console support on macOS (Mojave and above).

WAPT Core

- [REF] Removed unused functions.
- [REF] Removed unused headers.
- [IMP] waptservice: fix setting loglevel for specific components do not log WS listening too often. Fixed some action's "created_by" attributes which were not not set.
- [FIX] Windows setuphelpers: missing service_list in _all__.
- [FIX] **`wapt-get`**: moved *LoadOpenSSLFromPythonLib* to get proper path for *RegWaptBaseDir* on Linux.
- [NEW] Added armhf as a valid package architecture.
- [FIX] Fixed `scan_package` issue when there are packages without `package_uuid`. Packages table was growing at each `scan_packages`.
- [IMP] **`wapt-get`**: Added some help for `build-waptagent` and `add-config` / `reset-config`/ `set-config` *-from-url*.
- [IMP] `wapt-get reset-config-from-url`: removes dynamic configs from `conf.d` too.
- [IMP] Re-include empty folders in zipped WAPT packages.
- [FIX] Update for zip empty folder entries.
- [FIX] When checking files and directories from package manifest, create empty directories from the `manifest` file if thet do not exist yet.
- [UPD] `wapt-get update-package-sources`: Implicit transparent import of all functions from `packagesdevhelpers.py`.
- [FIX] Do not audit packages with `install_status` <> 'OK'.
- [SEC] waptpackage: Cleanup removed multiple MD type. We use only sha256 now.
- [NEW] waptconsole: Stuff `waptsetup` with *`json`* config for embedding into *waptupgrade* package.
- [FIX] waptpackage signature issue if the WAPT package is created from scratch with null attributes (ex. max_os_version). If signed, these null attributes are written to control file as sempty string, this breaks the signature control. So we initialize all default signed attributes to empty string instead of null.
- [UPD] `wapt-get create-waptagent`: Use *`json`* embedded config stuffed into certificate zone of executable signature.
- [FIX] Fixed regression in python _sign_control (encoding issue).
- [UPG] Upgraded python to 3.8.16.
- [IMP] `waptutils.py` cleanup and small fix in `user_is_member_of`.
- [REF] waptserver: Cleanup code with **`pyflakes`**.

- [IMP] Allow *none* loglevel.
- [NEW] Introduced `wapt-get reset-config-from-url`.
- [FIX] Fixed json_load_file() by adding encoding option. Default is "utf-8".
- [IMP] waptguihelper: Introduced StayOnTop argument for input_dialog() and grid_dialog()
- [FIX] Fixed `wapt-get add-config-from-url` in pure Pascal. The hash is retrieved from the filename if present, or as second parameter of command line.
- [REF] wapt python core: Removed sha1 compatibility with wapt 1.3 packages signatures.
- [FIX] Shows the proper logged user after login.
- [IMP] Fallback other method for get domain in get_hostname.
- [REF] `jsonconfig` data embedded in setup exe.
- [FIX] Default value for check verify cert.
- [UPD] Introduced uwaptjsonconfig (port of json config from python to FPC).
- [UPD] **`wapt-get`**: Added a command to list the initial configs available on server (in `wapt/conf.d`).
- [UPD] waptsetuputil: Added UnzipConfigFromExe.
- [FIX] Removed global variable for PopupEnterprise, check Licensing after closing the window.
- [IMP] buildlib: Do not remove unittest from python lib when creating the build environment.
- [FIX] `remove_file()` was unable to remove symlinks.
- [FIX] wapt core: Regression on uuid retrieval from WMI. 'System_Information' key is an array.
- [NEW] wapt core: added "wapt_temp_dir" `wapt-get.ini` parameter to specify the directory wher packages are unzipped at installation (for wyse terminal).
- [REF] Introduced packagesdevhelpers python module to remove helpers useful only for "packages source update" and reduce import time of setuphelpers.
- [IMP] windows_version() now getting the correct UBR (Update Build Revision) shown with "winver" command, adding windows_version_full in hardware inventory
- [IMP] waptguihelper: help improved for grid_dialog - also, introduced an (optional) Text parameter.
- [FIX] waptpackage: trim attributes value in `control` data. ('all' was retrieved as 'all ').
- [IMP] twaptpackage: Always set architecture and priority default.
- [UPD] Refactored SSLCABundle usage.
- [FIX] Fixed waptpackage build issue when sourceroot includes the ending path separator. Fixed self service package building. Fixed version incbuild result.
- [FIX] Fixed issue with in path in zipped files created with `CreateRecursiveZip`.
- [FIX] Fixed file not found when calling `GetServerCertificate`.
- [FIX] Fixed editing zipped package inplace (hosts packages).
- [FIX] Added call to mormot2 `RegisterOpenssl` for Access violation in **`waptlicences`**.
- [IMP] Grid editor: Show which column is currently focused even if grid has not the focus.
- [IMP] Use UTC time for expiration check of ACLs.

- [UPD] wapt core: use datetime in UTC for `audit_data`.
- [IMP] wapt core: allow usage of an environment variable *waptbasedir* to specify the location of root *waptbasedir*.
- [IMP] Default grid order set to descending signature date.
- [FIX] Allow ~ character in WAPT package names (for spaces in Organizational Units packages).
- [FIX] waptcrypto: Fixed certificate filename attribute not set when loading a certificate chain.
- [UPD] Refactored `SSLCABundle` usage.
- [FIX] Fixed using particular characters in passwords.
- [FIX] Fixed waptcore: Fixed the type for dynamic configuration.
- [FIX] copytree2 `replace_at_next_reboot`.
- [REF] Moved all the dynamic json config functions into the WAPT class to take in account the actual agent settings (specially directories).
- [UPD] Created a full version 1.2.3.rev-hash into file `wapt/version-full`.

WAPT Agent

- [IMP] When uninstalling the WAPT Agent, stop the **`waptservice`** only if the service exists.
- [FIX] Popping wrong license message on new installation.
- [FIX] waptservice socketio: Force get new ws params in case of connection error like when config is updated.
- [FIX] Fixed add new rule missing import for `isenterprise`.
- [NEW] Added disk drives to host overview template.
- [IMP] Reduced size of host *`json`* inventory data. Do not send host configurations data if not changed. Do not send audit_data headers if no data. Fixed last audit data that was always sent.
- [IMP] Improved local waptservice auth feedback.
- [REF] Refactored waptservice code.
- [FIX] Enable custom CA file for websockets certificate checking.
- [FIX] Fixed WAPT Agent `websockets_verify_cert`: error reading setting from *`ini`* file. Reset socketioclient to None when connection error to force recreating the object with new TLS settings.
- [IMP] waptdeploy: Use only registry wapt_is1 install location to get the WAPT base directory.
- [IMP] waptdeploy: Do not check **`wapt-get`** working condition.
- [FIX] Fixed waptdeloy argument parsing.
- [UPD] waptsetup: Removed distribution of **`innosetup`** as it is no longer needed.
- [NEW] waptdeploy: Check that the WAPT Agent installer and **`wapt-get.exe`** are digitally signed by Tranquil IT.
- [FIX] waptdeploy wapt basedir guessing. Hardened waptdeploy RunTask.
- [FIX] Fixed `wapt-get build-waptagent`: empty configuration name.
- [ADD] Check all rules signatures before doing anything else.
- [IMP] The agent version is obtained from the *`exe`*, not the server.
- [FIX] waptsetup auto json config: should accept `waptsetup-1.2.3_<confname>_<confhash>.exe`.

- [FIX] Fixed remote WakeOnLAN.
- [IMP] waptservice: Do not include *PrinterPaperNames*, *PaperSizesSupported* and self_service_rules in inventory sent to the WAPT Server.
- [FIX] waptexit: If unable to get licences from waptservice, assume *is_enterprise* is False.
- [FIX] wapt-get: Set password callbacks after reloading config.
- [FIX] Shortened the upgrade scheduled task argument, as it is limited to 256 chars.
- [FIX] Stuffed waptsetup: Append waptwua settings to *json*.
- [FIX] waptserver socketio: Host does not register / reconnect by itself when deleted from the WAPT Server.
- [NEW] waptsetup.exe : If waptagent.exe is named, and only one config is embedded, take the first available config for the name of the configuartion to install instead of hardcoded "default".
- [IMP] waptservice: Can start right after install even if no `wapt-get.ini`.
- [NEW] Added *nopassword* to config wizard for `service_auth_type`.
- [UPD] Added `wapt-get reset-config-from-url` and `set-config-from-url` json configuration.
- [FIX] Do not delete the files if the signature has failed.
- [IMP] waptsetup: Display a summary of embedded stuffed json configurations. Removed *use dynamic configuration* task.
- [FIX] waptserver: Fixed WakeOnLAN issue when no broadcast address exists in inventory.
- [FIX] `remove_user_appx` was not initialized from setuphelpers.
- [UPD] waptsetup: ApplyJsonConfigToIniFile when a *json* configuration is stuffed instead of `conf.d` dynamic configuration.
- [IMP] waptsetup: Do not update `wapt-get.ini` when using dynamic *json* configuration.
- [UPD] waptservice socketio: Do not require connection params update / reconnection try if there is no authorization token. When `allow_unauthenticated_connect` = **True** on the WAPT Server, the WAPT agents should be able to connect without getting a token.
- [FIX] waptself: Fixed next page button unavailable on last page.
- [UPD] waptexit: Add `waptexit_disable_skip_windows_updates` parameter in `wapt-get.ini` file and commandline argument to disable the checkbox for skipping Windows Updates.
- [UPD] wapt-get: Return ExitCode <> 0 when an exception is raised Added **`ping --service`** command to check waptservice accessibility from waptsetup.
- [UPD] waptself: Display details of WAPT package on top of packages list to avoid reframes.
- [UPD] Enable `waptservice_allow_all_packages` only for *nopassword* `service_auth_type`.
- [NEW] Added a waptservice parameter `waptservice_allow_all_packages` which allow all user to install / remove all packages as if they were part of the waptselfservice group.
- [NEW] If a *json* configuration is provided in waptsetup as stuffed data in certicode certificate area, use it for initial configuration.
- [FIX] Improved error message and wait cursor when waptselfservice is starting.
- [FIX] Fixed selfservice missing common module for `self_service_rules` when using the *nopassword* argument with the WAPT Enterprise version.
- [FIX] Changed Icon for *Add Dependencies* → *Trashcan* to *Plus*.
- [IMP] User is now informed when self service can not get a token (service not started).

- [FIX] Remove double slahs in url *//Packages*.
- [NEW] Added Ubuntu22 in waptsetup package.
- [FIX] Fixed waptmessage ambiguous '-s' option (use stdout and set window size), replaced by -c for init console.
- [FIX] Fixed tasks list on host.
- [FIX] Normalized view (lowercase) in grid for *target_os* from control part.
- [FIX] Fixed execution of waptmessage in file instead of base64 (to avoid too long command line).
- [FIX] Use cached trusted signer certificates store instead of recreating it each time.
- [FIX] Fixed signed_attributes written as string list (instead of python form) and signer is the signer certificate *Common Name*.
- [IMP] use `--not-interactive` with register if the installation runs in silent mode.
- [FIX] waptservice: Do not ignore broadcast for `WaptUpdateServerStatus` to avoid the WAPT Tray sticking upon sending data to the WAPT Server.
- [FIX] Fixed unable to synchronize remote repository.
- [IMP] waptmessage: No autosize if a size is specified on the command line.
- [FIX] Fixed no hash in clipboard, added missing helper for `add-config-from-url` in wapt-get.
- [IMP] Limit access right to Administrators to log directory (in case non public stuff gets written to logs).
- [IMP] `install_scheduling` work if not in PENDING_UPDATES status.
- [FIX] Fixed waptexit compilation: Removed specific `WaptIniFilename` function.
- [FIX] Fixed waptmessage unable to load sqlite.
- [IMP] Updated waptwua status to 'NEED-SCAN' on hosts when `download_wsusscan` is triggered and `wsusscn2.cab` file is downloaded.
- [NEW] wapt core: Added `as_dict` and descending parameters to `Wapt.read_audit_data_set`.
- [IMP] Do not take care anymore of maturity for version when it is compared to WAPT store version.
- [FIX] Fixed configuration package template `setup_package_template_conf.py`.
- [FIX] Fixed waptservice configuration: Set the `configs_dir` relative to `wapt-get.ini` full path.
- [FIX] Fixed waptservice 'start_waptexit' with arguments Faulty arguments boolean value decoding.
- [FIX] Fixed bad arguments sent to waptservice triggering upgrades with `only_priorities` and `only_if_not_process_running`.
- [FIX] Fixed `Wapt.write_audit_data_if_changed`: Write data if previous data has expired.
- [FIX] Updated the template of dynamic *json* configuration packages to match new location and naming of *json* configuration related functions.
- [NEW] Option `include_potentially_superseded_updates` in configuration wizard.
- [FIX] Fixed waptservice: Be sure to dynamically revert to default setting when a key is removed from `wapt-get.ini`.
- [FIX] Fixed waptservice: Make sure we have a random `secret_key` for local waptservice session.
- [NEW] WAPTWUA superseded support.
- [IMP] `wapt-get edit` now opens `update_package.py` too.
- [UPD] Added a *NEED-SCAN* waptwua.status, updated when `Wapt.update()` is called.

- [FIX] Fixed waptself: Set focus on search when opening.
- [IMP] Ignore history for waptwua status.
- [FIX] Fixed **wapt-get update-package-sources**: Handle properly relative path to package sources.
- [FIX] Fixed **wapt-get update-package-sources**: use devdirupdate_package.py to call update_package() hook if this file exists. Else use setup.py.
- [IMP] wapttray: Launch external **waptself** and **waptconsole** with OpenDocument instead of windows only ShellExecuteW.
- [FIX] Workaround fix when **pyscripter** is put as editor for packages. params_vscod_list fixed when space in parameters. Reupdated description.
- [IMP] **wapt-get edit** now opens changelog.txt, VSCod* now opens control file too. **wapt-get edit** can now be run as user with VSCod* updating wapt_sources_edit() description.
- [UPD] Changed default log path to wapt/log if writable.
- [UPD] Same logging initialization code for all UI executables with waptcommon.InitLoggingFromCommandLine.
- [IMP] waptservice waptself: localauth with file token (ie. nopassword). Handles local groups.

WAPT Console

- [FIX] Fixed icon on action ActWUAGetUnusedKB.
- [FIX] Fixed actions caption on toolbar in Windows Update.
- [FIX] Fixed removing ability to personalize toolbuttons on ISO, configs, and drivers in *OS Deployment*.
- [FIX] Fixed popup menus on toolbar in *OS Deployment*.
- [FIX] Fixed actions on toolbar in *Software Inventory*.
- [NEW] waptconsole / waptserver: Added a specific ACL for update_audit_data.
- [UPD] Increasing softwares max count limit in *Software Inventory* from 5000 to 20000.
- [FIX] Fixed locking some actions on non Enterprise versions.
- [FIX] Fixed waptconsole package zip build: CreateRecursiveZip.
- [IMP] cleanup of HTML templates on waptservice. Removed unused js.
- [IMP] Showing icons for *target_os*.
- [FIX] Fixed waptconsole TX509Store: when intermediate certificates are provided in user *.pem* certificate file, only trust the first one.
- [FIX] Fixed waptconsole waptcrypto: implement TX509Store.GetCertificatesChainFromFingerprint. Fixed self signed certificates are always trusted when checking the WAPT package.
- [FIX] Fixed waptconsole: when signing packages, make sure we end with LF only (n unix style) control files.
- [IMP] Basic POC for Auto Completion on Reporting Queries.
- [FIX] Fixed viewing TechPreview Features does not take care of display preferences.
- [FIX] Fixed the downloaded packages have now the chosen maturity.
- [IMP] Show *.cmd files in post install script selector.
- [NEW] Upload a default *json* configuration on the WAPT Server when building **waptagent.exe**. Fixed **waptsetup.exe** stuffing on the WAPT Server when uploading a *json* configuration.

- [FIX] Fixed the button Type for update package warning.
- [ADD] Confirm button before Update from the WAPT store.
- [FIX] Fixed waptconsole update from the WAPT store Introduced `StripPrefix` in `TPackageRequest` to allow searching in the repository on package name without prefix.
- [FIX] Include `min_os_version` and `max_os_version` in WAPT package identification to check which WAPT package is newest.
- [FIX] When building customized waptsetup, sometimes missing trusted certificate.
- [FIX] Fixed the copy of `wapt-get.ini` if there is no `waptconsole.ini`.
- [NEW] Menu item for restoring toolbars to default.
- [FIX] Fixed actions on toolbar in *WAPT Development* and *OS Deployment* forms.
- [FIX] Fixed removing certificates in create waptsetup [NEW] function for listing certificates from folder.
- [FIX] Fixed buttons links with actions on WSUS.
- [FIX] Fixed encoding problem for WSUS.
- [IMP] Removed GUI interface for the Update from the store action.
- [ADD] Added a warning message before updating a WAPT package.
- [ADD] Updated from the store button in private repository done.
- [IMP] Added Updated part for the Store Update Action.
- [IMP] Update from the store button (visual part).
- [FIX] Fixed regression on creating new *wuagroup* package.
- [UPD] waptconsole *build agent -> named with version*, config and hash instead of **`waptagent.exe/`**.
- [FIX] Fixed __pycache__ included in zipped package when built from waptconsole.
- [ADD] reporting: Added Unique save for each query.
- [FIX] Fixed SQL query editor: any query can be edited at any time, without erasing the others.
- [FIX] Fixed SQL query editor: if queries are already created and registered and have the same name, you can edit both without overwriting the other one.
- [IMP] Use system font for html viewers.
- [IMP] Allow package wizard without installer path.
- [NEW] Added “keys” mustache helper for html templates.
- [IMP] waptconsole: Do not try to ping servers before login dialog.
- [FIX] Fixed enabling build and upload if all information are set / pre configuration in case of portable app if an executable is found.
- [UPD] waptconsole Cyberwatch integration. Added Values mustache helper to format dict as list for Cyberwatch html report template. Added styled Cyberwatch example audit template.
- [IMP] Addied listening to ipv6 only if ipv6 is available.
- [FIX] Fixed waptconsole crash if custom column with empty size cell.
- [IMP] Added a warning when no DNS record is found (Remote repository).

- [FIX] Fixed call if app is currently closing (login cancelled).
- [IMP] Opening configuration by double-clicking on grid.
- [IMP] Package wizard for portable apps.
- [IMP] waptconsole, display bytes size in human readable format in grid.
- [FIX] Fixed OU options that are now available if the user is currently focusing the *OU* grid.
- [IMP] Improved asking credentials on http error 401.
- [FIX] Fixed waptconsole: random timeout error when running commands from the WAPT Console.
- [FIX] Fixed WAPT package creation for OUs.
- [ADD] Link to the official documentation for the Config Package Wizard.
- [IMP] Proper restore of GUI when WindowState is maximized. Prevent flickering if starting maximized.
- [IMP] Improved warning before deleting a valid licence.
- [FIX] Fixed waptconsole regression: import packages. Check the signature even if it is disabled in remote repository settings.
- [FIX] Fixed waptconsole regression on additional private repositories listed in the repositories tab, even if not defined in `repositories` setting in `waptconsole.ini`.
- [FIX] Fixed waptconsole: private key password is not asked again if a matching key can not be found or decrypted.
- [REF] waptserver model upgrade: removed unused database migration steps.
- [UPD] waptserversetup: avoid automatic restart when installing MSVC 2022.
- [FIX] Fixed error editing same OU package in one session.
- [ADD] ACL Edit Repo on Index for secondary repos
- [FIX] Fixed missing editing ACL *Edit Repo*.
- [FIX] Fixed waptconsole access violation when checking unzipped package signature.
- [FIX] Fixed waptonsole multiple update of hosts corrupt packages depends grid display.
- [IMP] **`waptself`**, **`wapt-get`**, **`waptexit`**, **`wapttray`**: kill check threads on close, even on linux to speed up application shutdown.
- [UPD] waptconsole: lazy loading of DMPython. Removed python source scripter tab on main form. Moved to (inactive) uvispysources. Removed debug panel on main form removed unused `uvissearchpackage`. Added some euristic icons on audit and reporting grids depending on well known values (OK, ERROR etc…).
- [IMP] Improved the interpretation of checkbox states due to label description.
- [IMP] Improved search when importing queries.
- [FIX] Fixed host configuration package that are not editable right after creating them.
- [FIX] Fixed waptconsole pkcs12 export and email in X509 certificates.
- [IMP] Removed Python dependency in the WAPT Console.
- [UPD] waptconsole: Added popup menu to Json hardware treeview.
- [IMP] Improved reporting import, now select all queries by default + some code improvement
- [IMP] Improved enabling or disabling ACL by double click.
- [FIX] Fixed waptconsole: html audit templates. Bad search order.

- [FIX] Fixed waptconsole: actions categories fixes and updates. Hide unused categories from toolbars customization.
- [FIX] Fixed waptconsole: empty success message for some actions. Updated translations.
- [FIX] Fixed waptconsole get agents installers: fixed MISSING -> OK status.
- [UPD] Fixed waptconsole: Added Edit html template popup menu action.
- [FIX] Fixed no logo resizing if smaller size.
- [UPD] Load html templates for `host_overview` and `host_audit` from user's `appdata` directory if it exists, else from `wapt`.
- [REF] waptconsole: Refactored `TFrmHtmlViewer` to lookup templates either in user templates directory (`%APPDATA%waptconsoletemplates`) or in default `wapt` installation directory (`%WAPTBASEDIR%templates`).
- [UPD] waptconsole: Improved drag & drop of columns into GridHosts.
- [FIX] Fixed blocking action editing WSUS package if no Enterprise licence is active.
- [FIX] Fixed waptconsole drag & drop audit values.
- [FIX] Fixed waptconsole regression when signing *unit* package or modyfing stripped down WAPT packages.
- [IMP] waptconsole: Load AD Groups in thread.
- [FIX] Fixed waptconsole compilation without *USE_WAPTPACKAGE* flag.
- [REF] waptconsole: Introduced an interface for uwaptpackage TWaptPackage WIP: fix compilation when *USE_WAPTPACKAGE* is defined TODO: implement IX509Store
- [FIX] Fixed waptconsole: fixed host overview layout if no html template.
- [UPD] waptconsole: host details layout changes: introduced html templates based overview if `templateshost_overview.html` file exists (mustache template).
- [FIX] Fixed waptconsole *sendmessage* gui splitter.
- [IMP] waptconsole: check that downloaded waptsetup version is the same or newer than that of the WAPT Server.
- [FIX] Fixed autosearch in `ttissearch` component.
- [NEW] waptconsole: Added a popumenu *copy to clipboard as json for audit data*.
- [IMP] waptconsole: allow drag & drop of a audit *`json`* value subkey from value tree explorer.
- [NEW] waptconsole: displays audit history and WIP audit data explorer (treeview + html template).
- [FIX] Fixed reporting queries grid layout not saved properly.
- [UPD] GUI Vis ACL: zebra colored lines and added possibility to change user password from one button (same action like in right click on user).
- [FIX] Fixed avoiding exception if no user was selected before adding ACL rights.
- [FIX] Fixed trigger downloads when triggering updates from the WAPT Console (missing import).
- [UPD] Updated icons on windows update status for WUA.
- [FIX] Fixed waptconsole check external repository version timeout exception raised in frontend.
- [FIX] Fixed waptconsole multiserver: fixed can't trigger action on servers other than main WAPT Server.
- [FIX] Fixed waptconsole: Avoid error message of no `repo_url` for last used package template section.
- [FIX] Fixed modifying a password if old password was empty.
- [ADD] Hide / show all columns in grids.

- [NEW] new option `check_package_version` in `waptconsole.ini`.
- [UPD] waptconsole reporting: Added a quick search filtering zone for the query result.
- [FIX] Fixed wrong message when no admin rights and the WAPT Agent needs to be upgraded or is not present.
- [UPD] Host menu for upgrading hosts part.
- [REF] waptconsole multiserver: Refactored `TriggerActionOnHosts` to send multiples actions to the right servers.
- [FIX] Fixed waptconsole: use ROOT in addition to CA windows system certificates stores when building **winpe** with `verify_cert` = **True**.
- [UPD] Deleted host popup.
- [NEW] Possibility to download WAPT packages when asking hosts for updates.
- [UPD] `trigger_host_update` adding possibility to download the WAPT package after update.
- [FIX] Fixed waptconsole: The WAPT Console crashed when checking newest packages if wapt-templates repository is protected with an encrypted client key.
- [FIX] Fixed saving configuration when new configuration was created.
- [FIX] Fixed saving language parameter.
- [FIX] Fixed waptconsole: access violation when access to external repository is blocked or needs a proxy.
- [FIX] Fixed waptconsole multiserver regression: unable to edit a WAPT package which was just edited.
- [FIX] Fixed waptconsole edit conf package: Do not close if error when uploading to the WAPT Server.
- [FIX] patched `setup_package_template_cert.py.tmpl`.
- [FIX] Fixed not adding "cn" in OU.
- [FIX] Fixed layout on Windows Update part.
- [FIX] Fixed the flow layout.
- [IMP] waptconsole: WIP multiserver. Mostly works for hosts, but not for packages management.
- [FIX] Fixed waptconsole: re-enable dataexport to *.csv* for grids.
- [NEW] Explicit hint on number version when the WAPT package is not up to date (GridPackages).
- [REF] Refactored private key password handling. Added a callback to clear cached key password in case of decrypt error in http client. Stores client https authentication key password in same storage as package private keys.
- [REF] WIP for multiserver console. WaptCookieManager takes in accounts the domain. TODO: send allowed session cookies for cross domain auth. Lazy loading of waptserver instance. Loads list of servers in `DMWaptConsole.ReloadConfigFile`. All sections with a `wapt_server` key are taken in account. Shares the WaptServerSession across all waptserver connections.
- [FIX] Fixed bad port for **veyon**.

WAPT Server

- [IMP] waptserversetup: Check if there is a CRITICAL log entry during `winsetup.py` and exit with an exitcode 1000 if it is the case.
- [IMP] waptserver: Do not automatically create users in wapt database when user logs in with kerberos (self-service case).
- [FIX] Fixed waptserverinstall windows: regression unable to install on new windows machine if wapt was not already installed.
- [REF] Server python code cleanup.
- [IMP] wapttasks: use environment variable on linux to pass `config` file path.
- [NEW] waptserver: reduced lifetime of session cookie to default 12h. `session_lifetime` can be changed in `waptserver.ini` using `session_lifetime` seconds parameter.
- [UPD] Updated to python 3.8.16 for all supported operating systems.
- [FIX] Fixed stuffed setup exe naming on the WAPT Server.
- [NEW] new parameter `list_subnet_skip_login_wads`.
- [FIX] Fixed waptserver: shorten SQL columns aliases for long `get_hosts` *`json`* queries.
- [SEC] Upgraded **`werkzeug`** 2.0.2 -> 2.1.1 for PYSEC-2022-203.
- [NEW] waptservice websocket: Enabled certificate checking on websockets.
- [IMP] waptserver: Added index on `computer_ad_ou`.
- [FIX] Fixed waptserver: by default, do not create stuffed `waptsetup` when a dynamic config is uploaded.
- [FIX] Fixed waptserversetup: if installService, configure the local service to reach newly installed server. Propose to start the WAPT Console right after for demo mode.
- [NEW] `model.py`: Added `upgrade-db` action and **`--overwrite-version=1.2.3`** option to force the replay of upgrade db.
- [FIX] Fixed waptserver **`nginx`** config, there can be spaces in path. quotes include.
- [NEW] Be sure to not start the WAPT Server if the database structure can not be upgraded properly.
- [NEW] If licences *`json`* data is empty, assume an empty list.
- [IMP] Getting storage used by KBs.
- [NEW] 22H2 build numbers in WindowsVersions class.
- [NEW] Added `hosts_sid` endpoint routing to uwsgi in nginx configuration templates.
- [FIX] Fixed **`wapt-get build-waptagent`**: create **`waptagent.exe`** link on the WAPT Server.
- [FIX] Fixed waptserver: ignore null bytes in audit data string values.
- [FIX] Fixed waptserver: allow access to agent download without client certificate auth.
- [FIX] Fixed waptserver model: remove references to unused HostExtData table.
- [FIX] Fixed waptserver multiinstance with uwsgi: takes in account `application_root` for interprocess `get_ws_connections` /api/v3/hosts_sid calls.
- [UPD] Added waptserver /api/v3/update_hosts_sid_table endpoint to fill the HostWebsocket table with the in memory ws_connections for reporting purpose.
- [UPD] Changed the path of the untouched **`waptsetup.exe`** on the WAPT Server: moved to the `wapt/waptagent` folder to be consistent with other agents location Same for **`waptdeploy.exe`**.

- [DEL] waptserver: Removed "enable_store" setting.
- [UPD] waptconsole multiserver: display unreachable servers.
- [FIX] Fixed waptserversetup: Reinclude waptwua even if service is not installed to allow **wapt-get** usage.
- [FIX] Fixed waptconsole multiserver dynamic config: bad server url for checking https certificate.
- [FIX] Fixed waptconsole multiserver: Do not include a server at startup if it is not pingable.
- [UPD] waptserversetup windows: Removed some additional unused files when waptservice is not installed.
- [UPD] waptconsole multi servers: Do not try to update / merge repo if `repo_url` is empty.
- [IMP] waptserver / waptservice websockets: When registering host, return an authentication token in response, so that websockets can connect without additional roundtrip.
- [IMP] `allow_unauthenticated_registration` is now like `use_kerberos`.
- [FIX] Postconf, current config is now autoselected.
- [UPD] waptsetup waptserversetup: Sign the installers and uninstallers using embedded iscc logic.
- [UPD] waptserver db: Changed the primary key of tables *HostPackagesStatus*, *HostExtData*, *Packages*, *HostSoftwares*, *HostGroups*, *HostWebsocket*, *HostAuditData*, *ReportingSnapshots*, *HostWsus*, *LogsAPI* to bigint.
- [UPD] waptserversetup: Check that the user is a LOCAL computer user and not a domain user.
- [FIX] Fixed waptserversetup: postgresql upgrade. Try to fix ACLs on data directory.
- [FIX] Added a conflict on apache2 in the Linux WAPT Server package to avoid old install leftovers.
- [REF] Removed `enterprise_common.py`.
- [UPD] Upgrade **nginx** on Windows.
- [UPD] Upgraded DB to **postgresql v14** for windows.
- [UPG] upgraded **postgresql** 9.6 to **v14** on CentOS7.
- [FIX] Fixed waptserver: Fixed sid map sharing in uwsgi mode (missing imports).
- [FIX] Fixed waptserver websocket: Be sure to not clear a SID which would be newer than current disconnect event. Not sure if disconnect / reconnect are always synchronous.
- [FIX] Fixed waptserver: Improved message when triggering action.
- [IMP] Added HTST header to nginx template.
- [FIX] Fixed waptserver `update_hosts_audit_data`: Updated values with same global key (host_id,value_id).
- [FIX] Added **trigger_host_action** ACL on /api/v3/connected_wol_relays (used by /api/v3/trigger_wakeonlan).
- [IMP] waptserver websocket auth: Put host certificates in cache.
- [UPD] waptserver websocket: Do not cache UUID twice.
- [REF] waptserver websockets: use a global in memory dictionary to hold the host uuid -> SID of connected host to avoid Database insert or updates.
- [FIX] Fixed server regression for custom *json* fields ValueError: too many values to unpack (expected 3).
- [IMP] waptserver: WIP endpoint `update_hosts_audit_data` to bulk insert hosts related data.
- [IMP] waptserver: update api/v3/get_agents_info to match the online `wapt_agent_list.json`.
- [FIX] Fixed glpi sync: simplified `glpi_upload_hosts.py` script.

- [FIX] Fixed waptserver huey tasks: `licences_list` not properly initialized when not using default `waptserver.ini`.
- [FIX] Fixed waptserver audit table structure upgrade: typo
- [FIX] Fixed avoiding GET method limits on `hosts_for_wua`.
- [FIX] Fixed waptserver unable to delete some hosts when CRL is enabled be tolerant if the host certificate is not issued by this server's CA.
- [FIX] Fixed waptconsole multiserver: Computers identified by fqdn uuid are not displayed properly in the grid.
- [UPD] Remove references to **`waptsetup-tis.exe`** -> renamed to **`waptsetup.exe`**.
- [FIX] Fixed `update_server_status` with dynamic configuration.
- [IMP] Include **`waptsetup.exe`** in **`waptserversetup.exe`**.

WADS

- [REF] Remove useless code on `get_wads_config` (Login WADS).
- [IMP] WgetWads does not require python to work.
- [FIX] Be more indulgent on *`json`* rules for WADS.
- [FIX] Fixed WADS working when no logging required.
- [ADD] Login in IPXE, more tests needed.
- [IMP] Proper way to secure `wads_get_config`.
- [ADD] Login on WADS register host and get wads configuration.
- [NEW] include hostname in `debian.ipxe` for OS deployment.
- [FIX] Fixed `djoin` with given domainuser parameter.
- [IMP] Added back support GET method on /api/v3/get_wads_config.
- [NEW] Added asset tag in HostOSDeploy.
- [IMP] Ask for a new hostname when starting to deploy if hostname equals to 'autoregister'.
- [IMP] Improved filtering keyboard faster + french translation in *Make WinPE*.
- [FIX] Fixed missing glob import in WADS `get_iso_config`.
- [NEW] Adding drivers in WinPE from WADS drivers.
- [IMP] Improved feedback when the `djoin` fails (already existing machine).
- [WADS] <Value> format in *`XML`* was incorrect and not complete for password definition.
- [IMP] Last error message added for failed `djoin`.
- [FIX] Fixed uninstall **`wapttftpserver`** when uninstalling **`waptserver`**.
- [IMP] Improved file upload with hash check wads *`iso`* files listed from the WAPT Server even if not saved in the WAPT Console.
- [NEW] Added customized WinPE export to zip file.
- [IMP] Improved showing the error message on upload failure.
- [IMP] Improved applying default configuration on wads host if no configuration has been set.
- [IMP] ISO download dialog box.

- [IMP] WADS: WinPE now pinging WAPT Server. Selected language keyboard layout will be available directly in a new cmd.
- [IMP] WADS: XML no longer disable UAC by default.
- [FIX] Fixed `mac_address` not returned with iPXE.
- [ADD] Added `ipxe_script_jinja_path` and two templates.
- [UPD] Added file type filters for loading the post-install script.
- [FIX] Restored a progression bar when uploading the *ISO* and the *winpe* files.
- [IMP] kill **wapttftpserver** and uninstall the service before installing it.
- [ADD] Added Windows 11 unattend *XML* template files.
- [IMP] Improved searching WADS data (hosts, isos, driver bundles, configurations).
- [FIX] Added tftp **firewalld** port opening.
- [IMP] Avoid creating WinPE on Windows partition + some ACL added.
- [UPD] Renamed drivers bundle filenames to sha256(filename).
- [ADD] Added a template for Debian.
- [UPD] *GridConfigDeploy* showing the platform now.
- [FIX] Fixed saving bundle names.
- [NEW] Injecting a:abbr:*OEM (Original Equipment Manufacturer)* key by **slmgr** command.
- [FIX] Fixed SELinux rules for wads.
- [FIX] Potential fix for (over 10 joins for djoin by a standard user on MSAD).
- [UPD] WADS grayed when windows update repository is selected.
- [UPD] Possibility to select an *iso* file even if not Windows.
- [FIX] Fixed waptconsole `uploadWinPE`: regression in upload progress bar and incomplete zip.
- [FIX] Fixed wads to include non CA certificates for WinPE build.
- [IMP] Added `ipxe_script` in DeployConfig table.

WAPT Agent MacOS

- [UPD] Delete old *pkg* if available in *pkg* list.
- [NEW] Added fake menu for macOS for letting user to quit the app from the *MainMenu*.
- [FIX] Improved support for macOS *MenuBar*.
- [FIX] Added WAPT Console *.app* plist file for macOS X.
- [FIX] Fixed some macOS X model and serial number reports.
- [FIX] Fixed macOS X `local_groups` key in `host_info`.
- [FIX] Updated mormot2 for **gssapi** on macOS X.
- [NEW] support WADS security, Network masks.
- [FIX] Fixed `installed_softwares` on MacOS.
- [NEW] Added timestamping to *pkg* signing.

- [FIX] Fixed getting agent version in `get_wads_config`.
- [NEW] Added entitlements file for macOS signing.
- [IMP] Force light UI when DarkMode is active on macOS.
- [FIX] Fixed opening maximized self service on macOS
- [FIX] Fixed loading hosts on macOS when *more options* in inventory is checked.
- [IMP] Better handle on input (utf8 convertion).
- [IMP] macOS: Updated build script to handle binary file signing and better debugging.
- [IMP] Patched `dmidecode` info for macOS.
- [FIX] Fixed macOS core `get_hostname` return binary string instead of str -> update_status loop.
- [IMP] Use `system_profiler_info` for `dmi_info` on macOS X.
- [REF] `plistlib.readPlistFromBytes` deprecation fix.
- [FIX] Fixed core macOS: use UUID from `system_profiler_info` instead of `dmidecode`.
- [FIX] Fixed duplicated macOS code in setuphelpers for `get_hostname()`.
- [IMP] Improved mounting content for *`.pkg`*, *`.mpkg`*, *`.app`* only if file is not symbolic.
- [NEW] Added the WAPT Console to Linux and macOS gui distribution.
- [IMP] Fixed keyword and name with `installed_softwares` in macOS and Linux.
- [FIX] Fixed register for macOS.
- [FIX] Fixed custom waptmessage logo linux.
- [FIX] Fixed **`harakiri`** on non Windows kills all running processes.
- [FIX] Fixed restart waptservice for macOS.
- [IMP] Silently attach *`dmg`* file.
- [FIX] Fixed `get_file_type` in macOS.

WAPT Agent Linux

- [FIX] Fixed `user_local_appdata` for Linux.
- [IMP] waptagent Debian package: removed system python3 dependency.
- [IMP] Avoid loop in checkbox events on search inventory especially on operating systems other than Windows.
- [IMP] Added `PYTHONNOUSERSITE` = **True** to all *`.sh`* scripts to avoid spoiling PYTHONPATH with locally installed libraries in user home directory.
- [UPD] Disable compression on unix WAPT agent bundle (each package is itself already compressed).
- [NEW] Added the WAPT Console to Linux and MacOS gui distribution.
- [FIX] Fixed *configpackage* wizard and main form layouts for Linux.
- [UPD] Updated virtualtreeview for Linux visual grid lines improvements.
- [IMP] Fixed keyword and name with `installed_softwares` in macOS and Linux.
- [FIX] Fixed **`harakiri`** on non Windows kills all running processes.

- [ADD] Added snap software inventory.
- [FIX] Fixed waptservice linux restart Linux: AttributeError: `WaptServiceRestart` object has no attribute *logger*.
- [NEW] Linux OS deployment.
- [FIX] Added **firewalld** rule on RedHat based server for **wapttftpserver**.

46.3 WAPT-2.2 Serie

46.3.1 WAPT-2.2.3.12481 (2022-11-30)

hash: ad3855c9

This is a security release with a few related bugfixes. All WAPT 2.0 versions below 2.2.3.12481 are affected.

Note: if you are using WAPTAgent deployment via GPO, do not forget to update your waptdeploy binary in the definition of the GPO.

WAPT Core

- [SEC] Upgraded **python** from 3.8.13 to 3.8.15.
- [SEC] Upgraded **openssl** from 1.1.1k to 1.1.1s.
- [SEC] Upgraded WAPT Agent kerberos lib from 1.19.3 to 1.20.1 (Linux / macOS).
- [SEC] Upgraded python modules with CVEs:
  - pylint==2.12.2 -> 2.15.6.
  - ujson==4.0.2 -> 5.5.0.
  - waitress==2.0.0 -> 2.1.2.

WAPT Agent

- **[SEC] waptdeploy.exe: Use only wapt_is1 install location from registry to get the current wapt installation directory.** Do not run **wapt-get** to check working condition.
- [FIX] Added fallback method to get domain in `get_hostname`.
- [FIX] Fixed windows, replaced **wapt-get.exe --hide** by **waptpythonw.exe wapt-get.py** to run **session-setup** because **--hide** does not actually hide the shell window.
- [FIX] Fixed WakeOnLAN relays.
- [REF] Cleaned up the WAPT Agent `common.py`: removed unused imports.
- [FIX] Fixed waptexit: fix `only_priorities` argument when starting waptexit from service.
- [IMP] MacOS: Updated build script to handle binary file signing and better debugging.

WAPT Console

- [UPD] WADS: Include hostname in template iPXE Debian Linux.
- [IMP] WAPT Console: Do not display empty confirmation messagebox.

WAPT Server

- [FIX] waptserver postconf: Force path when running **psql** command in postconf (linux).

46.3.2 WAPT-2.2.3.12463 (2022-09-29)

hash: fc306143

This release is mainly a bugfix release. The main new feature is tech-preview support for MacOS on Apple M1 architecture.

Note :

- due to EOL and security issue, the PostgreSQL database version has been updated on the WAPT Server for Windows and Redhat7 from version PostgreSQL 9.6.24 to PostgreSQL 14.5. The upgrade will be automatic on Windows during waptserversetup.exe install, and is done during postconf.sh run on Redhat7. Be sure to run the postconf.sh script after upgrading.

WAPT Server

- [UPD] WAPT Server for Redhat7 / Centos7: Upgraded **PostgreSQL** version from 9.6 to 14.5.
- [UPD] WAPT Server for Windows: Upgraded **nginx** to 1.22.0.
- [UPD] WAPT Server for Windows: Upgraded **vcredist** to 2022.
- [UPD] WAPT Server for Windows: Upgraded **PostgreSQL** version from 9.6 to 14.5.
- [FIX] WAPT Server for Windows: Fixed **icacls** for `migrate_pg_db`.
- [FIX] WAPT Server for Windows: Allow install and upgrade with any server admins (does not require to use the local Administrator with RID -500 for installing).
- [UPD] WAPT Server for Windows: waptserversetup: avoid automatic restart when installing MSVC 2022.
- [FIX] Fixed upgrade procedure: migrate data text to *jsonb* only if table hostauditdata in `data_type` text.
- [FIX] Patched `create_default_users` when upgrading from 1.8.2 to 2.2.
- [FIX] Fixed unhandled redirections in TWaptServer wget.
- [FIX] Added `RedirectMax` parameter in WaptServer WGet
- [UPD] Added ubuntu 22.04 in waptagent bundle.
- [FIX] Fixed postconf nginx: bad error string format.

WAPT Console

- [FIX] Fixed host configuration package that were not editable right after creating them.
- [FIX] Fixed error editing same OU package in one session.
- [FIX] Fixed `CleanupPackagesCache` proper unlock even if no assigned package.
- [FIX] Fixed access violation at startup when no server is defined in `waptconsole.ini` file.
- [FIX] Fixed waptconsole: When deleting a package in the *private repo* page, package is still listed until the WAPT Console is restarted, but the package is actually deleted on the WAPT Server.
- [FIX] Fixed waptconsole: Random timeout error when running commands from waptconsole

WAPT Agent

- [FIX] Fixed setuphelpers: reintroduce `running_as_system` for Linux and macOS (uid==0).
- [FIX] Fixed start waptservice only if `wapt-get.ini` configuration exists.
- [FIX] Fixed `remove_file()`: Was unable to remove symlinks.
- [FIX] Reset properly Wapt core settings to default when reloading config from `wapt-get.ini`.
- [FIX] Try to create a minimal `wapt-get.ini` file if it does not exist so that the service can be started without any prior configuration.
- [FIX] Fixed WAPT Agent for macOS: use `system_profiler_info` for `dmi_info` on macOS for support for Apple m1 architecture.
- [FIX] Fixed WAPT Agent for macOS: `plistlib.readPlistFromBytes` deprecation fix.
- [FIX] Fixed WAPT Agent for macOS: core macOS: use UUID from `system_profiler_info` instead of `dmidecode`.
- [FIX] Fixed WAPT Agent for macOS: change postinst script for `launchctl` compatibility.
- [FIX] Fixed WAPT Agent for macOS: macOS core: `get_hostname` returned binary string instead of str -> `update_status` loop.
- [IMP] Fixed WAPT Agent for macOS: Rationalize *pkg* filename.

46.3.3 WAPT-2.2.3.12454-rc2 (2022-09-26)

hash: 64bfc946

This is the second release candidate for WAPT 2.2.3.

The main new feature is tech-preview support for MacOS on Apple M1 architecture. Otherwise it is mainly a bugfix release.

Note :

- due to EOL and security issue, PostgreSQL database version has been updated on WAPT Server for Windows and RedHat7 from version PostgreSQL 9.6.24 to PostgreSQL 14.5. Upgrade will be automatic on Windows during waptserversetup.exe install, and is done during postconf.sh run on Redhat7. Be sure to run the postconf.sh script after upgrade.

Fixes since WAPT-2.2.3-rc1:

WAPT Server for Windows

- [FIX] Fixed **icacls** for `migrate_pg_db`.

WAPT Agent

- [FIX] Start waptservice only if wapt-get.ini config is exists
- [FIX] Added `PYTHONNOUSERSITE` = **True** to all *`.sh`* scripts to avoid spoiling PYTHONPATH with locally installed libraries in user home directory.
- [FIX] Fixed `remove_file()` that was unable to remove symlinks.
- [FIX] Fixed waptconsole : fix AV at startup when no server is defined in *`ini`* file.

WAPT Agent for macOS

- [FIX] Use `system_profiler_info` for `dmi_info` on macOS for support for Apple m1 architecture.
- [FIX] Fixed `plistlib.readPlistFromBytes` deprecation.
- [FIX] Fixed core macOS: use uuid from system_profiler_info instead of dmidecode
- [FIX] change postinst script for launchctl compatibility
- [FIX] macOS core get_hostname return binary string instead of str -> update_status loop
- [IMP] rationalize pkg filename

46.3.4 WAPT-2.2.3.12411-rc1 (2022-09-05)

hash: 29e18f23

This is mainly a bugfix release.

Note :

- due to EOL and security issue, PostgreSQL database version has been updated on WAPT Server for Windows and Redhat7 from version PostgreSQL 9.6.24 to PostgreSQL 14.5. Upgrade will be automatic on Windows during waptserversetup.exe install, and is done during postconf.sh run on Redhat7. Be sure to run the postconf.sh script after upgrade.

WAPT Server

- [UPD] WAPT Server for Redhat7 / Centos7 ! upgrade PostgreSQL version from 9.6 to 14.5
- [UPD] WAPT Server for Windows : upgrade nginx to 1.22.0
- [UPD] WAPT Server for Windows : upgrade vcredist to 2022
- [UPD] WAPT Server for Windows : upgrade PostgreSQL version from 9.6 to 14.5
- [FIX] WAPT Server for Windows : allow install and upgrade with any server admins (does not require to use the local Administrator with RID -500 for install)
- [UPD] WAPT Server for Windows : waptserversetup: avoid automatic restart when installing MSVC 2022
- [FIX] fix upgrade procedure : migrate data text to jsonb only if table hostauditdata in data_type text

- [FIX] patch create_default_users when upgrading from 1.8.2 to 2.2
- [FIX] Fix unhandled redirections in TWaptServer wget
- [FIX] Add RedirectMax parameter in WaptServer WGet
- [UPD] added ubuntu 22.04 in waptagent bundle

WAPT Console

- [FIX] host config package are not editable right after creating them.
- [FIX] error editing same OU package in one session
- [FIX] CleanupPackagesCache proper unlock even if no assigned package

WAPT Agent

- [FIX] setuphelpers. reintroduce running_as_system for linux and mac (uid==0)

46.3.5 WAPT-2.2.2.12388 (2022-07-22)

hash: 10e35aa7

This is mainly a bugfix release.

Note:

- There is a change in the wapt the wapt->glpi sync is working, please refer to documentation for upgrade.
- Tech preview: new multiserver console support (connect to multiple wapt server using one console).
- Added support for ubuntu 22.04 amd64.
- `def update_package()` function can now be located in a separate `update_package.py` file. New packages from wapt store will use this new format to make `setup.py` simpler and more readable. Older wapt version are not impacted for package import and package install, but may be impacted if one wants to update directly from the WAPT Console using `update_package` script.

WAPT Deployment Server (WADS)

- [NEW] injecting oem key by slmgr command
- [FIX] fix tftpserver window size handling (bug on Dell uefi bios)
- [FIX] allow djoin with machine in default container CN=computers
- [FIX] improve error message when using standard user on MS AD for djoin.exe when >10 machine quota join has been reached
- [FIX] allow saving / renaming bundle names and check for empty names
- [IMP] add ACL on WADS (before it needed admin level ACL)
- [NEW] add post_install script windows
- [NEW] add ignore_ipxescript and move conf file and ipxescript

- [NEW] Basic Linux OS Deploy support : add Debian ipxe script template
- [NEW] add {{server_url}} {{secondary_repo}} and {{hostname}} in get_wads_config
- [NEW] add mustach templating in ipxescript
- [FIX] waptconsole uploadWinPE : fix regression in upload progress bar and incomplete zip.
- [FIX] add a progression form when uploading ISO and winpe
- [IMP] add wapttftpserver service shutdown in upgrade sequence (throught net stop, not only taskkill)
- [IMP] add tftp firewalld port opening on Redhat

WAPT Console

- [NEW] techpreview: waptconsole reporting multiservers.
- [FIX] Fixed check that downloaded waptsetup version is same or newer than server.
- [NEW] Download from https://wapt.tranquil.it and upload on local waptserver agents for Linux and macOS directly from the WAPT Console.
- [NEW] Added a popupmenu *Copy to clipboard* as `json` for audit data.
- [NEW] Display audit history audit data explorer (treeview + html template) + allow drag/drop of a audit `json` value subkey from value tree explorer.
- [IMP] waptwua: update waptwua status to *NEED-SCAN* on hosts when `download_wsusscan` is triggered and `wsusscn2.cab` file is downloaded.
- [IMP] Package import: Don't take care anymore of maturity for version when it's compared to store version.
- [FIX] Added licence validity check tolerance +1 day.
- [FIX] Fixed trigger downloads when triggering updates from the WAPT Console.
- [FIX] Allow ~ in package names (for spaces in Organizational Unit packages).
- [UPD] Updated icons on windows update status for WUA.
- [NEW] New option `check_package_version` in `waptconsole.ini`.
- [FIX] Fixed saving empty value in Editor for packages.
- [UPD] waptconsole reporting: Added a quick search filtering zone for the query result.
- [FIX] Wrong message when no admin rights and waptagent need upgrade or not present.
- [UPD] When going outside modified rules. A popup will ask to save or not the rules.
- [UPD] Delete host popup.
- [NEW] Added feature to download packages when asking hosts for update.
- [UPD] `trigger_host_update` adding possibility to download the package after update.
- [FIX] Saving language parameter.
- [UPD] Added a *NEED-SCAN* waptwua.status, updated when Wapt.update() is called.
- [FIX] Fixed layout on Windows Update form.
- [NEW] waptconsole: multiserver: manage packages repositories by server.
- [FIX] waptconsole: re-enable dataexport to `csv` for grids.

- [NEW] Explicit hint on number version when the package is not up to date (GridPackages)
- [UPD] waptconsole: Improved drag drop of columns into GridHosts
- [NEW] waptconsole: New Htmlviewer for audit data and Html auditdataview template filename (`wapttemplates`) calculated from section and key, or section.
- [FIX] waptconsole drag/drop audit values.
- [IMP] waptconsole: Load Active Directory Groups in thread.
- [FIX] waptserver: Improved message when triggering action.

WAPT Server

- [FIX] glpi sync: simplified `glpi_upload_hosts.py` script.
- [NEW] techpreview waptserver: endpoint `update_hosts_audit_data` to bulk insert hosts related data (for third party data integration).
- [NEW] Added multiserver endpoint for multiserver WAPT Console.
- [FIX] waptserver update_audit_data fix on_conflicts for value_id.
- [IMP] waptserversetup: take in account wapt_folder parameter in `waptserver.ini` when upgrading a setup.
- [IMP] Use utc time for acls expiration check.
- [FIX] Fixed waptserver unable to delete some hosts when CRL is enabled.
- [IMP] waptserver db install: try to register *`jsquery`* extension to make *`json`* query more powerful for reporting (this is not yet mandatory).
- [IMP] Renamed **`waptsetup-tis.exe`** to **`waptsetup.exe`** on the WAPT Server.
- [IMP] Include **`waptsetup.exe`** in **`waptserversetup.exe`** on Windows.
- [IMP] Download from TIS / upload to the WAPT Server of the installation packages of the WAPT Agents.
- [UPD] Create a full version 1.2.3.rev-hash into file `wapt/version-full`
- [IMP] Added HTST header to nginx template.
- [DEL] Removed direct integration of GLPI sync into WAPT. Now switched to plugin sync
- [FIX] Added **`trigger_host_action`** ACL on /api/v3/connected_wol_relays (used by /api/v3/trigger_wakeonlan)
- [IMP] Force calc_md5 if new filename in server.
- [IMP] Improved websockets performance and reliability. Now websocket ids are stored in memory instead being written in the database.

WAPT Agent

- [FIX] Fixed threading exception in **WAPTExit** and **WAPTTray** that could prevent status updates.
- [NEW] WAPTWUA superseded support. option *include_potentially_superseded_updates* in configuration wizard.
- [NEW] Added snap software inventory.
- [FIX] waptmessage unable to load sqlite on Linux and macOS.
- [FIX] Fixed custom waptmessage logo on Linux.
- [FIX] Fixed waptservice configuration: sets the `configs_dir` relative to `wapt-get.ini` full path.
- [FIX] Fixed waptservice 'start_waptexit' with arguments
- [FIX] Fixed bad arguments sent to waptservice triggering upgrades with 'only_priorities' and 'only_if_not_process_running'
- [FIX] `Wapt.write_audit_data_if_changed`: writes data if previous data has expired.
- [IMP] **`wapt-get add-config-from-url`**: provides a meaningful message when hash is not provided.
- [FIX] Updated the template of dynamic *`json`* configuration packages to match the new location and the naming of *`json`* config related functions.
- [IMP] Improved dynamic configuration handling for the WAPT Agent.
- [FIX] waptservice: ensure a random secret_key for local waptservice session.
- [FIX] **`wapt-get update-package-sources`**: handles properly relative path to package sources.
- [IMP] **`wapt-get edit`** now opens changelog.txt, VSCod* now open `control` file too.
- [UPD] Changed default log path to `wapt/log` if writable.
- [IMP] waptservice waptself: local authentication with file token (ie. nopassword), handling of local groups.
- [NEW] use `--not-interactive` with **`register`** if install run in silent mode and not run update if install service.
- [IMP] waptself, wapt-get, waptexit, wapttray: kill check threads on close, even on linux to speed up application shutdown.
- [FIX] Linux: waptservice restart Linux: AttributeError: 'WaptServiceRestart' object has no attribute 'logger'.
- [IMP] macOS: normalize macos wapt install package name format.
- [FIX] macOS: Fixed registration failing in some cases.
- [IMP] macOS: Added mpkg support.
- [FIX] Fixed no hash in clipboard, added missing helper for `add-config-from-url` in **`wapt-get`**.
- [IMP] Limit access right to admins to log directory (in case non public stuff get written to log)

WAPT Core

- [IMP] Patched with_md5sum in `make_package_filename`.
- [IMP] Added options for update-package-sources.
- [UPD] wapt core: use datetime in UTC for `audit_data`.
- [NEW] wapt core: allow usage of an environment variable "waptbasedir" to specify the location of root waptbasedir.
- [FIX] configuration package template `setup_package_template_conf.py`.

- [IMP] Support for `def update_package` in file `update_package.py` instead of `setup.py` for better readability.
- [UPG] Upgraded **openssl** to 1.1.1o.
- [NEW] core: define path Wapt.configs_dir relative to Wapt.config_filename if the dir Wapt.config_filename..conf.f exists.
- [FIX] Fixed `waptcrypto`: certificate filename attribute was not set when loading a certificate chain.
- [FIX] Fixed new option copytree2 replace_at_next_reboot.
- [FIX] Avoid errors on `get_version_from_binary()` getting params.
- [FIX] Fixed keyword and name with installed_softwares in macOS and Linux.

46.3.6 WAPT-2.2.1.11957 (2022-06-02)

WAPT Deployment Server (WADS)

- [FIX] Fixed **wapttftpserver** restart on Linux.
- [IMP] Added *xml* template for windows 11 deployment.
- [FIX] if `verify_cert` is empty, then `verify_cert = False`.

WAPT Console

- [FIX] CheckLicence => licence is now valid one day before the real beginning.

WAPT Agents

- [FIX] Fixed **harakiri** on Linux.

46.3.7 WAPT-2.2.1.11949 (2022-05-18)

hash: 1b2dfbee

This is a bugfix release.

WAPT Deployment Server (WADS)

- [FIX] Fixed waptconsole: use ROOT in addition to CA windows system certificates stores when building winpe with `verify_cert = True`.
- [FIX] Fixed selinux rules for WADS.
- [FIX] Fixed non ascii character support in passwords.
- [IMP] wgetwads: add more logging data (wget). Disable exe signature certificate as this could be blocking if CRL can not be checked in winpe environment for example.
- [UPD] add a timer to wait for network in WADS.
- [UPD] Update **openssl** to 1.1.1n for WADS.

Other fixes

- [FIX] fix wrong GPO link on waptserver start page
- [FIX] fix some translation messages in console
- [FIX] wrong element order in message in ACL GUI
- [FIX] allow change password if user password has been cleared
- [UPD] update mormot2 for bug in TSynDictionary.AddOrUpdate()
- [UPD] update mormot statics for sqlite to 3.38.5 (required for mormot compatibility)

46.3.8 WAPT-2.2.1.11932 (2022-05-05)

hash: 6522dccb

This is a bugfix release.

WAPT Deployment Server (WADS)

- [FIX] wapttftpserver : better handling of UEFI PXE/TFTP boot
- [FIX] wads now include non CA certificates for winpe build
- [FIX] Not adding “cn” in OU
- [FIX] wapttftpserver : add firewalld rule on redhat based server for wapttftpserver
- [FIX] WADS : improve feed back on upload WinPE
- [FIX] wapttftpserver : kill wapttftpserver and uninstall service before installing it
- [IMP] waptserversetup: add wapttftpserver configuration for windows

WAPT Server

- [FIX] fix typo for rocky support as server
- [FIX] waptservice websocket reconnection: disable by default low level reconnect feature

WAPT Console

- [FIX] fix bad port configuration for veyon remote assistance support
- [FIX] Define default package prefix when creating empty package
- [FIX] patch setup_package_template_cert.py.tmpl
- [FIX] waptconsole: fix access violation when access to external repo is blocked or need a proxy.
- [IMP] package version in bold red if obsolete version compared to external repo for better accessibility

WAPT Agent

- [FIX] waptservice websocket reconnection: disable by default low level reconnect feature
- [FIX] add conf.d to rpm agent installers for the new agent configuration management
- [FIX] macOS: fix get_file_type in macos
- [IMP] macOS: silently attach dmg file
- [IMP] waptwua : improve consistancy between WUA history and WUA status
- [FIX] waptself: bad char case for png file (issue for linux)
- [IMP] add dummy running_on_ac for linux and mac for compatibility
- [FIX] waptutils.user_config_directory() did not work under system account.

WAPT Core

- [IMP] mormot2 static: add 3.38.2 hash
- [IMP] sync htmlviewer with latest github commits from https://github.com/BerndGabriel/HtmlViewer/tree/master
- [IMP] waptguihelper: improved the design for InputDialog form

46.3.9 WAPT-2.2.1.11899 (2022-04-06)

hash: 2d82654e

This is mainly a bugfix release.

A new tftpserver has been introduced and it will ease WADS installation and configuration as it will be directly integrated into WAPT.

WAPT Deployment Server (WADS)

- [NEW] add a wapttftpserver binary on windows and linux to act as a tftp server for WADS
- [FIX] WADS : don't use redirect
- [FIX] WADS : be tolerant if sendstatus can not be sent.
- [IMP] WADS : handle https for drivers (continued)
- [UPD] wads : get windows system certificates for WADS server bundle
- [UPD] implement https verifyCert in wads and wgetwads
- [IMP] add serial_number arg when calling server get_wads_config in wads
- [UPD] waptconsole wads: add audit columns (created/updated) in grids.
- [NEW] Add an action to prepare a host package in WADS OS Deploy grid
- [NEW] wgetwads : use code signing cert of TIS to check signature of json hashes file if no signer_certificate in json file

WAPT Console

- [UPD] OU "All" fixed to not editable on GridOrgUnits
- [FIX] waptconsole: wrong client https key password used for task polling thread.
- [FIX] waptwua packages : ALLOWED status in winupdates grid is kept between form display.
- [FIX] Package creation did not take silent flags in account
- [FIX] memory leak when refreshing packages list
- [FIX] waptconsole packages list: Showing all versions when "Last version only" is not checked
- [FIX] "property not found" in some grids when refreshing data.
- [FIX] running plugins on multiple hosts.
- [FIX] taking in account the platform when lookig for TIS store package version
- [FIX] nested progress notifications in uwaptserverconnection TWaptServer
- [FIX] Disabled pysources check at waptconsole startup.
- [FIX] external repo ini settings dialog when importing.
- [FIX] waptconsole. some ui elements are not disabled when switching to discovery on login.

WAPT Server

- [NEW] add support for postgresql 14 on centos7
- [UPD] wapt windows server: update to nginx 1.20.2
- [IMP] server postinstall : put nginx backups in a different dir than nginx config
- [FIX] waptserver: fix empty error message when trying to activate an existing licence

WAPT Agent

- [NEW] added new waptguihelpers : grid_dialog, filename_dialog, input_dialog, combo_dialog
- [FIX] waptdeploy multiple setupargs raise "Invalid variant operation"
- [FIX] missing root certificates when exporting system store certificates in lazarus app (GetSystemCABundlePath). Must trust CA + ROOT stores
- [FIX] setuphelpers: regression in maintaining backward compatibility for some const which are functions too (programfiles etc..)
- [FIX] be tolerant if uuid can not be regenerated (on linux, dmidecode can't be run as normal user in session-setup)
- [FIX] fix wget waptdeploy.exe waptagent.exe in wads and detect mismatch drivers config
- [FIX] waptagent regression : Revert "[UPD] waptservice : tasks don't notify server by default to avoid too frequent updates of database."
- [FIX] wapt-get : try to fix get service password on unix.
- [NEW] splitting remove_appx() with new function remove_user_appx() to avoid unexpected behavior
- [NEW] Add restart-waptservice action in wapt-get.py

- [FIX] fix publisher and version in installed_softwares macos
- [FIX] use waptservice to check if is_enterprise in waptexit (avoid direct access to local waptdb) (fix unable to access sqlite db on linux / mac)

WAPT to GPLI connector

- [FIX] glpi fix install_date
- [FIX] regression in glpi export (Softwares)

46.3.10 WAPT-2.2.0.11720 (2022-03-15)

hash: 8e07f388

This is the first release of the 2.2 serie of WAPT.

WAPT Core

- [NEW] Discovery mode for the WAPT Console
  - when checking acls, the licencing status is taken in account to enable or not actions.
  - maximum number of 300 managed hosts in discovery mode.

WAPT Deployment Server (WADS)

- [NEW] tech preview Automated Windows OS deployment called WADS :
  - Using a winpe image (network boot or usb key boot).
  - Shipping wimboot, ipxe.efi, undionly.kpxe, 7z.dll.
  - Added openssl win64 binaries for WADS
  - Added **`wads.exe`** and **`wgetads`** custom binaries in distribution.
  - Added WADS repo option in repo rules.
  - Added a WAPT Console page to list raw registered hosts, upload winpe images, define default config, uplaod drivers bundles.
  - On WAPT Server: added `/var/www/wads/` add a non protected `/wads` in **`nginx`** config.

WAPT Console

- [NEW] add columns in private repo to display newest software version (Tranquil IT effort to parse softwares providers download sites) and newest package version (from Tranquil IT store database).
- [NEW] Dynamic Agent configuration using *`.json`* files stored on the WAPT Server:
  - Added a `last_update_config_fingerprint` local param to keep track of current config.
  - Added 'configurations' (merged config overview) data when uploading host status to the WAPT Server.
- [NEW] Dynamic Agent configuration using config packages:

  - Added `templates/setup_package_template_conf.py.tmpl` package template.
  - Added a `wapt/conf.d` directory on the WAPT Agent to hold the installed *`.json`* configuration files.
- [NEW] New in the WAPT Console: added option to show the host WAPT Agent configurations overview.
- [NEW] New in the WAPT Console: option to display a graph of host packages dependencies.
- [NEW] New in the WAPT Console reporting: tabbed interface to displays multiple query results.
- [NEW] New in the WAPT Console: option to filter host inventory based on the result of a SQL query:
  - In reporting, right click on column which represent a host UUID and "choose as Host UUID" abnd save.
  - The query is then available in the combobos "Filter hosts on SQL query" in hosts inventory.
- [NEW] New in the WAPT Console: add a *Tech preview* Tab for packages development workflow:
  - Create from template;
  - Displays `waptdev` directory sources package status;
  - Basic git commands.
- [IMP] Improved the WAPT Console send message : enable use of HTML (copy & paste). HTML Preview.
- [IMP] Do not clear selection on mouse right-click when selecting package names in package edits.
- [IMP] refactored the WAPT Console code to remove most python calls:
  - removed `waptdevutils.py`, removed calls to WaptRemoteRepo, replaced by pure fpc code.
- [UPD] Updated the WAPT Console: merged selected hosts add/remove depends, add/remove conflicts in a single action/form
- [UPD] Updated the WAPT Console update package source: add a checkbox to enable package version increment.
- [UPD] Updated the WAPT Console 'plugins' config: warn user if not saved.
- [UPD] Updated the WAPT Console: removed obsolete Add ADS Groups to selected host action.
- [UPD] Updated the WAPT Console action *Refresh Host Inventory* triggers a **`update_server_status`** instead of a full computer register.
- [UPD] Updated the WAPT Console: host additional tools (rdp, vnc, etc) which requires to look for a connected IP are now run in a thread to avoid freezing the UI.
- [UPD] Start of use of mormot2 for X509 and RSA crypto instead of python bindings in the WAPT Console
- [FIX] waptconsole : store executable signature with new key name format (xxx.exe keys)
- [FIX] duplicated panels in initial configuration package wizard.

WAPT Self-Service

- [IMP] waptself: add logger.

WAPT Server

- [IMP] Improved the WAPT Server authentication: try ldap authentication only if `ldap_auth_server` is defined.
- [UPD] Updated the WAPT Server licencing: use **`waptlicences.pyd`** instead of pure python code.
- [UPD] Updated the WAPT Server: add config options `wads_folder` and `agent_folder`.
- [UPD] Updated the WAPT Server: improve GLPI export, add 'smodel' on GLPI exports and add 'monitors'.
- [IMP] force en_US.utf8 locale for linux services.
- [IMP] add /api/v3/latest_installed_package_version.
- [UPD] upgraded jquery to v3.6.0.

WAPT Service

- [NEW] Added `/opt/wapt/wapt-get.bin` to linux distributions.
- [NEW] New in the WAPT service: added a *WaptUnregisterComputer* task and **`unregister_computer`** socketio action.
- [IMP] Improved the WAPT service: improved logger.
- [IMP] Improved the WAPT service and the WAPT Agent take into account the licencing status:
  - Added a `licences` local params to store the current registered licences retrieved from the WAPT Server during the last update.
- [UPD] **`waptcrypto.py`**: made optional the joining of signer certificate when signing claims.
- [UPD] Updated the WAPT Deployment utility: increased timeout from 4s to 15s when pinging the current http WAPT service.
- [UPD] Upgraded **`dmidecode`** to v3.3 on windows.
- [UPD] Updated the WAPT service: do not check battery level for *WaptAuditPackage* task.
- [REF] Installers : merged `wapt.iss` and `common.iss`.
- [FIX] wapttasks: took in account non default config filename.
- [FIX] Fixed the WAPT service: reporting properly the user which created a task (either locally or using websockets).
- [FIX] Fixed the WAPT service: fixed icons in package local webpage.

wapt-get

- [IMP] wapt-get new config actions. Added actions:
  - **`add-config-from-file`**;
  - **`add-config-from-base64`**;
  - **`add-config-from-url`**;

  with parameters:
  - `--not-interactive`: Disables dialog to ask credential users (for batch mode);
  - `--waptbasedir`: Forces a different wapt-base-dir then default dir of `waptutils.py`;
  - `--devmode`: Enables devmode. dbpath is set to memory and certificate/key paths are in `userappdata`;

  - `--json-config-name`: The name of the *.json* file given with the action **json-config-from-file/base64/url**;
  - `--json-config-priority`: The priority of the json file given with the action json-config-from-file/base64/url.
- [UPD] Removed **update-packages** action synonym for **scan-packages**.
- [IMP] wapt-get added **update-status** action in service mode **wapt-get -S update-status**.
- [IMP] Enabled `--CAKeyFilename` and `--CACertFilename` wapt-get options
- [IMP] Added logger for waptguihelper pyd module. if `--loglevel` = `debug` in commandline, logger is activated.
- [IMP] Reporting the `use_repo_rules` flag to the WAPT Server in wapt_status
  - Report `is_enterprise` flag to the WAPT Server
  - Report installed antivirus and monitors in host inventory
- [IMP] Audit loop granularity based on actual installed packages:
  - Added **get_next_audit_datetime()** on Wapt class.
  - `waptaudit_task_period` attribute is now in the Wapt class instead of the WAPT service.
- [UPD] Removed the not functional `--dry-run` wapt-get option.
- [IMP] Improved **register** computer fallback from kerberos to password based authentication:
  - Do not send audit data when registering to limit workload.
- [IMP] Try registering computer if **update_server_status** fails because of authentication.
- [IMP] **waptpython.exe**, **waptpythonw.exe**, and **nssm.exe** are now signed with Tranquil code signing key.
- [NEW] added **pylint** and **black** modules. Added black configuration to **vscode** project template.
- [NEW] Added `setuphelpers.getscreens`.
- [IMP] Improved *SetupHelpers* unzip : new `extract_with_full_paths` argument (default True).
- [NEW] New *SetupHelpers* `listening_sockets()`.
- [IMP] Added `templates/setup_package_template_portable_exe.py.tmpl` and `templates/setup_package_template_portable_zip.py.tmpl` package templates.

Others stuff

- [IMP] Added `windows_version_prettyname` and `windows_version_releaseid` in `host_info`.
- [IMP] Always use **RunAsAdminWait** to copy package certificate to the local WAPT service `waptssl` directory.
- [IMP] Improved the WAPT Console config: stores WAPT Server certificate in `AppUser` folder (`roamingwaptconsolesslserver`).
- [IMP] Reset TLS client key password in the WAPT Console config if connection error.
- [UPD] Retire python `GetPrivateKeyPath`, raise exception if `GetPrivateKey` does not succeed.
- [FIX] Clear cached TLS client key password when validating the the WAPT Console config dialog.
- [IMP] Improve GLPIlpi settings windows.
- [IMP] Clean up the html error page from the WAPT Server when checking the WAPT Server and WAPT repository URL.

- [FIX] Don't reenter the private key password dialog if already asking the user. This issue can be triggered if several therad are using a key, or if cooperative multitasking like TAction messages (OnUpdate) triggers a Get with client side certificate authentication.
- [SEC] Fix `dhparam` on the WAPT Server postconf.
- [FIX] Fix failover on file version with **`remove_outdated_binaries()`**.
- [IMP] Add `asset_tag` to sysinfo api.
- [FIX] `Get_antivirus_info`: test if timestamp attribute exists.
- [IMP] New getscreens function.
- [IMP] Added columns *uuid manufacturer* and *product serialnumber* in database.
- [UPD] Added `mac_addresses` to `LocalSysinfo`.
- [UPD] Expanded LocalSysinfo with uuid, serial_number and sku_number, fixed keys with underscore.
- [IMP] Improved matching of reachable IPs of client using new GetReachableIP from mormot2.
- [UPD] GetReachableIP: connection tests are performed in parallel using mormot GetReachableAddr instead of one after the other to reduce delay when launching IP based command to remote hosts from the WAPT Console.
- [FIX] Take `--config option` in account for wapt-get fpc code.
- [UPD] waptcrypto: implemented `TX509Certificate.CN`, removed `TX509Certificate.DN`.
- [UPD] Updated *SetupHelpers* **`need_install`**: now comparing software versions with 4 members. Assumes that 1.2 == 1.2.0.0 and 1.2.3.4.5 == 1.2.3.4, **`remove_previous_version`**: use version with 4 members.

46.4 WAPT-2.1 Serie

46.4.1 WAPT-2.1.2.10652 (2022-01-10)

hash: 7dd63b61

- [UPD] shorten the default package filename. If `target_os` is alnum, do not include md5sum in the filename. If `target_os` is in tags, do not duplicate it in filename
- [FIX] disable debug data for linux
- [FIX] try to circumvent issue with Trend antivirus blocking the **`WaptTaskManager`**. Looks like the issue is with platform.win32_ver using win32api.GetVersionEx…
- [FIX] Installed softwares invalid conditions
- [FIX] fix local_user and local_group on macOS
- [FIX] removed workaround on 60s delay for websocket disconnect
- [FIX] use CompressGZip instead of CompressZLib on the WAPT Server, compression is GZip
- [FIX] Allow '~' in package filenames
- [FIX] try to not update records in database if data has not changed
- [FIX] Wake on lan relay now equals is remote repository
- [FIX] fix group members

- [FIX] return only local and user group (ignore nsswitch)
- [FIX] backported the WAPT Exit utility (improved detailed logging) from 2.2
- [FIX] backport waptlicences py module from 2.2
- [SEC] check that hostname matches https certificate in the WAPT Console http client.
- [FIX] backport uwaptlicencing: allow empty json licencing data
- [FIX] fix WaptHttpPostData
- [FIX] check valid uri in wapthttputils waptwget WaptWget_Try
- [FIX] init LastModifiedDate to '' if not found in THttpResponse
- [FIX] add a 50ms report delay for httpprogressnotification
- isolate wapt python engine: PyFlags:= [pfNoUserSiteDirectory, pfIsolatedFlag];
- [FIX] Fixed *SetupHelpers*: backported changes from 2.2 is_linux64 type_rhel fix installed_softwares for type_redhat upd uninstall_apt with autoremove
- [FIX] `user_appdata` = `user_local_appdata` for unix
- [IMP] introduced get_powershell_str, get_default_app remove_appx
- [IMP] introduce InitLogger for the WAPT Exit utility
- [FIX] Fixed the WAPT Console: generalize the use of a fallback package_uuid in case of old packages without package_uuid field.
- [FIX] Fixed the WAPT Console: use editable dropdown in frmpackagedetails for maturity
- [FIX] backport issue with inc version of some group packages when importing
- [FIX] Disable client side ssl authentication on root WAPT Server url (regression)
- [FIX] isolate from user python env when building binary packages
- [UPD] improved feedback message for license activation on the WAPT Server.
- [UPD] wapt-scanpackages.py: add option -d to disable update of database Packages table.
- [FIX] The -b switch is True by defaut, so there were no way to disable update of database table.
- [UPD] Updated the WAPT Console: be tolerant for old package without package_uuid
- [UPD] strip ending slash in {{data.wapt.hostname}} server template properties to avoid double slashes in templates result
- [UPD] backport openssl build parameter from 2.2
- [FIX] Fixed the WAPT Agent url link in the WAPT Server index page.
- [FIX] setproctitle only for unix
- [FIX] locate packages in host packages grid using package_uuid instead of id, so that refreshing grid works properly with a multiselection of hosts.
- [UPG][SEC] upgrade python version from 3.8.11 to 3.8.12
- [FIX] remove python3 dependencie. Now python3 is included in wapt

46.4.2 WAPT-2.1.2.10605 (2021-11-30)

hash: e2a0e2a0

- [FIX] Fixed the WAPT Console: backport edit multiple hosts add/remove depends/conflicts (issue "no password available yet" when kerberos enabled) backport IpExecute from 2.2
- [FIX] unable to edit stripped down package with integrated package editor. (setup.py file hash issue) update package size
- [FIX] bad path for nginx dhparam for Windows server
- [FIX] upgrade mormot2
- [FIX] waptself local admin NOPASSWORD setting did not work anymore log authentication user when task is triggered from local wapt webservice don ot raise exception in check_auth_groups but return (None, None) instead to avoid Error 500 in browser backport fix for integer attributes in packages index backport fix for loading ssl libraries
- [FIX] Update wake on lan with broadcasts
- [FIX] Error "Add: Unexpected [%] object property in an array" for old package with empty package uuid
- [FIX] Acl handle boolean as global ACL
- [FIX][SEC] issue with acls : action is enabled when acl is set to json false

46.4.3 WAPT-2.1.2.10588-rc1 (2021-11-22)

hash: e70d9039

- [FIX] fix installed_softwares for older debian and improve inventory performance
- [FIX] fix glpi inventory failure (exception on int conversion)
- [SEC] [FIX] invalid condition on package hash check
- [SEC] [FIX] cleanup nginx config templates
- [NEW] add uwsgi support for Debian server
- [FIX] add user information in audit
- [FIX] Improve lazarus ini parser to support other values than '1'/'0' as boolean values (True, true, 1, 01, etc. same behavior as python iniparse)
- [IMP] support for message previsualisation and templates in waptmessage editor and better multiline support
- [UPD] waptsetup : do not use kerberos by default
- [NEW] show certificate when double click in acl tab
- [IMP] Do not propose to start the WAPT Console after install (due to different user context)

46.4.4 WAPT-2.1.1.10568 (2021-11-08)

hash: 978c00ae

This is a bugfix version with some small improvements. The main fix is for websocket issue.

- [IMP] Prevent multiple websockets connections from same host uuid on the WAPT Server (bugged wapt clients can maintain multiple websockets, which leads to a lack of avalable connections on the WAPT Server)
- [FIX] Fixed restart of the WAPT service with exit code 10 (managed by the nssm service manager)
- [FIX] Fixed case on the WAPT service where different threads access simultaneously to a shared Wapt instance
- [IMP] Introduced some randomness when the WAPT service reconnects its websocket.
- [IMP] Checking more cases to determine if token for websocket has to be updated.
- [IMP] Introducted a wait in the socket client until it is actually disconnected before trying to reconnect to avoid multiple websocket threads from same client.
- [IMP] Do not re-create a new SocketIOClient at each reconnection, but reuse existing one to minimize risk of multiple connections.
- [FIX] Do not consider '%' char as unsafe in filenames
- [IMP] Improved logging of the WAPT service (logger wapttasks report main actions triggered by the service in `waptlogwaptservice.log`). Removed 'flask.app' logger config.
- [IMP] Remove the WAPT packages's persistent directory on the WAPT client when a WAPT package is forgotten
- [IMP] Added `ignore_empty_names` argument to *SetupHelpers*.installed_softwares
- [IMP] Improved display of `package_uuid` with command wapt-get list
- [IMP] Added *redhat_based* tag for WAPT package operating system tags
- [FIX] Fixed `decrypt_fernet` / `fernet_encrypt` functions
- [IMP] Improved the reporting of key as name in softwares inventory for softwares without a descriptive name
- [FIX] The `server_uuid` column in hosts database updates properly.
- [FIX] Fixed the removal of packages when `only_if_not_process_running` = `True`.

Known issues:

- When the websocket is reconnecting, if the IP adrress has changed, the main IP adrress is not updated in IP adrress column in the WAPT Console.

46.4.5 WAPT-2.1.0.10550 (2021-10-08)

hash: 953c9552

This is a bugfix version with some small improvements.

- [FIX] Fixed mass add / remove on multiple host at once.
- [FIX] Fixed issue when editing a package without a "description_en" attribute in control file.
- [FIX] Fixed drag drop when editing *selfservice* package.
- [IMP] Improved feedback when uploading WAPT packages.
- [IMP] Improved handling of the list of wakeonlan relay.

- [IMP] Improved remote repository is now by default a wakeonlan relay.
- [FIX] Fixed access violation error when viewing certificate list.
- [FIX] Fixed do not enable verbose logging by default on the WAPT Console, the WAPT Exit utility and waptselfservice (might fill up %APPDATA% ...).
- [FIX] Fixed use `templates/wapt-logo.png` in the WAPT Exit utility if it exists.
- [IMP] Improved login error message.

46.4.6 WAPT-2.1.0.10517 (2021-09-30)

hash: fa2af298

This is the first release of the 2.1 branch. It is mainly a incremental improvement with many small but worthy fixes on the 2.0 branch.

The WAPT service

- [IMP] During upgrade, **`wapt-get session_setup`** is not run if no userspace configuration is defined for the installed WAPT packages.

The WAPT Deployment utility

- [IMP] Improved automatic proxy detection and configuration possible with the new `--http_proxy` = `True` / `False` parameter or explicit url command line parameter.
- [IMP] Disabled https verification when downloading **`waptagent.exe`** if a fingerprint is provided (allows installation with on out-of-date computer with expired certificate store).
- [IMP] Do nothing if no –waptsetupurl argument is provided (it reduces the probability of false positive on antivirus check).
- [IMP] Double check WAPT installed version after install and report error message if it does not match (allow detection of installation that have been blocked by a misconfigured antivirus for example).

The WAPT Console

- [NEW] tech preview: new tab to provide basic package editing functionnality directly in the WAPT Console without having to open **`Pyscripter`** or **`VSCode`**.
- [NEW] New tech preview: new tab to browse the developement directory directly from the WAPT console.
- [NEW] Single Sign On with Kerberos authentication (if `service_auth_type` = `waptserver-ldap` and `use_kerberos` = `True`).
- [NEW] New button to display WAPT packages that have a specific WAPT package as a dependency in the private repository tab.
- [NEW] New message box to decrypt message sent by the WAPT Agents (using `encrypted_data_str` / `print_encrypted_data` in waptcrypto). This allows an admin to upload sensitive information from desktop that will be asymetrically signed by the Administrator's public key.
- [NEW] New set of icons and many small visual improvments.
- [NEW] New software inventory tab to display installed software (not packages) and see which hosts have that specific software.
- [NEW] New button to delete Windows Update KB files that are not used anymore by any computers. This allows to keep the Windows Update storage volume under control.
- [NEW] New tab to have a user-friendly display of the certificates that are deployed on a specific host.
- [NEW] New tab to display the certificates that are available on a WAPT repository.

- [NEW] New warning icons on the hosts tab when the computer needs a restart (after a windows update for example).
- [NEW] New filter by OS option.
- [NEW] New icons in the OU tree view if a OU package exists for that Organizational Unit.
- [NEW] New information message about the choice of maturity when creating new WAPT Agent and by default uploading in DEV maturity (to avoid being directly deployed to all client computers, this allow to test the new WAP Agent on a subset of computer before full scale deployment).
- [IMP] Made GLPI export configuration more intuitive.
- [IMP] Improved the WAPT Console plugin versatility. All inventory attribute can now be used in command lines (it use the "mustache" template syntax, eg. {{ main_ip }} {{ computer_fqdn }} {{ host_capabilities.os_version }} "{{#host_capabilities.tags}}{{.}},{{/host_capabilities.tags}}" etc.
- [IMP] Allow non standard port in the WAPT Console configuration.

waptself

- [NEW] allow custom logo in waptselfservice
- [NEW] Single Sign On using Kerberos (`needs service_auth_type` = `waptserver-ldap` and `use_kerberos` = `True`)
- [IMP] allow customisation of package details view using template engine

WAPT Exit utility

- [IMP] allow custom logo (on Windows, Linux and macOS)

wapt-get

- [NEW] better handling of licence information. Now the licence is uploaded on the WAPT Server and it is not necessary to install it on every admin WAPT Console computer
- [IMP] propagate ExitCode from Python calls for better error handling
- [IMP] better handling of websocket reconnection (check of socket status every 120s)
- [IMP] periodic check of the UUID and the current certificate of the WAPT Agent for consistency between the WAPT Agent and the client computer
- [NEW] waptsetup et waptserversetup new parameters: `set_verify_cert` and `set_kerberos`

46.5 WAPT-2.0 Serie

46.5.1 WAPT-2.0.0.9470 (2021-10-07)

hash: 5065cb57

This is a security release with a few related bugfixes. All Wapt 2.0 version below 2.0.0.9467 are affected.

- [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] Do not use PackageEntry filename attribute to build target package filename as it is not signed.

- [UPD] **`wapt-get remove`**: reraise exception if there is exception in uninstall script return traceback in 'errors' key return code 3 if there are errors when removing packages in **`wapt-get remove`**.
- [FIX] handles wildcards in certificates in the WAPT Console config and create waptsetup update UI in external repositories config when setting CA bundle.
- [FIX] use PackageEntry.localpath only for local status of a package.
- [UPD] split PackageEntry non_control_attributes into *repo_attributes* and *local_attributes*. *local_attributes* are not put into Packages index as they are not relevant for remote access.
- [UPD] update python modules requirements following urllib3 upgrade idna==3.2 (from 2.10) certifi==2021.5.30 (from 2020.12.5) requests==2.26.0 (from 2.25) urllib3==1.26.6 (from 1.26.5)

46.5.2 WAPT-2.0.0.9450 (2021-08-10)

hash: 7bc6920c

This is a security fix version affected by CVE-2021-38608.

Please visit the *security bulletin* to learn more.

46.5.3 WAPT-2.0.0.9449 (2021-06-22)

hash: 70283a14

This is a bugfix version with some small improvements.

WAPT Agent

- [FIX] Fixed Windows Update fix in the progress bar.
- [IMP] Allow the WAPT Agent to upgrade even when on batteries.

The WAPT Server

- [IMP] Many fixes in GLPI sync.
- [FIX] Better handling of service_delete exception cases.
- [FIX] Fixed database migration handling with `create_defaults_users` procedure.
- [FIX] Fixed on windows skip the WAPT Agent build if there is no available certificate for signing.

The WAPT Core

- [IMP] Improved the compatibility of `Packages` file for easing upgrade from WAPT 1.8.2.
- [IMP] Improved the WAPT Deployment utility: behavior to avoid wrong red flag from AV softwares.

Caveat

For macOS support one should use the WAPT Agent 2.1 version available in nightly channel.

46.5.4 WAPT-2.0.0.9428 (2021-05-06)

hash: 4b33cf96

This is a bugfix version with many small improvements.

WAPT Console:

- [IMP] Improve *CreateWaptSetup* form layout.
- [IMP] Restore focused column visibility when refreshing grid data.
- [FIX] Fix wrong path for wapt-get.py in vscode project.
- [UPD] Update No fallback in rules to true by default.
- [FIX] `enable-check-certificate` with wildcard.
- [FIX] take into account the `use_http_proxy_for_repo` ini setting (if not present, assume `False`).
- [FIX] Fix `setup_package_template_msu.py.tmpl` for package Wizard.
- [IMP] Add new template for creating package with certificate.
- [IMP] Add option to check downloaded package with VirusTotal in package import GUI.
- [IMP] Add update-package source action directly in Private repository in the WAPT Console.

WAPT Agent:

- [IMP] Use task queue for the forced installs instead of running them inline.
- [FIX] Database not opened when we check Hosts who are secondary repositories.
- [IMP] Restart partial download of Windows Update files.
- [IMP] Improved icons handling in **`WaptSelfService`**.
- [IMP] On macOS use host certificate store by default for https certificate validation.
- [IMP] `reload_config_if_updated` now reload config if `public_certs_dir` has changed.
- [FIX] WUA: better handling of return code “does not apply to this computer”.

WAPT Server:

- [FIX] Fixed bad migration of PGSQL databse server side.
- [FIX] Improved database upgrade in corner cases.

SetupHelpers

- [FIX] Fixed `register_windows_uninstall` calculation and using correct x86_64 environment with **`register_uninstall`** and **`unregister_uninstall`**.
- [IMP] Improved inline function description for documentation.

46.5.5 WAPT-2.0.0.9343 (2021-04-08)

hash: 117d62b8

This is mainly a bugfix release after the initial 2.0.0 release.

WAPT Console:

- [IMP] Show an explicit message if the user can not build a customized WAPT Agent.
- [IMP] Enabled remote repo sync if there are repo configured (making `remove_repo_support` parameter obsolete).
- [IMP] Better filtering on `maturities`.
- [FIX] Fixed templates for vscode

WAPT Server:

- [IMP] Include certificates from WaptUsers table in result of /api/v3/known_signers_certificates.

WAPT ACL handling:

- [UPD] ACL: added an action to show the user certificate.
- [UPD] Creates default (empty) WaptUserAcls record on user login even for non ldap logins.
- [IMP] Better naming for ACL domains.

SetupHelpers

- [FIX] Fixed `register_uninstall`.
- [FIX] Do not change silently `maturity` and `locale` in `check_package_attributes`.
- [FIX] Fixed regression in wget resume.

Other technical stuff:

- [IMP] Added support for installation on OracleLinux.
- [FIX] Tightened files ACLs on Linux + fixes + SELinux fixes in postconf.
- [IMP] Introduced **`mORMot2`** framework in Lazarus code.
- [FIX] Fixed datetime conversion in the WAPT Console.

46.5.6 WAPT-2.0.0.9300 (2021-03-30)

hash: 018b8b57

This is the first release of the 2.0 series. After one year in development and more than 1600 commits it brings a bunch of new features and enhancement to the last major update of WAPT 1.8.2. On the technical side WAPT 2.0 now embed Python3 and now support 8 new platforms (some of them backported to 1.8.2).

The switch to Python3 may require minor adjustment to the existing package that may have been development in-house (refer to the corresponding doc page). The packages offered by Tranquil IT through the WAPT Store are already compatible with WAPT 2.0.

From a sysadmin point of view

- [NEW] ACLs.
- [IMP] WAPT Server side ACLs in addition to certificate validation.
- [IMP] User management interface with certificate listing.
- WAPT Console:
- [IMP] gui: change maturity directly from the WAPT Console.
- [IMP] gui: all WAPT package types are grouped in one tab.
- [IMP] helpers: build and upload locally development package from the WAPT Console.
- [IMP] helpers: import default reporting queries from internet.
- [IMP] helpers: restart the WAPT Agent and restart client computer from the WAPT Console.
- [IMP] Package wizard: support for RPM/DEB/PKG/DMG.
- [IMP] Remote repositories: status bar for progression of creation/ update of `sync.json` for repo sync.
- [IMP] Windows Updates: new search bar, view host with specific KB.
- [IMP] Faster import and resigning of package, change of maturity, etc.
- [IMP] **`waptmessage`**: better handling of user oriented notification.
- [IMP] Better logging of WAPT Console actions and WAPT Agent activity.
- Performance improvements for larger installations:
- [IMP] Better handling of insert / update of inventory.
- [IMP] Better handling of websocket updates.
- [IMP] GLPI integration: synchronize WAPT inventory to GLPI server.
- Better OS integration:
- [IMP] TLS certificate handling: **`certifi`** uses local OS certificate store instead of Python **`certifi`** integrated certificate store.
- [IMP] Increased the number of supported platform, improved packaging for Linux (deb and rpm) with support for a WAPT Agent running on arm64 and macOS BigSur 64bit.
- Package development:
- [IMP] Improved package wizard.
- [IMP] Many small fixes and improvements to *SetupHelpers* and better support for Linux and macOS.
- [IMP] Improve os targeting now you can specify targeted OS and specific version of OS : eg. Debian(>=9,<=10).

From a technical point of view

- Python: switch from Python2.7 to Python3:
- Linux: use of venv by default with distrib python 3 version.
- Windows: switch python3 install to embedded edition 3.8.7.
- Different installer for WinXP / WinVista / Win2k3r2 / win2k8 (nonr2) (recent CPython version does not support older Windows systems anymore).
- Better handling of passwords with special chars.
- Upgraded WAPT core libs and scripting environment.
- Upgraded to Python3 and Python libraries, changed kerberos and websocket libraries.
- Upgraded to Lazarus 3.0.10 and FPC 3.2.

Caveat

- Support for non supported Windows version (WinXP, WinVista, Win2k8 (non-R2) and Win2k3) is still baking in the oven and should be ready shortly after the 2.0 release date.
- Redhat8 and derivative distributions: for upgrade it is necessary to remove WAPT SELinux rules before using postconf again.

CHAPTER

FORTYSEVEN

WAPT END USER LICENSE AGREEMENT

NOTICE: READ THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE YOU DOWNLOAD, INSTALL OR USE Tranquil IT's PROPRIETARY SOFTWARE. BY INSTALLING OR USING THE SOFTWARE, YOU AGREE TO BE BOUND BY THE FOLLOWING TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THE FOLLOWING TERMS AND CONDITIONS, DO NOT INSTALL OR USE THE SOFTWARE.

47.1 Definitions

"You and your" means the party licensing the software hereunder.

"Software" means the computer programs provided under the terms of this license by Tranquil IT together with any documentation provided therewith.

"WAPT Server" means the system running the WAPT server software.

"Managed computer" means a computer running the WAPT service agent software.

47.2 Grant of rights

47.2.1 General

The license granted for software under this agreement authorizes you on a non-exclusive basis to use the software. The license is personal to you and may not be assigned by you to any third party.

47.2.2 License Provisions

Subject to the receipt by Tranquil IT of the applicable license fees, you have the right use the software as follows:

> You may use and install the WAPT client software for the duration of the license on as many "managed computers" as the license agrees. Nothing in this agreement shall permit you, or any third party to disclose or otherwise make available to any third party the licensed software, source code or any portion thereof. You agree to indemnify, hold harmless and defend Tranquil IT from and against any claims or lawsuits, including attorney's fees, that arise as a result from the use of the software; You do not permit further redistribution of the software by your end-user customers

47.3 No derivative works

The inclusion of source code with the License is explicitly not for your use to customize a solution or re-use in your own projects or products. The benefit of including the source code is for purposes of security auditing. You may modify the code only for emergency bug fixes that impact security or performance and only for use within your enterprise. You may not create or distribute derivative works based on the software or any part thereof. If you need enhancements to the software features, you should suggest them to Tranquil IT for version improvements.

47.4 Ownership

You acknowledge that all copies of the software in any form are the sole property of Tranquil IT. You have no right, title or interest to any such software or copies thereof except as provided in this Agreement.

47.5 Confidentiality

You hereby acknowledge and agreed that the software constitute and contain valuable proprietary products and trade secrets of Tranquil IT, embodying substantial creative efforts and confidential information, ideas, and expressions. You agree to treat, and take precautions to ensure that your employees and other third parties treat, the software as confidential in accordance with the confidentiality requirements herein.

47.6 Disclaimer of warranties

EXCEPT AS OTHERWISE SET FORTH IN THIS AGREEMENT THE SOFTWARE IS PROVIDED TO YOU "AS IS", AND Tranquil IT MAKES NO EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO ITS FUNCTIONALITY, CONDITION, PERFORMANCE, OPERABILITY OR USE. WITHOUT LIMITING THE FOREGOING, Tranquil IT DISCLAIMS ALL IMPLIED WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR FREEDOM FROM INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO YOU. THE LIMITED WARRANTY HEREIN GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY FROM ONE JURISDICTION TO ANOTHER.

47.7 Limitation of liability

You ACKNOWLEDGE AND AGREE THAT THE CONSIDERATION WHICH Tranquil IT IS CHARGING HEREUNDER DOES NOT INCLUDE ANY CONSIDERATION FOR ASSUMPTION BY Tranquil IT OF THE RISK OF YOU CONSEQUENTIAL OR INCIDENTAL DAMAGES WHICH MAY ARISE IN CONNECTION WITH YOUR USE OF THE SOFTWARE. ACCORDINGLY, YOU AGREE THAT Tranquil IT SHALL NOT BE RESPONSIBLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS-OF-PROFIT, LOST SAVINGS, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF A LICENSING OR USE OF THE SOFTWARE.

47.8 Indemnification

You agree to defend, indemnify and hold Tranquil IT and its employees, agents, representatives and assigns harmless from and against any claims, proceedings, damages, injuries, liabilities, costs, attorney's fees relating to or arising out of your use of the software or any breach of this Agreement.

47.9 Termination

Your license is effective for a defined period and is terminated when this period is over. You may terminate it at any time by destroying the software or returning all copies of the software to Tranquil IT. Your license will terminate immediately without notice if you breach any of the terms and conditions of this Agreement, including non or incomplete payment of the license fee. Upon termination of this Agreement for any reason: you will uninstall all copies of the software; you will immediately cease and desist all use of the software; and will destroy all copies of the software in your possession.

47.10 Updates and support

Tranquil IT has the right, but no obligation, to periodically update the software, at its complete discretion, without the consent or obligation to you or any licensee or user.

YOU HEREBY ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

CHAPTER

FORTYEIGHT

EXTERNAL COMPONENT LICENSES USED IN WAPT

WAPT software development was started in March 2012 by Tranquil IT.

With WAPT >= 1.9, developments done within WAPT are licensed under a *proprietary license*.

Table 1: External components licenses used in WAPT

WAPT components	License
Python	Python Software License
Python Libraries	Various open-source licenses
Lazarus	GNU Public License
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
mORMot2	Various open-source licenses