# Table of Contents

1. Copyright ........................................................................................................... 1

2. Introduction ......................................................................................................... 2
   2.1. Conventions of this document ........................................................................ 2

3. Requirements ........................................................................................................ 3
   3.1. Supported distributions for server ................................................................. 3
   3.2. Hardware requirements ................................................................................. 3
       3.2.1. Notes on determining hardware requirements ....................................... 4
   3.3. Configuration requirements ......................................................................... 4
       3.3.1. Valid DNS domain name ...................................................................... 5
       3.3.2. Valid DNS hostname .......................................................................... 5
       3.3.3. Correct name resolution for the server ................................................ 5
       3.3.4. Localization settings ......................................................................... 5
   3.4. Needed network ports .................................................................................. 6
   3.5. Proxy settings ................................................................................................ 7
       3.5.1. http_proxy ............................................................................................ 7
       3.5.2. https_proxy .......................................................................................... 7
       3.5.3. no_proxy .............................................................................................. 7

4. opsi-server Installation ......................................................................................... 8
   4.1. opsi-server Basic installation ........................................................................ 8
       4.1.1. Starting up the uib preconfigured Virtual Machine ............................... 8
       4.1.2. Opsi-QuickInstall ................................................................................ 13
       4.1.3. Prerequisites for an installation on a server ......................................... 18
       4.1.4. Installation on Debian / Ubuntu ............................................................... 20
       4.1.5. Installation on a Univention Corporate Server (UCS) ............................ 22
       4.1.6. Installation on openSUSE or Suse Linux Enterprise Server (SLES) .... 25
       4.1.7. Installation on CentOS or RedHat Enterprise Linux (RHEL) ............... 26
   4.2. Update and Configuration of the opsi-server ................................................ 28
       4.2.1. Proxy Entry in apt-configuration File ..................................................... 28
       4.2.2. Update of the opsi-server ...................................................................... 28
       4.2.3. Backend Configuration ........................................................................ 29
       4.2.4. Set Samba Configuration and Change Passwords .................................. 31
       4.2.5. Create users and configure the groups opsiadmin and opsifileadmins ... 31
       4.2.6. DHCP Configuration .......................................................................... 33
       4.2.7. Configuration of the name resolution .................................................... 34
   4.3. Importing the minimal opsi products .............................................................. 35
       4.3.1. Automatic import of the minimal opsi products ..................................... 35
       4.3.2. Manual import of opsi products ............................................................. 36

5. Management interface opsi-configed ................................................................. 37
   5.1. Installation of the management interface opsi-configed ................................ 37

6. Adding clients to opsi .......................................................................................... 38
   6.1. Creation of a new opsi client ......................................................................... 38
       6.1.1. Creating a new opsi client via the graphical management interface .... 38
       6.1.2. Creating a new opsi client via the command line ................................. 38
       6.1.3. Creating a new opsi client using the opsi-client-booted ....................... 39
   6.2. Integration of existing Windows clients ......................................................... 41
       6.2.1. Using the opsi-client-agent-installer .................................................... 41
       6.2.2. Using service_setup.cmd on Windows NT6 (outdated) ..................... 42
9.3.1. Installation When the User is Logged on .............................................................................. 97
9.3.2. Customization after a silent/unattended Installation ............................................................. 98
9.3.3. Integration with Automated Answers for the setup Program .............................................. 98
9.3.4. Analyze and Repackage ........................................................................................................ 99
9.3.5. How to uninstall Products ................................................................................................... 99
9.3.6. Known Issues with the 64 Bit Support ................................................................................ 101
10. More Information ..................................................................................................................... 102
1. Copyright

The Copyright of this manual is held by uib gmbh in Mainz, Germany.

This manual is published under the creative commons license 'Attribution - ShareAlike' (by-sa).

A description of the license can be found here: https://creativecommons.org/licenses/by-sa/3.0/

The legally binding text of the license can be found here: https://creativecommons.org/licenses/by-sa/3.0/legalcode

Most parts of the opsi software is open source. Not open source are the parts of the source code which contain new extensions, that are still under cofunding, which have not been paid off yet. See also: opsi cofunding projects

All of the open source code is published under the AGPLv3.

The legally binding text of the AGPLv3 license can be found here: http://www.gnu.org/licenses/agpl-3.0-standalone.html

Information about the AGPL: http://www.gnu.org/licenses/agpl-3.0.en.html

For licenses to use opsi in the context of closed source software, please contact uib gmbh.

The names 'opsi', 'opsi.org', 'open pc server integration' and the opsi logo are registered trademarks of uib gmbh.
2. Introduction

These instructions explain in detail the installation and starting of an opsi-server. It starts from the provided installation package and leads to the test installation of a client.

The installation and commissioning of an opsi-server is done in several steps:

1. Basic installation of the server
2. Configuration of the server (adaptation to network conditions, setting up users and passwords, installation of products to be distributed)
3. Recording and integration of computers in opsi.
4. Deploying Windows to Clients.
5. Packaging and distribution of own software

Then an operating system including software can be automatically installed on a client and a hardware and software inventory can be performed.

Further features are described in the opsi manual. There, you will also find explanations about the co-financed extensions and their setup.

The shown network configuration is exemplary and refers to a network without competing DHCP servers (e.g. an isolated test network in which the opsi-server and its clients can be placed for the first tests).

We strongly suggest that you make your first tests with opsi in a test network that is separate from other DHCP servers, but which you can temporarily connect to your main network, e.g. to download updates and packages from the Internet.

For the integration into existing networks you can use consulting services by uib.

2.1. Conventions of this document

Commands are highlighted separately:

```
this is a command
```

During installation and configuration, you can usually copy and execute the commands from these fields one after the other using copy & paste from this document.
3. Requirements

Subsequently the requirements for the installation of an opsi-server will be described.

3.1. Supported distributions for server

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Opsi 4.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian 11 Bullseye</td>
<td>✔️</td>
</tr>
<tr>
<td>Debian 10 Buster</td>
<td>✔️</td>
</tr>
<tr>
<td>Debian 9 Stretch</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 22.04 LTS Jammy Jellyfish</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 20.04 LTS Focal Fossa</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 18.04 LTS Bionic Beaver</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 16.04 LTS Xenial Xerus</td>
<td>✗</td>
</tr>
<tr>
<td>RHEL 8</td>
<td>✔️</td>
</tr>
<tr>
<td>RHEL 7</td>
<td>✗</td>
</tr>
<tr>
<td>CentOS 8</td>
<td>▲</td>
</tr>
<tr>
<td>CentOS 7</td>
<td>✗</td>
</tr>
<tr>
<td>Alma Linux 8</td>
<td>✔️</td>
</tr>
<tr>
<td>Rocky Linux 8</td>
<td>✔️</td>
</tr>
<tr>
<td>SLES 15 SP1</td>
<td>✔️</td>
</tr>
<tr>
<td>SLES 15 SP2</td>
<td>✔️</td>
</tr>
<tr>
<td>SLES 15 SP3</td>
<td>✔️</td>
</tr>
<tr>
<td>SLES 12SP*</td>
<td>✗</td>
</tr>
<tr>
<td>SLES 12</td>
<td>✗</td>
</tr>
<tr>
<td>openSuse Leap 15-3</td>
<td>✔️</td>
</tr>
<tr>
<td>openSuse Leap 15-2</td>
<td>▲</td>
</tr>
<tr>
<td>openSuse Leap 15-1</td>
<td>▲</td>
</tr>
<tr>
<td>openSuse Leap 15</td>
<td>▲</td>
</tr>
<tr>
<td>UCS 4.4</td>
<td>✔️</td>
</tr>
<tr>
<td>UCS 5.0</td>
<td>✔️</td>
</tr>
</tbody>
</table>

✔️: Supported  ✗: Unsupported  ▲: Under development  △: Discontinued

3.2. Hardware requirements

For an opsi-server on real hardware is needed:

- x86-64 or ARM64 system.
- At least 2GB RAM
- At least 2 CPU cores
- Disk space requirements depend heavily on the number of opsi packages. For productive systems the following applies:
There should always be at least 15GB of free space in the '/tmp' directory.

The directory '/var/lib/opsi' should have at least 60GB and should be flexible expandable.

How many opsi clients access an opsi server at the same time depends on the configuration and the daily routines in the respective environment. In large environments, with many simultaneous client connections, the RAM and CPU requirements can increase significantly.

The central opsi service 'opsiconfd' needs about 250MB RAM per worker process. There should be about one worker process available for 20 simultaneous connections. The number of CPU cores should be about half the number of worker processes.

In the standard configuration, additional resources for Samba, MySQL and Redis should be planned on the same system.

**3.2.1. Notes on determining hardware requirements**

Hardware requirements depend heavily on usage. So here are a few tips to calculate the system requirements.

**Memory requirements**

Each active Samba connection starts its own Samba process. Estimates vary between 800 kB and 4 MB. How many opsi clients access an opsi server at the same time depends heavily on the daily routines in your environment.

The following values were recommended for Samba 3:

<table>
<thead>
<tr>
<th>Process</th>
<th>1 user</th>
<th>130 users</th>
<th>500 users</th>
</tr>
</thead>
<tbody>
<tr>
<td>smbd</td>
<td>4 MB</td>
<td>520 MB</td>
<td>2000 MB</td>
</tr>
</tbody>
</table>

Since we do not have any values for current Samba versions, the above figures can only be regarded as a rough estimate and should be extended with a safety margin of 50%.

The memory consumption of opsiconfd depends heavily (but not only) on the number of clients. The following minimum memory consumption can be derived from existing installations. The specified number of users are not active users at the same time, but the total number.

<table>
<thead>
<tr>
<th>Process</th>
<th>100 users</th>
<th>2000 users</th>
<th>4000 users</th>
</tr>
</thead>
<tbody>
<tr>
<td>opsiconfd</td>
<td>500 MB</td>
<td>1000 MB</td>
<td>2000 MB</td>
</tr>
</tbody>
</table>

You should also implement a safety margin here.

**CPU**

Opsiconfd currently uses only one core. This core is only fully loaded when many opsi clients (> 100) access the server exactly at the same time. But the operating system, Samba, the database, etc. also require computing time.

I.e. with 500 clients two CPU cores should be sufficient, with 1000 clients four CPU cores should be provided.

Also note that opsi-depots put a strain on the opsi-configserver, which is significantly larger than that of a single client.

**3.3. Configuration requirements**

Your server and your network have to comply to the following requirements to install and work with opsi:
3.3.1. Valid DNS domain name

Your DNS domain name must consist of at least one domain and one top-level domain. In other words: the fully qualified domain name must contain at least one point. Furthermore, the top-level domain must consist of at least two characters.

Valid domain names are e.g.: 'domain.local', 'uib.de', 'subdomain.domain.de'. An invalid example: 'mydomain.d' because this is only one character at the top-level domain. An invalid example: 'mydomain' because this is only a top-level domain.

see also:

• https://en.wikipedia.org/wiki/Domain_name
• https://en.wikipedia.org/wiki/Hostname

3.3.2. Valid DNS hostname

The hostnames (also of the clients) must comply with the guidelines. This includes, for example, that they must not contain any underscores.

Make sure that at your opsi-server, returns a 'fully qualified domainname', in which at least come two dots, e.g. 'opsiserver.domain.local':

```bash
hostname -f
```

Should the result not look like this (e.g. '127.0.0.1' or 'localhost') then you check your '/etc/hosts' directory or the name resolution first.

see also:

* https://en.wikipedia.org/wiki/Hostname

3.3.3. Correct name resolution for the server

Check the entry for the opsi-server in the file '/etc/hosts', or check the output of:

```bash
getent hosts $(hostname -f)
```

The result should look like the following example:

'192.168.1.1 server.domain.tld server'

Here the IP address should belong to the network interface, to which the Clients will be connecting.

If the result looks different from the above example (contains eg. '127.0.0.1' or 'localhost'), or the full qualified hostname does not contain one or more dots, then you must correct your name resolution (DNS or /etc/hosts file).

The names must be in accordance of the rules of a DNS system but a DNS server is not required for the usage of opsi.

opsi does not require Active Directory or similar. Integrating opsi is possible but not required.

3.3.4. Localization settings

opsi requires configured language settings ('locale') on the server. It is recommended to use an UTF-8 compatible locale.

The following command performs a simplified check:
If the output is `ok` locales are set. If the output is `check locales:` you should check if the following list has settings for `LANG` or `LC_ALL` that are according to your used language.

For English we recommend `en_GB.UTF-8` or `en_US.UTF-8`.

The following commands show how these settings can be changed if nothing or an undesired value is set:

```bash
sudo locale-gen en_GB.UTF-8
update-locale LANG=en_GB.UTF-8
```

To apply these settings systemwide the server should be restarted.

For more information please consult the documentation of your Linux distribution.

### 3.4. Needed network ports

This is an overview of the used ports and network protocols.

Config server = central opsi-server to manage all clients.
Depot server = opsi-server in the individual locations.
If there is only one opsi-server, it takes both roles.

- opsi-server web service: 4447/tcp +. Client to config and depot server, depot server to config server, also connections via localhost.
- opsi-client web service: 4441/tcp
  Config server to client, connection from client to itself via localhost.
- opsi-client Notifier: 45000-45099/tcp
  Connection from client to itself via localhost.
  A random port from the given range is selected.
- TFTP: 69/udp
  Connection from the client to the depot server.
- TFTP: 1024-65535/udp
  Connection from the depot server to the client.
- CIFS/SMB: 445/tcp
  Client to depot server.
- CIFS/SMB: 445/tcp + connection from Depot server to client.
  Only necessary if opsi-deploy-client-agent (winexe, smbclient) is used for Windows clients.
- Grafana web service: 3000/tcp
  Connection from client to config server.
  Only if the management web page of the config server is to be accessed from the client.
- SSH: 22/tcp
  Connection from client to Config server.
  Management access to Config and Depot server.
- SSH: 22/tcp
  Connection from depot server to client.
  Only necessary if opsi-deploy-client-agent is used for Linux / MacOS clients.
- DNS: 53/tcp, 53/udp +. All servers / clients must be able to reach your DNS.
- Wake-on-LAN (WoL): 7/udp, 9/udp, 12287/udp
3.5. Proxy settings

If a HTTP-proxy is used, the proxy settings should be configured system-wide using environment variables. These environment variables are usually stored in the `/etc/environment` file. The following environment variables should be created, all lowercase.

### 3.5.1. http_proxy

The environment variable `http_proxy` configures which proxy should be used for HTTP connections. A complete URL should be used, not only hostname and port. If the proxy requires authentication, this can be specified in the URL.

```
http_proxy=http://<user>:<password>@<proxy-address>:<port>
```

Example: `http_proxy=http://10.10.10.10:3128`

### 3.5.2. https_proxy

Like `http_proxy`, but for HTTPS connections.

Example: `https_proxy=https://10.10.10.10:3128`

### 3.5.3. no_proxy

The environment variable `no_proxy` defines for which addresses no proxy should be used. Multiple addresses are separated by commas. The following rules should be observed for the individual addresses: * Use lower case throughout. * Only IP addresses should be used if access is also made directly to the IP address. No name resolution takes place when evaluating the exceptions. * No IP address ranges (CIDR matching, such as: 192.168.0.0/24) may be used. * Exceptions must also be defined for local addresses and loopback addresses, such as `localhost` and `127.0.0.1`. * No wildcards, such as `*` or regular expressions, may be used. * Each name is evaluated as a suffix, so `domain.com` defines an exception for all hostnames ending with `domain.com`. * For each address, optionally, a port can be specified after a colon. Then the exception applies only to the specified port.

Example: `no_proxy=127.0.0.1,localhost,mydomain.example,hostname.domain.com:8080`. 
4. opsi-server Installation

This chapter describes the installation and configuration of an opsi server.

After you have worked through this chapter, you have a functioning opsi server. This serves as a basis for all further chapters.

In the following chapters we assume that you have a working network configuration on your server.

4.1. opsi-server Basic installation

In this section different variants of the installation of an 'opsi-server' are shown. You will end up with a server system ready for final configuration and commissioning. To evaluate opsi we recommend using the pre-installed virtual machine. Otherwise, you should select the operating system you are most familiar with. In this case please make sure that the packages of the server are up to date.

If a proxy server is used in your network to access the Internet, remember to enter this on your opsi-server as well. In particular the environment variables `http_proxy` and `https_proxy`.

In case of problems you can check the free support provided by the community.

4.1.1. Starting up the uib preconfigured Virtual Machine

An 'opsi-server' can be installed as a virtual machine, because the load on the system is low. A ready-to-use and pre-configured virtual machine is provided by uib. You can download the VMware or Virtualbox files from the uib website or opsi.org. The free of charge VMware player or Virtualbox is sufficient to run this machine. You may also use VMware server or ESXi.

First Start

**VMware**

If you have a server running VMware or VMware player, it only takes a few mouse clicks to install a base 'opsi-server':

- Download the opsi server VM from opsi.org
- Unzip the file and a directory 'opsivm' will be generated.
- Start VMware player. Open "Open a Virtual Machine", look for the directory `opsivm` and in it the file `opsivm.ovf` in its file selection dialog. You may have to change the file types to be displayed to ovf. You can now import the server under its own name. The virtual server can then be started.

**ESXi-Server**

- Download the opsi server VM from opsi.org
- Unzip the file and a directory 'opsivm' will be generated.
- Start vSphere Client.
  - Install a new client with 'File' / 'Deploy OVF Template....' and answer the next questions.

**Virtualbox**

- Download the opsi server VM from opsi.org
- Unzip the file and a directory 'opsivm' will be generated.
- Start Virtualbox.
  - At the menu 'File' / 'Import Appliance' select your `opsivm.ovf` file and import it.

**General**
The VMware player is free of charge and available for all common operating systems at vmware.com. Usually it can be installed without any problems, as long as the resources of the host computer (especially memory) meet the needs of running software systems in parallel.

**Language selection**

The first step is to choose the preferred language:

![Language selection](image)

*Figure 1. Language selection*

**First boot**

The opsi-server needs to be connected to the Internet to work properly. The script `1stboot.py` will automatically start at the first boot in order to configure the opsi-server network settings. If something goes wrong while running '1stboot.py', then you may run `1stboot.py` again from the command line.

The log file of `1stboot.py` is located at `/var/lib/1stboot/1stboot.log`.

⚠️ You cannot use `1stboot.py` to rename your 'opsi-server' afterwards!

![Startup mask](image)

*Figure 2. Startup mask*

Fill in the configuration information for your network and answer the questions.

![Input mask](image)

*Figure 3. Input mask*
In the following, you will be asked for:

**server name**
Name of this server (without domain) e.g. opsidemo

**domain**
DNS-Domain (not Windows-Domain) the name has to include a dot e.g. opsi.local

**ip address**
Address of this server e.g. 192.168.1.50

**netmask**
Netmask of this server e.g. 255.255.255.0

**windows domain**
Name of the Windows Domain (not the DNS domain)

**gateway**
IP-address of the Internet gateway e.g. 192.168.1.1

**proxy**
If required for Internet access, the proxy information: e.g. http://myuser:mypass@192.168.1.5:8080

**DNS server**
IP address of the name server e.g. 192.168.1.1

**mail relay**
IP address of the mail server e.g. 192.168.1.1

**tftp server**
IP address of the tftp server (usually the server)

**Password of root**
Password of root

**Password of adminuser**
Password of local opsi-admin.

After the program '1stboot.py' finishes, the virtual machine will be rebooted.

**Second Start**

After the reboot, or after completing the network configuration, login as 'adminuser' with your password.

The graphical user interface of the opsi-server should have already started (a lightweight window manager is used). A "Firefox" browser window appears at startup, and displays this document and further information.

If you get a message that there is no network connection, this might be caused by the special configuration of the virtual appliance. Before trying other options, you should reboot the server again. (i.e. use the shutdown button in the GUI)
If the network was correctly configured in the previous steps, then you should be able to remotely access the opsi-server, for example:

- use ‘ssh’ at the command line to access the server (‘ssh’ should already be installed on linux systems, for Windows use putty)
  Use ‘root’ as the user name, and authenticate with the root password.

**Terminal Window**

In the following sections, some commands have to be entered into a command line interface. It may be the easiest way to work through these instructions.

The commands are input into a window called a “terminal window”. Here are examples that explain how to access a terminal window:

- Remote access per ssh on the opsi-server (see previous section)
- Open a terminal window in the opsi-server graphical interface with a click on the terminal icon in the icon bar.
- Open a terminal window in the opsi-server graphical interface with a right mouse click inside the interface, and choose “Terminal”.
  Note: the graphical interface has multiple desktops that are reachable using the variety of buttons in the upper-left-hand corner of the display.

We recommend cutting and pasting commands from this handbook directly into the opsi-server terminal window (most applications support cut and paste).

Example snippets from configuration files are formatted like this:
depoturl = smb://smbhost/sharename/path

Example snippets for commands that you have to execute are formatted like this:

```
cd /tmp
ls -l
```

Angle brackets '<>' mark abstract names. When entering commands, please replace the '<abstract name>' with a real name. For example: The file share, where opsi places the software packages, may abstractly be noted as '<opsi-depot-share>'. If the real file share is /var/lib/opsi/depot, then you have to replace the abstract name by this exact string. The location of the package '<opsi-depot-share>/ooffice' becomes /var/lib/opsi/depot/ooffice.

### Check the Network Connection

If the network configuration is correct, and the computer is connected to the Internet, then you can access any website using the browser in the start window.

If not everything is working, then you have to open a terminal window (maybe this is not yet possible from a remote connection, only from the server GUI) and then perform the necessary network connection checks and fixes.

You can re-enter the network configuration by entering this command in the terminal window:

```
1stboot.py
```

A reboot is forced with the command:

```
reboot
```

If the network connection works, then you can install opsi packages or update them, and configure the environment for the first installation test. If you want to use the virtual machine (and not install the opsi-server directly to your host system), then skip to Update and Configuration of the opsi-server.

### Update the opsi-Server

To update your opsi-server you need to double click the Icon 'Update OS' on the desktop. To do this please enter the current password for the adminuser and confirm if necessary.

If necessary for your Internet access, adapt the file /etc/apt/apt.conf to your network circumstances (enter correct proxy or comment / delete line). You can edit these using the program any text editor for example, 'midnight commander':

```
mcedit /etc/apt/apt.conf
```

### Install the standard opsi-products

By performing a double click the Icon 'First package installation' the minimal opsi-products will be installed. To do this please enter the current password for the adminuser. This automatically fetches the current opsi packages, including templates for OS deployments, from the opsi repositories and installs them on the server.

For more information see Importing the minimal opsi products.
Starting opsi-Server Interface

You can start the management interface by double clicking on the icon `Opsi Configuration Editor`. For a description of the management interface check Installation of the management interface opsi-config.

You have a running opsi server now, i.e. the opsi application itself is fully configured.

You can now proceed with:

- Integration of existing Windows clients
- Installation of a new Windows PC with opsi (OS Installation)

4.1.2. Opsi-QuickInstall

Opsi-QuickInstall is a program to quickly and easily install an opsi-server on the following Linux distributions (We give the distribution names here according to the lsb_release output since these are the names Opsi-QuickInstall works with):

**QuickInstall 4.2.0.7-21**
- AlmaLinux 8,
- Debian 9, Debian 10, Debian 11,
- openSUSE 15.2, openSUSE 15.3, openSUSE 15.4,
- RedHatEnterprise 8,
- Rocky 8,
- SUSE 15.1, SUSE 15.2,
- Ubuntu 18.04, Ubuntu 20.04, Ubuntu 22.04,
- Univention 4.4, Univention 5.0

**QuickInstall 4.2.0.7-2**
- AlmaLinux 8,
- Debian 9, Debian 10, Debian 11,
- openSUSE 15.2, openSUSE 15.3,
- RedHatEnterprise 8,
- Rocky 8,
- SUSE 15.1, SUSE 15.2,
- Ubuntu 18.04, Ubuntu 20.04, Ubuntu 22.04,
- Univention 4.4, Univention 5.0

**QuickInstall 4.2.0.6-3**
- Debian 9, Debian 10, Debian 11,
- openSUSE 15.1, openSUSE 15.3,
- SUSE 15.1, SUSE 15.2,
- Ubuntu 18.04, Ubuntu 20.04,
- Univention 4.4

**QuickInstall 4.2.0.1-2**
- Debian 9, Debian 10, Debian 11,
- openSUSE 15.1, openSUSE 15.2, openSUSE 15.3,
4. opsi-server Installation

- SUSE 15.1, SUSE 15.2,
- Ubuntu 18.04, Ubuntu 20.04,
- Univention 4.4

**Preparation**

Check the entry for the opsi-server in the file `/etc/hosts`, or check the output of:

```
getent hosts $(hostname -f)
```

The result should look like the following example:

`'192.168.1.1 server.domain.tld server`

Here the IP address should belong to the network interface, to which the Clients will be connecting.

If the result looks different from the above example (contains eg. `'127.0.0.1` or `'localhost`), or the full qualified hostname does not contain one or more dots, then you must correct your name resolution (DNS or `/etc/hosts` file).


**Download and first Start**

You can download Opsi-QuickInstall as zip-file under the following link: [https://download.uib.de/opsi4.2/stable/quickinstall/](https://download.uib.de/opsi4.2/stable/quickinstall/).

Unzip the file and open the folder `opsi-quickinstall`. Now you must decide whether you want to execute the installation with or without graphical user interface.

Both versions are described below.

You can find more information on the properties QuickInstall asks for in the opsi-manual under chapter "9.5.10.1 The product l-opsi-server": [https://download.uib.de/opsi4.2/stable/documentation/opsi-manual-v4.2-en.pdf](https://download.uib.de/opsi4.2/stable/documentation/opsi-manual-v4.2-en.pdf). There you also find the default values of the properties. Different from the manual is only the default-value of the property `allow_reboot`; this one is QuickInstall `false`.

**Opsi-Quickinstall GUI-Version**

Open the folder `gui` and execute the file `opsi_quick_install_project` (for example per double click).

A window appears in which you can first select the language for Opsi-QuickInstall and the type of installation:
In the custom installation you can make more detailed settings.

Click on 'next>' and answer the questions. For some questions you will find information signs on the right hand side. These can give you more information about the question on mouse click.

The questions on name and password of the opsi admin user give example values (also shown on the image Information). For security
reasons you should change these values. Do NOT use the examples!

After the queries, QuickInstall will show you an overview where you can check all your answers. If everything is correct, click ‘finish’, type in your password and click ‘finish’ again. Then the installation of the opsi-server will start.

![Installation](image)

**Figure 7. Installation**

The installation may take some minutes. In the end, QuickInstall shows you whether it was successful.

If the result is 'failed', you can search in the log files for the error or, if you have a support contract, you can directly contact uib.

### Opsi-Quickinstall No-GUI-Version

#### Start

Open the folder ‘nogui’ and execute the file ‘opsi_quick_install_project’ with one of the following parameters on the console as root:

- `-d`, to use the default values for the installation of the opsi-server and immediately start the installation (IMPORTANT: Through that, QuickInstall will also create the opsi admin user with example values for the name and password, which are 'Alexandra'(Opsi-QuickInstall version 4.2.0.1) or 'adminuser'(from Opsi-QuickInstall version 4.2.0.1-2 on) as name and ‘linux123’ as password. For security reasons you should change these values afterwards!),

- `-f <file>`, to use the values from a file for the installation of the opsi-server and immediately start the installation,

- `-n`, (recommended) to start a setup program on the console, in which you can set the values for the installation seperately.

So for example execute

```
sudo ./opsi_quick_install_project -n
```

The operation of the setup program is shortly described in the following.
Setup Program

If you chose the parameter `-n`, answer the questions that are asked. On each question you also have the possibility to type in one of the following commands:

- `-b`, to jump back to the previous question,
- `-h`, (only for questions that are marked with a `*` at the end) to get further information on this question,
- type nothing in and press Enter to use the default value for this question.

Afterwards QuickInstall will show you an overview where you can check all your answers. If everything is correct, click Enter to start the installation of the opsi-server.

Installation of the opsi-server

The installation of the opsi-server may take some minutes. In the end, QuickInstall shows you whether the installation was successful.

![installation result]

Figure 9. Result


If the result is 'failed', you can search in the log files for the error or, if you have a support contract, you can directly contact uib.

4.1.3. Prerequisites for an installation on a server

From version 4.2 the opsi-server needs access to a Redis and a Grafana instance. If these services will also be provided by the opsi server, we recommend switching to the `opsi-server-full` package during the migration. This package installs and configures everything that is necessary on the opsi server (this will be referred to as a single server setup). This recommendation applies to all supported operating systems, except for Univention UCS. As previously, the `opsi4ucs` package should be installed on these systems.

The `opsi-server-full` package installs all necessary components to run opsi on one server.
If certain components should not be installed, for example if Redis, MySQL or Grafana should run on another server, the packages `opsi-server` or `opsi-server-expert` can be used instead.

We recommend using the official Grafana repositories for Grafana:

Debian/Ubuntu/UCS:
If you want to use mysql instead of mariadb, then you must specify a user with mysql_native_password when using opsi-setup --configure-mysql.

To activate mysql_native_password for the root user, the following steps are necessary:

- enter skip-grant-tables in the mysql configuration under [mysqld].
- restart mysql service
- log in as root with mysql-u root -p
- flush privileges; and
- ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'NewPassword'; execute.
- Remove skip-grant-tables again and restart the service.
4.1.4. Installation on Debian / Ubuntu

In this chapter, we assume you are familiar with the Debian package system (you will find information about this in the appropriate Debian books, in the manual pages, or under Debian documentation).

Please check the requirements and preparations!

We recommend to install the following packages:

```
apt install host pigz apt-transport-https software-properties-common curl gpg
mkdir -p /usr/local/share/keyrings
```

Furthermore, samba needs to be installed:

```
apt install samba samba-common smbclient cifs-utils
```

To start with the installation of opsi add the opsi repository to apt:

**Ubuntu 22.04 LTS Jammy Jellyfish:**

```
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/xUbuntu_22.04
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

**Ubuntu 20.04 LTS Focal Fossa:**

```
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/xUbuntu_20.04
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

**Ubuntu 18.04 LTS Bionic Beaver:**

```
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/xUbuntu_18.04
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

**Debian 11 Bullseye:**

```
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/Debian_11
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

**Debian 10 Buster:**
Debian 9 Stretch:

```bash
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/Debian_9.0
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

Check for success of the key import:

```bash
gpg /usr/local/share/keyrings/opsi.gpg 2>/dev/null
```

should contain the output:

```
pub   rsa2048 2017-09-30 [SC] [expires: 2023-11-09]
    2E98F7B5A5B2C8FE7F609705D1F933E608361F81
uid          home:uibmz:opsi OBS Project <home:uibmz:opsi@build.opensuse.org>
```

If necessary for your Internet access, adapt the file `/etc/apt/apt.conf` to your network circumstances (enter correct proxy or comment/delete line). You can edit these using the program any text editor for example, `midnight commander`:

```bash
mcedit /etc/apt/apt.conf
```

Before installing the opsi packages, make sure that your server has a valid FQDN:

```bash
hostname -f
```

The output FQDN must contain at least two dots (see: requirements).

Execute the following commands in order to install opsi on your server:

**Single server setup:**

```bash
apt update
apt install opsi-server-full
```

**Manual setup:**
If you are asked for the tftp directory during the tftpd-installation answer with /tftpboot.

Assuming all of the above steps completed successfully, we can assume that the network is properly configured. Next continue on with configuration.

### 4.1.5. Installation on a Univention Corporate Server (UCS)

Please check the configuration requirements and preparations!

The installation on a Univention Corporate Server is possible through the Univention App Center as well as the classic way by using the repositories maintained by ubi.

Both are equally supported methods of installations. We recommend using only one method per server. If new packages for an operating system are released they are available right away if the repositories maintained by ubi are used. If the installation is made through the App Center the change to a newer UCS version (i.e. from UCS 4.4 to UCS 5) will be blocked until all installed apps are available for the new version of the operating system.

With opsi 4.2 the ucs support was adapted to the opsi-standard like on other supported distributions. The function of opsi4ucs was implemented in opsi-server package and its variants. The opsi4ucs package exists in opsi 4.2 as a transition package to make the migration easier. This package will automatically removed during the upgrade process.

The first opsi-server in an environment will have its backend configured to make use of the installed MySQL server. All subsequent servers will be registered as depots in opsi.

### Manual opsi-installation on UCS (without App-Center)

Please check the requirements and preparations!

Necessary preparations:

- Samba has to be configured. For the use on a server with the 'member' role, univention-samba has to be used instead of univention-samba4.
- univention-mariadb or univention-mysql has to be installed.
- If the machine should also work as DHCP-server, then the dhcpd daemon has to be configured and should be running.

The installation of opsi is possible on a server with the roles 'master', 'backup', 'slave' or 'member'. For the installation on a 'member' you need to read Hints about installing opsi on an UCS server with the role 'member'!

The following documentation describes an installation on a 'master' with Samba 4.

When installing on a 'slave' the server must be already joined to the 'master' and Samba 4 has to be installed first.

UCS configuration is usually done on the 'master' while the installation and configuration of opsi takes place on the 'slave'.

The classic installation with the user 'pcpatch' in the primary group 'pcpatch' cannot be adhered to with UCS. Samba 4 has the same
fundamental restrictions as Active-Directory, so groups with the same name as a user are not allowed. For this reason the
configuration file `/etc/opsi/opsi.conf` has been introduced for UCS 3. This file controls how the group used for the Samba shares will
be named. Since UCS 3 the group name ‘pcpatch’ will be renamed to ‘opsifileadmins’ with this file. This means that users that need
rights for opsi (opsi package builders for example) should not be members of the group ‘pcpatch’ but must be members of the group
‘opsifileadmins’. This peculiarity applies only to UCS and is different to other distributions and different to the next chapters in the
opsi-documentation. With UCS the user ‘pcpatch’ is created as a full domain user. For more information about this new configuration
file please refer to the opsi-manual.

• Next add the opsi4ucs repository:

Univention UCS 4.4:

```
REPO_URL=https://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/Univention_4.4
REPO_KEY=/usr/local/share/keyrings/opsi.gpg
echo "deb [signed-by=${REPO_KEY}] ${REPO_URL}/ /" > /etc/apt/sources.list.d/opsi.list
curl -fsSL ${REPO_URL}/Release.key | gpg --dearmor | sudo tee ${REPO_KEY} > /dev/null
```

For installation the following commands must be entered next:

**Single server setup:**

```
univention-install opsi-server-full
```

**Manual setup:**

```
univention-install redis-server redis-timeseries grafana
systemctl daemon-reload
systemctl enable grafana-server
systemctl start grafana-server
univention-install opsi-server
```

If the role of the target system different than ‘master’ or ‘backup’ then we have to run the opsi4ucs Join-Script:

```
univention-run-join-scripts
```

A link to the management interface can be found at the URL `https://<servername>:4447`.

To use the opsi configuration editor the user has to be a member of the group opsiadmin. The group membership can be edited by
using Univention-Admin. The user Administrator will automatically be added to this group during the opsi installation.

Finally, in UDM, for the ‘opsi_depot’-share we have to set the following option under Advanced Settings → Advanced Samba Settings:
‘follow symlinks’ must be set to ‘yes’. The same should be done for the ‘opsi_depot_rw’-share, so the driver integration will run
without problems. If the directory `/var/lib/opsi/depot` is located on an extra partition or hard disk then the option for wide links
should be set to ‘yes’.

To make sure that opsi is running with the proper settings restart opsi by entering the following commands:

```
opsi-setup --init-current-config
opsi-set-rights
systemctl restart opsiconfd.service
systemctl restart opsipxeconfd.service
```
Please be advised that samba 4 will not be automatically restarted, since it is an important service on which other software may depend. You have to restart it manually. After restarting samba there may be a slight delay before the new shares are accessible.

Because there is no direct connection between the Univention LDAP and the opsi-backend all Clients have to be created twice. First in the Univention-LDAP using UDM and then in opsi including all system information (in particular the MAC address). Deleting a LDAP client in Univention will not delete the client in opsi and vice versa. This problem is further discussed in Synchronising data from LDAP to opsi.

Since opsi was installed on an existing server we assume that the network configuration is correct. Continue with the installation by skipping forward to configuration.

Hints about installing opsi on an UCS server with the role 'member'

Installing opsi on a server with the role 'member' is possible.

After an installation you need to make sure that the user that will be used to access the depot exists in the current domain. Check the host parameter `clientconfig.depot.user` for this. Let's assume that the domain is `backstage`, then the value has to be `backstage\pcpatch`. If it is `memberserver\pcpatch` then it has to be changed.

Setting the password for the user `pcpatch` through `opsi-admin` fails because of the missing AD write access of a 'member' server. To change the password you have to do so additionally on a server with write access - a 'master', 'backup' or 'slave'.

PXE-Boot configuration for operating system installation

If the PXE-Boot should be used for OS installations the DHCP-service on the relevant UCS-System has to be reconfigured. There are two characteristics which differentiate UCS from other supported distributions.

- The configuration is not made automatically during the opsi installation on an active UCS infrastructure because often the configuration is already in use.
- The opsi-tftp-hpa is not configured as usual using the directory `/tftpboot` as base directory, instead the `/var/lib/univention-client-boot` is used. All important files of opsi-linux-bootimage will be moved from `/tftpboot` to the base directory. The side effect is that the DHCP-Option filename must be `pxelinux.0` instead of `linux/pxelinux.0`.

To implement these settings, a policy must be created in the UCS system. This policy interacts with the existing policies, and has to be implemented appropriately. If opsi was installed on an UCS test system without existing policies, check if the DHCP-service is installed. If the DHCP-service is already installed the easiest way to create the policy is in the UMC-webinterface (Univention Management Console) of the UCS-server. To do this choose the category "Domain" and underneath the module DHCP-server. Next you have to choose the service (in a testing system you will usually find only one entry). In the following view choose the menuitem policies. The policy we need is a DHCP-Boot policy. In the policy configuration choose `cn=default-settings` as default entry (there should be only one entry) and choose 'edit'. Under basic settings - DHCP-boot enter for the bootserver option the IP address of the opsi-server and enter for the boot-filename option `pxelinux.0`.

If the policy is configured like mentioned above, this affects every device that uses DHCP from this server. Therefore, this instruction is meant only for testing opsi and UCS together. In a productive UCS environment you should not configure this policy as described previously.

Optionally, these settings can be done at the console with the udm command. You can find more information about this in the UCS-documentation.

Synchronising data from LDAP to opsi

In an opsi4ucs installation Windows clients have to be created in the UDM first and then they have to be created in opsi-configured. Changes to the client in UDM will not be passed on to opsi. For example if a client's MAC address changes in LDAP and in opsi a
netboot-product is set to setup, the boot configuration would be provided with an incorrect MAC address.

The solution for this is the extension 'opsi-directory-connector'. Please consult the manual for more information.

4.1.6. Installation on openSUSE or Suse Linux Enterprise Server (SLES)

Please check the requirements and preparations!

Necessary preparations:

- Samba must be installed and configured.
- mariadb-server must be installed.
- If the machine should also act as DHCP-server then the dhcpd daemon has to be configured and running.

You can use zypper to add the opsi repositories:

**openSUSE Leap 15.1:**

```bash
zypper addrepo
```

**openSUSE Leap 15.2:**

```bash
zypper addrepo
```

**openSUSE Leap 15.3:**

```bash
zypper addrepo
```

**SLES 15SP1:**

```bash
zypper addrepo
```

**SLES 15SP2:**

```bash
zypper addrepo
```

**SLES 15SP3:**

```bash
zypper addrepo
```

After adding the repository, the installation can be started:
Single server setup:

zypper refresh
   Do you want to (r)eject the Key, (t)emporary or (a)lways trust? [r/t/a/?] (a): a
zypper -v install opsi-server-full

Manual setup:

zypper refresh
zypper install redis-server redis-timeseries grafana
systemctl daemon-reload
systemctl enable grafana-server
systemctl start grafana-server
zypper -v install opsi-server
zypper -v install opsi-windows-support

Please make sure that your firewall configuration allows connections to the following ports:

- tftp: 69/UDP
- opsi: 4447/TCP and 4441/TCP

In case you used an utility like yast or autoyast to help you with your network configuration it is possible the tool created an entry in the /etc/hosts file like:

127.0.0.2 <fqdn> <hostname>

If you want to leave the configuration of the DHCP server to opsi, this entry has to be changed to the public IP address of the server.

Please continue with configuration.

⚠️ The unix commands used in the following chapters are based on Debian systems. You may have to adapt them to the corresponding commands for your linux system.

4.1.7. Installation on CentOS or RedHat Enterprise Linux (RHEL)

The installation of opsi on CentOS, Red Hat Enterprise Linux (RHEL), Alma Linux or Rocky Linux differs only by the used repository.

⚠️ Please check the configuration requirements and preparations!

When using Red Hat Enterprise Linux, you must register with the Red Hat Network to have access to all required packages in the Red Hat repositories:

subscription-manager register
subscription-manager attach --auto

Necessary preparations:

- Install Samba and the database:
yum install mariadb-server samba samba-client

- Configure samba and database:

  systemctl start smb.service
  systemctl start nmb.service
  systemctl start mariadb.service
  systemctl enable smb.service
  systemctl enable nmb.service
  systemctl enable mariadb.service
  mysql_secure_installation

- If the machine should also act as DHCP-server then the dhcpd daemon has to be configured and running.

Add the repositories:

CentOS 8:

  cd /etc/yum.repos.d/
  yum makecache

RHEL 8:

  cd /etc/yum.repos.d/
  yum makecache

Alma 8:

  cd /etc/yum.repos.d/
  yum makecache

Rocky 8:

  cd /etc/yum.repos.d/
  yum makecache

After adding the repository you may start the opsi installation:

Single server setup:

  yum install opsi-server-full

Manual setup:
You may be asked to import the GPG key of the repository. The message is pretty similar to the following one:

```
Importing GPG key 0xD8361F81 "home:uibmz OBS Project <home:uibmz@build.opensuse.org>" from
http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.2:/stable/CentOS_8/repodata/repomd.xml.key
Is this ok [y/N]: y
```

Please answer with ‘y’.

Please make sure that your iptables and SELinux configuration allow access to the following ports:

- tftp: 69/UDP
- opsi: 4447/TCP and 4441/TCP

Assuming all of the previous steps were completed successfully we can assume that the network is properly configured. Next continue with configuration.

4.2. Update and Configuration of the opsi-server

In this chapter, the installed opsi-server is configured.

4.2.1. Proxy Entry in apt-configuration File

If necessary please adapt the file `/etc/apt/apt.conf` to your network configuration (enter the correct proxy or comment/delete unnecessary lines). You can edit your file with a program like midnight commander:

```
mcedit /etc/apt/apt.conf
```

4.2.2. Update of the opsi-server

Bring the opsi-server up to date by executing the following commands one after the other in a terminal window:

```
apt update
apt upgrade
```

If you are asked during the update whether the `smb.conf` should be overwritten, you have to confirm this. If the `smb.conf` has already been changed, you should keep the default and compare the files later. If this question has already been answered with no, you can do this later on the opsi-server by running `opsi-setup --auto -configure-samba`.

The unix commands used in the following chapters are based on Debian systems. You may have to adapt them to match your CentOS/RHEL/Alma/Rocky system.
4.2.3. Backend Configuration

Opsi supports different backends for data storage.

These are essentially:

- **file** - data storage in files
- **mysql** - data storage in a MySQL database

Besides these there are some backends for special purposes:

- **opsipxeconfd** - the service used for network booting with opsi
- **dhcpd** - used for configuring and restarting the dhcp service on the opsi-server
- **jsonrpc** - for forwarding all requests to another server

By default the mysql-Backend is used for inventory data. The usage of the file-Backend for inventory data is possible but noticeably slower and therefore not recommended.

The use of the mysql-Backend for inventory data is free and does not require activation. More information about the activation can be found in the opsi manual.

Some distributions use MariaDB instead of MySQL. The mysql backend also functions with MariaDB.

We will now configure the mysql-Backend. It is assumed that a MySQL server is installed and configured, and that the credentials for a database administrator are known. For specific information on installation and configuration of the database please refer to the manuals of your distribution.

For the initial configuration of the mysql-Backend use the command:

```
opsi-setup --configure-mysql
```

The command will ask for the credentials for database access, to create a database for opsi and to create an user with appropriate rights to access that database.

The following screenshots show examples for the MySQL configuration setup:

![MySQL config](image)

*Figure 10. Dialog opsi-setup --configure-mysql: Input mask*
Connecting to host 'localhost' as user 'root'
Successfully connected to host 'localhost' as user 'root'
Creating database 'opsi'
Database 'opsi' created
Creating user 'opsi' and granting all rights on 'opsi'
User 'opsi' created and privileges set
Testing connection to database 'opsi' as user 'opsi'
Successfully connected to host 'localhost' as user 'opsi'
Updating backend config '/etc/opsi/backends/mysql.conf'
Backend config '/etc/opsi/backends/mysql.conf' updated
Initializing mysql backend

Figure 11. Output: opsi-setup --configure-mysql: Output

You may accept the defaults for all questions except the 'Database Admin Password'. The 'Database Admin Password' is linux123 on the pre-installed opsi-VM, otherwise it is the password you entered during the mysql-server installation.

Different kinds of data may be stored in different types of backends. For some actions (such as method calls) more than one backend is involved. For this purpose, the different opsi method calls are assigned to the backends. This is configured in the file /etc/opsi/backendManager/dispatch.conf.

Here an example:

```bash
# = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
# =      backend dispatch configuration                                     =
# = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
#
# This file configures which methods are dispatched to which backends.
# Entries has to follow the form:
# <regular expression to match method name(s)> : <comma separated list of backend name(s)>
#
# Backend names have to match a backend configuration
# file basename <backend name>.conf beneath /etc/opsi/backends.
# For every method executed on backend dispatcher
# the first matching regular expression will be decisive.
#
# Recommended standard configuration (dhcpd not at the opsi server)
# file as main backend, mysql as hw/sw invent
# and license management backend and opsipxeconfd backend:
backend_.*    : file, mysql, opsipxeconfd
host_.*       : file, opsipxeconfd
productOnClient_.* : file, opsipxeconfd
configState_.* : file, opsipxeconfd
license.*     : mysql
softwarelicense.* : mysql
audit.*       : mysql
.*             : file
```

At the top of this file information and examples are given. In the first column is the name of the opsi method being called (with wildcard .) and after the colon is the list of backends used by that opsi method. For every called method procedure the first column of this list is checked to determine which backend has to be used. The first line that matches the method name is used. The last line . matches all opsi method calls.

The default configuration after the installation is the usage of the file-Backend as main backend and the mysql-Backend for license management and inventory data.
Make sure that all used backends are listed in the line starting with \texttt{backend_*}.

Whenever the file \texttt{dispatch.conf} is changed, the following commands should be executed. Even if you have not changed the file during the initial setup execute these commands now.

\begin{verbatim}
opsi-setup --init-current-config
opsi-set-rights
systemctl restart opsiconfd.service
systemctl restart opsipxeconfd.service
\end{verbatim}

\section*{4.2.4. Set Samba Configuration and Change Passwords}

Opsi requires certain samba shares. To ensure that they are available please enter the following command:

\begin{verbatim}
opsi-setup --auto-configure-samba
\end{verbatim}

Please restart the samba services using the following commands:

\begin{verbatim}
systemctl restart smbd.service
systemctl restart nmbd.service
\end{verbatim}

If the server is updated and it asks if the file \texttt{smb.conf} should be overwritten, you have to confirm this. If the \texttt{smb.conf} has been customised before, you should keep the default and merge the files later. If this question has already been answered with no, you can repeat this later on the opsi-server by running \texttt{opsi-setup --auto-configure-samba}.

A 'pcpatch' pseudo-user is created on the system. Clients login with this user to install software and to get access to the installation files on the samba shares. The user 'pcpatch' must be created with a correct password - simultaneously as a system user, as a samba user and as an opsi user.

In a terminal window the program 'opsi-admin' should be executed, which will set the pcpatch-password (for the opsi, unix and samba user).

\begin{verbatim}
opsi-admin -d task setPcpatchPassword
\end{verbatim}

After executing the command you are asked to enter the password.

\section*{4.2.5. Create users and configure the groups opsiadmin and opsifileadmins}

Administrative control of opsi is only allowed for members of the UNIX-group 'opsiadmin'.

In the following example, we create the user 'adminuser'.

Firstly we create the user:

\begin{verbatim}
useradd -m -s /bin/bash adminuser
\end{verbatim}

We then set the unix password:
and now the samba password:

```
    smbpasswd -a adminuser
```

Do not use the character '§' in the passwords, because this character is not permitted when connecting to the opsi service.

Now we create and test the group membership with these commands:

```
    usermod -aG opsiadmin adminuser
    getent group opsiadmin
```

The `getent` command should show a result like this:

```
    opsiadmin:x:1001:opsiconfd,adminuser
```

When 'root' is not a member of the opsiadmin, then 'root' will not be able to use all administrative opsi commands!

To perform everyday tasks on your opsi server, it is usually not necessary to be logged in as 'root'. Our recommendation is to use a normal user and use the `sudo` command whenever administrative privileges are required.

All users who build opsi packages (opsi-makepackage), install opsi packages (opsi-package-manager), or manually edit the configuration files also have to be members of the group 'opsifileadmins':

```
    usermod -aG opsifileadmins adminuser
```

Test the results by entering:

```
    getent group opsifileadmins
```

The result should look like

```
    opsifileadmins:x:998:adminuser
```

To make `sudo opsi-set-rights` available for users of the group 'pcpatch', please execute:

```
    opsi-setup --patch-sudoers-file
```

Afterwards `opsi-set-rights`, which does the same as `opsi-setup --set-rights`, can be executed not only as root, but also with sudo by members of the group 'opsi-file-admins':

Example:
4.2.6. DHCP Configuration

A correctly working name resolution and DHCP are essential for the correctly functioning of opsi. To simplify the setup the opsi-server VM is supplied with a working DHCP server. On the other hand, in many environments there often already exists a DHCP server, which will be used with opsi. Both alternatives are described below.

Using a DHCP Server at the opsi-server

Using the opsi-Server VM:

The preconfigured opsi VM already has a DHCP server installed. The DHCP server on the opsi-server VM is configured with no free leases, so no unknown clients will get an IP address from this DHCP server.

If you create a client on the opsi-server using opsi-configed, you must supply the IP address and MAC address of the client. This will be entered into `/etc/dhcp/dhcpd.conf` and the DHCP service will be restarted.

Your own installation:

If you want to use the opsi server as a DHCP server, you have to install the corresponding DHCP server package.

e.g.

```bash
apt install isc-dhcp-server
```

After the installation the dhcp configuration file has to be adjusted. This is done by the following command:

```bash
opsi-setup --auto-configure-dhcpd
```

To restart the DHCP server, as described in `/etc/opsi/backends/dhcpd.conf`, an entry in `/etc/sudoers` is required. This is created using the command:

```bash
opsi-setup --patch-sudoers-file
```

The permissions for the dhcpd configuration file should look similar to this:

```
-rw-r--r-- 1 opsiconfd opsiadmin 80174 Dec 22 14:37 /etc/dhcp/dhcpd.conf
```

Using an External DHCP Server

Using the opsi-Server VM:

If you use an external DHCP server, then you can uninstall the DHCP server on the opsi-server.

This is done by entering this command:

```bash
apt remove isc-dhcp-server
```
Your own installation:

Since opsi 4.0.3 a DHCP server will not be installed automatically in this situation.

You have to configure the external DHCP server, so a PXE boot from the opsi-server is possible. If your external DHCP runs on Linux, then you need the following entries for the clients in the DHCP daemon configuration file (i.e. `etc/dhcp/dhcpd.conf`):

```
next-server <ip of opsi-server>;
filename "linux/pxelinux.0";
```

Replace `<ip of opsi-server>` with the IP address of your opsi-server.

If the opsi server runs on openSUSE or SLES, then `filename=opsi/pxelinux.0`.
If the opsi server runs on UCS, then `filename=pxelinux.0`.

If you are using a Windows DHCP server, then the corresponding entries are `bootserver (Option 66)` and `bootfile (Option 67)`.

If you create a client on the opsi-server, then you only have to supply the MAC-address, but not the IP address.

Checking the Backend Configuration for DHCP Entries

Regardless of whether or not you use an external DHCP server, the configuration of the opsi-server must be changed.

The file `/etc/opsi/backendManager/dispatch.conf` defines which backends are used (i.e. 'file', 'mysql').

The lines with the `backend_.*` and `host_.*` entries configure whether or not the opsi-server should work with the local DHCP configuration. If you are using the DHCP server on the opsi-server, then the backend dhcpd has to be added here. The corresponding entry with `file` backend must then look like this:

```
backend_.*         : file, opsipxeconfd, dhcpd
host_.*            : file, opsipxeconfd, dhcpd
```

If the local DHCP service on the opsi-server isn't used (because another server in the local network performs this task, and is also used for the opsi-clients), then the backend `dhcpd` is not required:

```
backend_.*         : file, opsipxeconfd
host_.*            : file, opsipxeconfd
```

After editing the backend configuration, the configuration has to be initialised and the opsiconfd service has to be restarted:

```
opsi-setup --init-current-config
opsi-set-rights
systemctl restart opsiconfd.service
systemctl restart opsipxeconfd.service
```

4.2.7. Configuration of the name resolution

To install software on the clients before login, generally only the clients have to know how to contact the opsi-server.

However, opsi also has a number of 'push' features such as 'on_demand' events, sending messages, starting remote control software, and retrieving session information.

For all these functions the server must be able to reach the client and therefore needs to determine the IP address of the client.
this works best depends on the specific configuration of DNS and DHCP. There are a large number of possible configurations.

Therefore we show two typical extremes:

1. The clients are not known by the DNS, and they have dynamically assigned frequently changing IP addresses.
2. The DNS always provides the correct IP address of a client.

To adapt the opsi server to different situations, you may change the following parameters:

- The entry resolveHostAddress in the file `/etc/opsi/backends/hostcontrol.conf`
  If this option is set to 'True', when connecting from the opsi-server to an opsi-client, the IP address of the client is first determined via the name resolution. To give preference to the IP address stored in the opsi backend, the option must be set to 'False'.

- The entry update ip in the file `/etc/opsi/opsiconfd.conf`
  If this entry is set to 'yes', whenever the opsi-server receives an IP address from a client (e.g. on every connection the client makes) the IP address stored in the backend will be updated. The default is 'yes'.

For the first variant, then you should probably set resolveHostAddress to 'False' and update ip to 'yes'.

For the second variant, then the best configuration is to set resolveHostAddress to 'True' and update ip to 'no'.

You should decide for yourself which combination fits your situation best.

If you changed anything in these files, then you should restart the opsiconfd:

```
systemctl restart opsiconfd.service
```

### 4.3. Importing the minimal opsi products

For deploying software with opsi ready-made packages are available. One of these contains the agent (`opsi-client-agent`), which must be installed on the clients to enable management.

It is possible to install the packages in automated or manual fashion. The automated way is recommended.

#### 4.3.1. Automatic import of the minimal opsi products

For the automatic installation of opsi products the `opsi-package-updater` tool is available, which as configured as in `/etc/opsi/opsi-package-updater.conf`, automatically fetches the current packages from the opsi repository and installs them on the server.

```
opsi-package-updater -v install
```

If a proxy is needed to access the internet, this may be entered in the `.repo` configuration files in `/etc/opsi/package-updater.repos.d/` as the value for `proxy`. Since opsi-utils version 4.1.1.33 a global proxy can be configured in `/etc/opsi/opsi-package-updater.conf`.

```
[repository_uib_windows]
...
proxy =
```

To later update the installed packages, this can be done with the following command:

```
More information on `opsi-package-updater` can be found in the manual.

Please note that OS installation products like win10-x64, are not immediately ready for use after installation. The installation has to be supplemented by the installation files from the corresponding installation media (i.e. DVD, see OS-Installation: Complete the Base Package for Windows).

### 4.3.2. Manual import of opsi products

There is also the option of manually downloading and installing the packages.


We recommend to save these `.opsi`-files in `/var/lib/opsi/repository`. To make sure opsiconfd is allowed to access these files run `opsi-set-rights /var/lib/opsi/repository`.

After the download you have to install the packages on your server with the command `opsi-package-manager -i <packagename>.opsi`. If the packages are stored under `/var/lib/opsi/repository`, the following command can be used for the initial installation:

```
opsi-package-manager --install /var/lib/opsi/repository/*.opsi
```
5. Management interface opsi-configed

Opsicomfiged offers a comfortable management interface. It communicates via HTTPS with the opsi server and can therefore be used on any computer that can establish a corresponding connection.

When using a virtual machine, make sure that the virtual screen has a large enough resolution. For opsi-configed a minimum resolution of 1024x768 pixels is required. To improve the graphics and mouse driver integration at a higher resolution, it is helpful to install the 'VMware Tools' on a VMware machine or the virtual guest additions on a VirtualBox machine.

5.1. Installation of the management interface opsi-configed

The management interface is installed as a local application on the administration PCs. In your web browser, go to the address https://<opsidepotserver>:4447/. There you will find links to installers for different operating systems.

Alternatively, you can find corresponding installers under https://download.uib.de/opsi4.2/misc/helper/.

The Windows installer must be executed with administrative rights. To do this, right click to open the context menu of the installer and then select 'Run as administrator'.

Once one PC is equipped with the management interface, further PCs can have easily have the interface Section 7.1, “Deploying opsi standard products: opsi-configed” installed with the localboot product opsi-configed, as long as the opsi agent is already installed on the PC.

5.2. Start of the management interface opsi-configed

Start opsi-configed via the shortcut in your Start menu.

Log in as a user who is a member of the group opsiadmin.

The operation of the management interface is pretty much self explanatory. You will find detailed instructions in the opsi manual.

Changes in the opsi management interface must be saved before they take effect and changes in the data must be retrieved from the server via the 'Reload data' button.
6. Adding clients to opsi

To be able to manage computers with opsi, they must be known to the opsi system. In addition, an agent must be running on these computers so that communication between the server and client is possible. No management is possible without this client agent.

Depending on the environment in which opsi is to be used, there are different procedures. If there are already clients in the environment with an installed operating system that are to be managed with opsi, they can be integrated in different ways.

The alternative to this is that the computers to be managed by opsi are equipped with a new operating system. As part of the installation of the operating system, the required agent is also installed by opsi. However, any previously installed software (including the operating system) will be removed. To use this procedure you first add a client to opsi and then perform an OS installation.

6.1. Creation of a new opsi client

To manage computers, they must be known to the opsi-server. This chapter describes different ways to create a client in opsi for later management. This is particularly helpful if you want to install an operating system on your computer using opsi.

For the integration of clients with an already installed operating system, please read the chapter integration of existing Clients.

6.1.1. Creating a new opsi client via the graphical management interface

A client can be added to the opsi-server through the opsi-configed graphical user interface.

From the menu, choose OpsiClient / Create new opsi client and enter:

- Client name
- DNS domain (if different from the default)
- Client description
- IP address (required if DNS can not be used resolve the address of the client)
- MAC address (required if the opsi-server is the DHCP server or if you want to use PXE boot with this client)

After completing the input, the client will be created on the opsi-server, and if the opsi-server is also the DHCP server, the client will also be created in the DHCP configuration, as a PXE client.

The list of configured opsi clients can be viewed at any time in the opsi-configed mode Client configuration under the clients tab.

6.1.2. Creating a new opsi client via the command line

A client can added through the command line using the tool opsi-admin.

The syntax is the following:

```
opsi-admin -d method host_createOpsiClient <client-id> [opsiHostKey] [description] [notes] [hardwareAddress] [ipAddress] [inventoryNumber] [oneTimePassword] [created] [lastSeen]
```

Missing values usually use a default value - most fields are then empty.

The following command will create the client testclient.domain.local with a random host key, the description Testclient, no notes, the MAC address of 00:0c:29:12:34:56 and the IP address 192.0.2.1.
6.1.3. Creating a new opsi client using the opsi-client-bootcd

On the download page of uib you will find various ISO images of the 'opsi-client-boot-cd' at [https://download.uib.de/opsi4.2/boot-cd/](https://download.uib.de/opsi4.2/boot-cd/). Download the latest and burn it to a CD.

Start the computer from the CD. You then should see the following screen:

![Figure 12. Start image opsi-client-boot-cd](image)

Choose Start **opsi (English)**. After a while, the following screen will appear. If your DHCP server assigns IP addresses to unknown DHCP clients, then most fields will already have valid values. Otherwise you have to complete the missing data by hand. You must at least give the hostname.
Then choose **OK**.

Then choose **Admin account**. This tells the client to register itself at the opsi-server using provided credentials.

Now you will get a login window, where you must authenticate yourself as a member of the opsiadmin group. If this was successful, then the client sends its data to the server, at which point the client will be created at the server. In the next step, the client asks the server for the list of available netboot products, and makes them available for you to choose from.
Now you may choose the operating system that you would like to install (or e.g. hwinvent).

### 6.2. Integration of existing Windows clients

To include existing Windows clients in opsi, the opsi-client-agent must be installed on them. This can be realised in several ways. After you have installed the opsi-client-agent as described below, the client will also appear in the client list of opsi-configed, unless you have already added the client there.

Basically there is the possibility to install the agent on the client or to start the installation from the server.

Executing the installation directly on the client is suitable for individual computers. For a mass rollout of the agent, have a look at opsi-deploy-client-agent. If there is already another way to distribute software available, then it is also possible to distribute the opsi-client-agent through it and execute the script silent_setup.cmd included in the package.

Once the agent is installed, available opsi products can be installed on these clients.

#### 6.2.1. Using the opsi-client-agent-installer

1. Logon to the client.
2. Download the installer from your configserver. It is located at https://<fqdn_or_ip_of_the_configserver>:4447/public/opsi-client-agent/ and has the file name:
   - Windows: opsi-client-agent-installer.exe
   - Linux: opsi-linux-client-agent-installer.sh
   - macOS: opsi-mac-client-agent-installer.sh

#### Index of /public/opsi-client-agent/

Use address \\bossfarm\opsi\public to open in Windows Explorer.
1. Execute the installer (for Linux and MacOS this must be done with root-rights, on Windows a UAC-Request may be displayed)

2. The installer will extract itself into a temporary local directory and start the oca-installation-helper.

This shows a user interface with input fields for Client-ID, Opsi Service URL, Username and Password. The fields are pre-filled (if possible e.g. if a old opsiclientend.conf is found), but you may need to add or change some of the data.

- **Client-ID** should be the fqdn of the Client.
- **Opsi Service url** should have the format `https://<fqdn_or_ip_of_the_configserver>:4447`.
- **Username** and **Password** should correspond to a user of the group opsiadmin in case of a first installation. For reinstallation it is also possible to use Client-Id and pckey for authentication.

After starting the Installer by clicking the button **Install** the installer connects to the server to register the client at the server. Afterwards the installer calls the included opsi-script to execute the setup.opsiscript of the opsi-[linux-|mac-]client-agent.

If the installation is finished the installer terminates.

Further information around the opsi-client-agent Installer and the command line parameters and other possibilities to install the opsi-client-agent you will find at the opsi-manual in the chapter **Subsequent installation of the opsi-client-agents**


The subsequent described methods: service_setup / silent_setup are only for backward compatibility to opsi 4.1 and the corresponding opsi-client-agent versions 4.1. Please use as possible the opsi-client-agent Installer.

### 6.2.2. Using service_setup.cmd on Windows NT6 (outdated)

The method described over here is only for backward compatibility to opsi 4.1 and the corresponding opsi-client-agent versions 4.1. Please use as possible the opsi-client-agent Installer.

1. Logon to the Windows client with administrative privileges.
2. Mount the share `\<opiserver>\opsi_depot` on a drive letter.

3. On the drive from the previous step, start the script `opsi-client-agent\service_setup.cmd`
   Do not start the script elevated (via right mouse click: ‘as Administrator’) because an elevated script has no access to the network share.

4. The script copies the needed files to a temporary local directory and starts from there the opsi-script (`opsi-script.exe`) elevated in order to do the installation. This may cause an UAC Message at this point.

5. The script connects to the server via the opsi webservice in order to create the client on the serverside and to retrieve the pckey.
   This is tried first with the user and password provided in config.ini. If the connection fails, a login window will appear, with the Service-URL (opsi-configserver), and user and password. The user required here needs to be a member of the group 'opsiadmin'.
   It is also possible to use a user which only has rights to call the method `host_createOpsiClient`.

6. After installation the client reboots without notice.

### 6.2.3. Using `opsi-deploy-client-agent`

The `opsi-deploy-client-agent` script installs the opsi-client-agent directly from the opsi-server on the clients. This makes it easy to integrate a large number of clients from a server into an opsi environment.

Requirements for the clients are:

- an open C$ share
- an open admin$ share
- an administrative account
- `winexe` must not be blocked by an antivirus program.

The program `winexe` must be available on the server. This is part of the `opsi-windows-support` package.

The `opsi-deploy-client-agent` script can be found at `/var/lib/opsi/depot/opsi-client-agent`

Execute the script with ‘root’ privileges. If the script is not executable, you can solve this issue by executing the following command:

```
```

The script creates the client on the server, then copies the installation files and the configuration information, including the pckey, to the client. After copying the necessary information, `opsi-deploy-client-agent` starts the installation on the client.

There are two ways to copy the installation files. The first method will use the 'mount'-command on the server to mount the C$ share of the client, and copy the files to the share for installation. The second variant will use 'smbclient'-command on the server for mounting C$ share of the client, and copy the files to the share for installation.

With the `opsi-deploy-client-agent` script you can also install to a list of clients. To do this, either any number of clients can be passed as the last parameter or the clients can be read from a file using the ‘-f’ option. When using a file, there must be a client on every line.

The script can work with IP addresses, hostnames or FQDNs. It will try to automatically detect what type of address it is processing.

Possible parameters can be found by using `--help`:

```
opsi_server:/home/user# cd /var/lib/opsi/depot/opsi-client-agent
opsi_server:/var/lib/opsi/depot/opsi-linux-client-agent# ./opsi-deploy-client-agent --help
```

   [--debug-file DEBUGFILE] [-username USERNAME]
   [-password PASSWORD]
   [-use-fqdn | --use-hostname | --use-ip-address]
   [-ignore-failed-ping]
   [-reboot | --shutdown | --start-opsiclientd | --no-start-opsiclientd]
   [-hosts-from-file HOSTFILE]
   [-skip-existing-clients]
### Deploy opsi client agent

Deploy opsi client agent to the specified clients. The c$ and admin$ must be accessible on every client. Simple File Sharing (Folder Options) should be disabled on the Windows machine.

**Positional arguments:**
- `host` The hosts to deploy the opsi-client-agent to.

**Optional arguments:**
- `-h, --help` show this help message and exit
- `--version, -V` show program's version number and exit
- `--verbose, -v` increase verbosity (can be used multiple times)
- `--debug-file DEBUGFILE` Write debug output to given file.
- `--username USERNAME, -u USERNAME` username for authentication (default: Administrator). Example for a domain account: `-u "<DOMAIN>\<username>"
- `--password PASSWORD, -p PASSWORD` password for authentication
- `--use-fqdn, -c` Use FQDN to connect to client.
- `--use-hostname` Use hostname to connect to client.
- `--use-ip-address` Use IP address to connect to client.
- `--ignore-failed-ping, -x` try installation even if ping fails
- `--reboot, -r` reboot computer after installation
- `--shutdown, -s` shutdown computer after installation
- `--start-opsiclientd, -o` Start opsiclientd service after installation (default).
- `--no-start-opsiclientd` Do not start opsiclientd service after installation.
- `--hosts-from-file HOSTFILE, -f HOSTFILE` File containing addresses of hosts (one per line). If there is a space followed by text after the address this will be used as client description for new clients.
- `--skip-existing-clients, -S` skip known opsi clients
- `--threads MAXTHREADS, -t MAXTHREADS` number of concurrent deployment threads
- `--depot DEPOT` Assign new clients to the given depot.
- `--group GROUP` Assign fresh clients to an already existing group.
- `--smbclient` Mount the client’s C$-share via smbclient.
- `--mount` Mount the client’s C$-share via normal mount on the server for copying the files. This imitates the behaviour of the ‘old’ script.
- `--keep-client-on-failure` If the client was created in opsi through this script it will not be removed in case of failure. (DEFAULT)
- `--remove-client-on-failure` If the client was created in opsi through this script it will be removed in case of failure.
7. Rollout existing products

For the rollout of software on clients the ‘opsi-client-agent’ must be installed. This can be deployed on existing computers. If an operating system is installed via opsi, the ‘opsi-client-agent’ will be installed automatically.

Afterwards the management interface opsi-configed is used to distribute software to clients.

7.1. Deploying opsi standard products: opsi-configed

One of the opsi standard products is the product opsi-configed, which installs the opsi Management Interface. This Application is a Java application, therefore a Java Runtime Engine is bundled with the product.

Using opsi-configed, in the mode Configuration of clients, choose the appropriate client in the tab Clients.

If you have not already done so, update the data of opsi-configed by using File / Reload all data or click the reload icon.

Switch to the tab Product configuration, look for the line with the product opsi-configed. Click in the column Requested Action, and select the action setup.

The check mark in the icon menu bar should change its color to red. If you click on it, the new settings will be transmitted to the opsi-server, afterwards its color will be green again.

Reboot the client. The opsi-client-agent should start and install the product opsi-configed. After the installation you can find opsi-configed in the start menu.

7.2. Inventory with the localboot products hwaudit and swaudit.

In opsi-configed, Client configuration mode, under the Clients tab, select the client under consideration.

If not already performed, update the opsi-configed’s dataset using Reload File/Data or clicking the corresponding icon.

Go to the Product configuration tab, click in the Requested column for the hwaudit product, this will open a list/dropdown menu and there select the setup action. Repeat this for the swaudit product.

The check mark in the icon menu bar should change its color to red. If you click it, the new settings will be transmitted to the opsi-server, afterwards its color will be green again.

Then restart the client. It should now start the opsi-client-agent and install the hwaudit and swaudit products. With hwaudit and swaudit, hardware and software information, respectively, is collected and transmitted to the opsi-server. The collected information is displayed under the Hardware Information and Software Inventory tabs, respectively.

7.3. Hardware Inventory with the netboot product hwinvent

If the product 'hwinvent' is installed on your opsi server and you have added a client Section 6.1, “Creation of a new opsi client” which is configured to boot over the network, you can do something else useful: Hardware inventory when there is no operating system installed.

Using 'opsi-configed', in the mode 'Configuration of clients', choose the appropriate client in the tab 'Clients'. If you have not already done so, update the data of opsi-configed by using 'File / Reload all data' or click the reload icon. Switch to the tab 'Netboot products', look for the line with the product hwinvent. Click in the column 'Requested Action', and select the action 'setup'. The check mark in the icon menu bar should change its color to red. If you click on it, the new settings will be transmitted to the opsi-server, afterwards its color will be green again.

Then reboot the client. It should now pull a Linux image over the network (via PXE), to scan the hardware of the PC and then reboot it. If the computer was not otherwise already set up, afterwards the message appears that no operating system is installed on the disk.
The results of the hardware scan have been transmitted to the opsi-server. The results can be viewed under the 'Hardware information' tab.

In case the screen remains black after booting the bootimage or if the network card does not work (correctly), the start parameters of the bootimage may have to be adjusted for this specific hardware. This can be achieved using 'opsi-configed' in the 'Host parameters' tab by editing the entry 'opsi-linux-bootimage.append'. More information can be found in the opsi manual, in the chapter 'netboot products'.

7. Rollout existing products
8. Installation of a new Windows PC with opsi (OS Installation)

The following describes how a computer with no operating system can get a Windows OS installed with opsi.

Suitable clients are real or virtual computers with at least 2048 MB RAM and a network card with network boot support: This means that they support the PXE protocol for booting systems via the network. The network boot has to be activated in the BIOS menu or moved to the first position of the bootorder options.

Virtual hardware is usually well supported by the Windows standard drivers, which can be tried if perform a test installation of Windows. To install Windows on newer real-world machines, you may need to integrate additional drivers first. For an initial test, you can use a VMware Appliance that contains an empty machine and can run in VMware Workstation Player.

For the following chapter you should create a corresponding client in opsi Section 6.1, “Creation of a new opsi client”. This can be done easily through opsi-configed.

Some tools useful for deploying Windows with opsi are installed through the ‘opsi-windows-support’ package.

8.1. OS-Installation: Complete the Base Package for Windows

The opsi win-OS-packages contain only the files that are necessary to perform our automated OS installation, but not the operating system software itself.

For an automatic installation of a Windows operating system, you have to copy your existing original Windows installation files (and if necessary, store the Windows license key on the server).

8.2. NT6 family: Win7 / 2008R2 and up

In order to perform an OS Installation, a so-called WinPE is being used as a ‘Live OS’. You can create it using an opsi package (opsi-winpe), or manually if you so desire. Generally speaking, the Windows version of the PE is independent of the Windows OS version being installed. Above all, the availability of drivers for disk- and network devices is important. Microsoft recommends a 32-Bit PE for 32-bit installations, and a 64-Bit PE for 64-bit installations.

"To install a 64-bit version of Windows you must use a 64-bit version of Windows PE. Likewise, to install a 32-bit version of Windows, you must use a 32-bit version of Windows PE."

In any case you will need an "Assessment and Deployment Kit" (ADK, Windows 8.1 or 10), or its predecessor "Windows Automated Installation Kit" (Windows AIK; until Windows 7), which you install on a supported (preferably 64-bit) Windows OS:

- Windows 10 / 8.1 ADK

Install the Windows PE Add-On for ADK (if possible on a 64-bit machine) in the suggested path under Program Files (x86). Select only the “Windows Pre-Installation Environment (Windows PE)”; Dependencies are automatically selected.

- link: WAIK Windows 7

This site provides you with an ISO file, which may then be burned to a CD or mounted, and then installed.
8.2.1. Creating a PE

The simplest method requires a computer that has opsi-client-agent installed, as well as the Windows ADK (Win8.1, Win10). The manual method is described below in Section 8.2.1.2, “Manual PE creation for Windows 10 & Windows 8 (ADK)

Automated PE creation using opsi

• Using opsi-configed set the localboot-product opsi-winpe to once for the client you intend to use, if desired adjust the product property to x86 instead of x64 at the lower right side, and save (right click > save).

• If the opsi-product opsi-winpe is missing, install it onto your opsi-server with the command opsi-package-updater -v install opsi-winpe.

• Launch an installation event for the client (right click > on-demand, or reboot).

• After a successful completion of this action, move or copy the contents of the now existing folder on your client C:\winpe_<ARCH>\media\ into the pre-existing folder within the OS folder you want to install: \\opiserver\opsi_depot_rw\<operating system>\winpe\.

• Finally run the following command on the console of your opsi server. Finished.

  opsi-set-rights

Manual PE creation for Windows 10 & Windows 8 (ADK)

The console commands are very similar in 32- or 64-bit versions, except for the <ARCH> entries. These have to be replaced with either x86, amd64 or ia64.

Run Start ⇒ "Windows Kits" ⇒ "Windows ADK" ⇒ "Deployment and Imaging Toolkits Environment" from the Start Menu. A command prompt will open which has the required environment variables set.

• Copy the WinPE

  copype.cmd <ARCH> C:\winpe

• Mount the Image

  dism /Mount-Wim /WimFile:C:\winpe\media\sources\boot.wim /index:1 /MountDir:c:\winpe\mount

• replace startnet.cmd

  echo c:\opsi\startnet.cmd > "C:\winpe\mount\Windows\System32\startnet.cmd"

(Note: The file c:\opsi\startnet.cmd will be created by the opsi linux bootimage after the script setup.py is executed. The startnet.cmd contains the call to wpeinit.)

• Unmount the Image

  dism /Unmount-Wim /MountDir:c:\winpe\mount /Commit

• Copy the contents of C:\winpe\ISO to /var/lib/opsi/depot/<productid>/winpe.

  Adjust the access rights by entering:
Manual PE creation for Windows 7 (WAIK)

The console commands are very similar in 32- or 64-bit versions, except for the `<ARCH>` entries. These have to be replaced with either `x86`, `amd64` or `ia64`.

Start a command prompt as Administrator with elevated rights (Start ⇒ Programs ⇒ Accessories ⇒ right click on "Command Prompt" ⇒ "Run as" ⇒ Administrator).

- Copy the WinPE

```
"%ProgramFiles%\Windows AIK\Tools\PETools\copype.cmd" <ARCH> C:\winpe
```

- Mount Image:

```
"%ProgramFiles%\Windows AIK\Tools\<ARCH>\imagex.exe" /mountrw "C:\winpe\winpe.wim" 1 "C:\winpe\mount"
```

- replace startnet.cmd

```
echo c:\opsi\startnet.cmd > "C:\winpe\mount\Windows\System32\startnet.cmd"
```

(Note: The file `c:\opsi\startnet.cmd` will be created by the opsi linux bootimage after the script `setup.py` is executed. The `startnet.cmd` contains the call to `wpeinit`.)

- Unmount the Image

```
"%ProgramFiles%\Windows AIK\Tools\<ARCH>\imagex.exe" /commit /unmount "C:\winpe\mount"
```

- Move the WinPE now (From this target dir more files will be moved to the server).

```
move "C:\winpe\winpe.wim" "C:\winpe\ISO\sources\boot.wim"
```

- Copy the contents of `C:\winpe\media` to `/var/lib/opsi/depot/<productid>/winpe.

Adjust the access rights by entering:

```
opsi-set-rights /var/lib/opsi/depot/<productid>/winpe
```

8.2.2. Extending a PE

In some cases it is useful to extend a PE. Especially when using Dell-Hardware. Dell provides special network and storage drivers specially recommended for use in a PE. These instructions only work with Windows 7. (Windows Vista does not inherit the needed DISM- Deployment Image Servicing and Management.) These instructions assume that you have already completed the previous chapter and have created a PE.

The Windows Automated Installation Kit is not needed for following instructions.
The first step is to download Dell PE drivers from the Dell Website. For Windows 7, you will need the WINPE 3.0 Drivers from Dell. The downloaded CAB-File must be extracted to the local disk. This can be done with 7-zip or the command-line-tool Expand.exe. For simplicity, we recommend creating a directory called “dell-driver” on the local disk, and then extracting the CAB-File into this directory.

- First dism is used to scan the image, in order to determine the required index number. Start a command prompt as administrator (Start ⇒ Programs ⇒ Accessories ⇒ right click on “Command Prompt” ⇒ ”Run as” ⇒ (Administrator) and run the following command:

  `dism /Get-WimInfo /WimFile:C:\winpe\ISO\sources\boot.wim`

In the output of this command, you can see which images are included in the image file. Normally a PE-image is a one-image-file, so you can generally use the index 1, but it is better to check first.

- The next command mounts the image for modification:

  `dism /Mount-Wim /WimFile:C:\winpe\ISO\sources\boot.wim /index:1 /MountDir:c:\winpe\mount`

- To integrate the extracted drivers into the mounted image, you need to execute this command:

  `dism /Image:C:\winpe\mount /Add-Driver /Driver:c:\dell-driver\winpe\x64 /Recurse`

If the architecture is 32-bit, the `x64` must be replaced with `x86`. The Driver-CAB from Dell contains the drivers for both architectures.

If only one driver has to be integrated, then leave out the option `/Recurse`, and point directly to the driver-inf-file instead of the driver-directory. Furthermore, with the option `/ForceUnsigned` it is possible to integrate unsigned drivers to the image.

- Finally the image is unmounted, and the changes are committed:

  `dism /Unmount-Wim /MountDir:c:\winpe\mount /Commit`

- Copy the contents of `C:\winpe\ISO` to `/var/lib/opsi/depot/<productid>/winpe`. Adjust the access rights by entering:

  `opsi-set-rights /var/lib/opsi/depot/<productid>/winpe`

8.2.3. unattend.xml

The control file for the unattended installation is the XML file `unattend.xml`, which you can find under `/var/lib/opsi/depot/win7/custom`. Any modifications to this file should be made in this directory and not in the opsi directory.

The file `unattend.xml` that comes with the opsi package, contains references to the netboot productproperties, which among other things is responsible for activating the Administrator account with the password ‘nt123’.

Documentation for `unattend.xml` can be found in the directory `C:\Program Files\Windows\Waik\docs\chms`, after installing the WAIK.
8.2.4. Driver Integration

The driver integration proceeds as described here: Section 8.6, “Simplified Driver Integration during the unattended Windows Installation”.

8.2.5. Providing the Installation Files

Copy the complete installation DVD to /var/lib/opsi/depot/<productid>/installfiles And adjust the rights and ownership:

```
opsi-set-rights /var/lib/opsi/depot/<productid>/installfiles
```

8.2.6. Installation Log files

- `c:\Windows\Panther\setupact.log`:
  Log until the end of setup phase 4 (running under WinPE)
- `c:\Windows\Panther\setupact.err`:
  Error log until the end of setup phase 4 (running under WinPE)
- `c:\Windows\Panther\UnattendGC\setupact.log`:
  Log from the specialize phase
- `c:\Windows\Panther\UnattendGC\setupact.err`:
  Error log from the specialize phase
- `c:\Windows\System32\Winevt\Logs\*`
- `c:\Windows\ntbtlog.txt` (only when startup logging is activated)

8.3. Windows Product Key

If you have the opsi license management module, you can manage the Windows product keys using the license management module. Read the license management manual or the corresponding chapter in the opsi manual.

If you do not have the license management module, or do not want to use it, proceed as follows.

If you have already set up opsi clients, you can enter a Windows product key per client in the opsi configuration editor:

- select a client
- switch to the netboot products tab
- select the product (e.g. win10-x64)
- change the product property productkey in the lower right corner
- enter the key in the value field
- save by clicking on the “red tick” and leave the field
- save the changes in the backend (“red tick” at the top right).

Or you can assign a default for the Windows product key for the complete opsi depot, which can also be done via the opsi configuration editor:

- Select the depot properties in the configuration editor (tile top right)
- Switch to the Product Default Properties tab
- select the product (e.g. win10-x64)
8. Installation of a new Windows PC with opsi (OS Installation)

- Go to the property line productkey in the switch list on the right
- Enter the key in the value field and add it by clicking on “+”
- save by clicking on the "red tick" and leave the field
- save the changes in the backend ("red tick" at the top right).

8.4. Start the Windows Installation

To start a Windows installation, select the relevant client in opsi-configed, set in the 'Netboot products' tab the action to 'setup' for the desired operating system (e.g. win10-x64). Click on the red checkmark (which turns green again).

The client should now load the opsi-linux-bootimage via the network when booting, where you have to confirm the new OS installation again. Then everything should continue automatically until the logon prompt of the installed Windows is finally on the screen.

If the screen remains black after loading the boot image or the network card does not work correctly, the start parameters of the boot image may have to be adjusted for this specific hardware. You can do this in 'opsi-configed' in the 'Host parameters' tab at the entry 'opsi-linux-bootimage.append'. You can find details on this in the opsi manual in the 'Netboot Products' chapter.

Beware of clients with a hard disk larger than 2 TB. In a non-UEFI system, the maximum partition size is 2 terabytes. If a larger partition is to be created, the installation will fail. This a technical limitation of the standard partition table. You need to split the hard drive into partitions. You can control this via the product properties. Or you can purchase the UEFI module, which eliminates this technical limitation.

8.5. Structure of the Unattended Installation Products

This chapter applies to the Windows netboot products.

8.5.1. Directory Tree Overview
8. Installation of a new Windows PC with opsi (OS Installation)

8.5.2. File Descriptions

- `setup.py`
  This is the installation script which is executed by the boot image.

- `<productid>_<version>.control`
  Contains the metadata of the product as prepared from the package maintainer. These files are here for information purposes only. Changes to this file have no effect on the system.

- `<productid>.files`
  This file is created automatically and should not be changed.

- `create_driver_links.py`
  `show_drivers.py`
  These scripts are for driver integration, which is explained in more detail in the chapter Simplified driver integration in the automatic Windows installation.

8.5.3. Directory installfiles / winpe

- `installfiles`
  This directory contains all files from the installation CD/DVD.

- `winpe`
  Contains a bootable winpe image.

8.5.4. Directories opsi and custom

These two directories contain scripts and configuration files for controlling the operating system installation. During installation, priority is given to files in the custom directories.

The opsi directory contains files that can be overwritten without notice by updates. So no changes to these files should be made. For adjustments, you can make changes in the directory custom, which is preserved during updates.
The subdirectory `postinst.d` contains scripts which are started via the `postinst.cmd` after the actual installation of the operating system, e.g. to install the opsi-client-agent. The scripts are processed in alphabetical order. To clarify the order of execution, the file names begin with a two-digit number (`10_dhcp.cmd`). If you want to make extensions here, you can store scripts in the custom/postinst.d directory with starting numbers between decades (`13_myscript.cmd`). The starting numbers 10, 20, 30,... are reserved for maintenance by opsi.org/uib. The script `99_cleanup.cmd` is the final script and ends with a reboot.

8.5.5. Directory drivers

This directory is used for the integration of drivers and is described in the following chapter.

8.6. Simplified Driver Integration during the unattended Windows Installation

When managing a group of PCs that have devices whose drivers are not included in the standard Windows installation, it usually makes sense to integrate these drivers directly into the installation. In the case of network devices, this can sometimes be unavoidable, because a Windows without a network card is not easily accessible for the administrator.

Opsi supports the automatic integration of drivers into the installation, and therefore simplifies driver deployment. The drivers simply need to be placed into the correct directory. By executing a script, the driver directories are searched and a catalog is created, based on which the bootimage can automatically identify and integrate the correct drivers. Standard drivers, USB drivers, HD audio drivers as well as drivers for hard disk controllers (text mode drivers) can be stored and automatically integrated.

In order for the drivers to be installed with the Windows installation, they must be stored in a specific form on the server. Suitable drivers contain a '*.inf' file that describes the driver for the Windows Setup program. Any drivers in `setup.exe`, '*.zip' or packed any other way are not usable. If you have a computer that already has the drivers installed, then you can extract the drivers in the correct format with the program 'double driver' ([http://www.boozet.org/dd.htm](http://www.boozet.org/dd.htm)).

There are multiple levels of driver integration:

- General driver packages
- Drivers that are suitable for your hardware but are not specially assigned
- Drivers that are manually assigned to computers
- Drivers that are automatically assigned to the computers via the `<vendor>/<model>` fields of the inventory.

How these different levels can be used is described below:

8.6.1. General Driver Packages

When the hardware configuration across the computers is very heterogeneous, then it can make sense to work with general driver packages. General drivers can be placed under `/drivers/drivers`.

Drivers which are found in `/drivers/drivers/`, will be matched to the corresponding hardware using the PCI IDs (or USB- or HD_Audio-IDs) in the description file, and then integrated into the Windows setup if needed.

8.6.2. Drivers that suitable for your hardware but not specially assigned

In case you have to support few different hardware configurations, you can use the drivers provided by the manufacturers. Additional or tested drivers belong in their own directories (name and depth of the directory structure do not matter) below the directory `/drivers/drivers/preferred`.

Drivers located in the directory `/drivers/drivers/preferred` are prioritised over the drivers in `/drivers/drivers/` by using the PCI IDs (or USB- or HD_Audio-IDs) in the description file, and then integrated into the Windows setup if needed.

Problems can occur when the same PCI ID can be found in the description file of different drivers in `preferred`. In this case a direct
assignment of the drivers to the client is necessary.

### 8.6.3. Drivers manually assigned to clients

Additional drivers that are to be installed regardless of their assignment or detection via the PCI- or USB-IDs must be in their own directories (name and depth of the directory structure are irrelevant) below the directory `./drivers/drivers/additional`. Via the product property `additional_drivers` you can assign one or more paths of driver directories within `./drivers/drivers/additional` to a client. Directories specified in the `additional_drivers` product property are searched recursively and all included drivers will be integrated. Symbolic links are also followed. You can use this to create a directory for certain computer types (e.g. dell-optiplex-815).

If a driver for a matching PCI device (or HD audio, USB) is found in the driver directories specified via `additional_drivers`, then no other driver from `drivers/preferred` or `drivers/` is integrated for this device (‘additional_drivers’ can be thought of as super-preferred). This means that ‘additional_drivers’ has the function of adding drivers that would not be found via normal driver detection.

### 8.6.4. Drivers automatically assigned to the clients using the inventory fields

The mechanism of direct assignment of drivers to devices described in the previous section can be automated since opsi 4.0.2. The directory `./drivers/drivers/additional/byAudit` is searched for a directory name that corresponds to the ‘vendor’ found during hardware inventory. A search is now made in this ‘vendor’ directory for a directory name that corresponds to the ‘model’ found during hardware inventory. If such a directory is found, this directory is treated as if it were manually assigned via the product property `additional_drivers`. The directory name ‘byAudit’ is case sensitive. The directory names for ‘Vendor’ and ‘Model’ are not case sensitive (‘Dell’ and ‘dELL’ are treated the same way).

Since opsi 4.0.5, the drivers for a opsi-client can be made available via opsi-configed in the Hardware Inventory tab (see: opsi manual "Automatic driver upload").

The opsi-linux-bootimage looks for drivers in the order:

- `<vendor>/<model> (<sku>)`
- If in the previous no match is found `<system vendor>/<system model>` is checked.
- If in the previous no match is found `<motherboard vendor>/<motherboard model>` is checked.

Some manufacturers use model names, which are very unfavourable for this method, because you can not use some special characters such as `/` in file- or directory names. An example of this would be a model name like: “5000/6000/7000”. A directory with this name is not permitted due to the special characters. Since opsi 4.0.3 the following special characters: `< > ? " : | / *` have therefore been replaced internally by a `_`. With this change you can create the directory for the example as: “5000_6000_7000” and the directory is automatically assigned, although the information in the hardware inventory does not correspond to the directory structure.

### 8.6.5. Structure of the Driver Directory and Driver Files

---

---
8.6.6. Processing of the Different Levels of Driver Integration

The top priority is to include all drivers that are found using the property 'additional_drivers' or using the inventory data in ./drivers/drivers/additional/byAudit. As part of the integration of drivers, it is checked for which hardware of a device (based on the PCI-, USB-, HD-Audio IDs) a driver has been made available in this way. Only for devices that are not matched by a driver, the following methods are used in order to find a matching driver.

For devices for which a driver has not been assigned via 'additional_drivers' (or 'byAudit'), a suitable driver is searched for and integrated using the PCI ID (or USB-, HD-Audio ID).

'Integration' of drivers means the following:

- The driver will be copied to the local hard drive at c:\drv\<num>.
- The Windows Setup is told in the unattended file to search for matching drivers in c:\drv\.

8.6.7. Add and check drivers

After adding a driver or any other change in the ./drivers/drivers directory (or below), execute the following command in the root directory of the netboot product directory to set the rights correctly:

```bash
opsi-set-rights ./drivers
```

After storing drivers in the directories ./drivers/drivers or ./drivers/drivers/preferred, then run the script ./create_driver_links.py. The script searches the directories under './drivers/drivers' and generates a list of links that can be used to identify the assignment of the drivers to specific hardware (PCI-IDs, USB-IDs and HD-Audio-IDs). The script will prioritize the drivers in the preferred directories.

The script setup.py of the bootimage examines the hardware of the computer to be installed and identifies the necessary drivers. These are then copied to the hard disk and the unattend.xml will be patched accordingly.

If a hardware inventory is available for a client, you can use the command:

```bash
./show_drivers.py <clientname>
```
This will show which drivers the boot image would choose for installation via PCI-IDs, USB-IDs, HD-Audio-IDs and 'additional_drivers' (or 'byAudit') and for which hardware no driver is available yet.

Use the output of `show_drivers.py` to check if the desired drivers will be integrated.

It is possible that driver directories from manufacturers contain drivers for different operating system versions (e.g. Windows 7/8.1/10) or different configurations (SATA / SATA-Raid). This cannot be differentiated automatically. If you suspect that the wrong driver will be used, move this driver to the `drivers/exclude` directory and then run `create_driver_links.py` again. Drivers in the directory 'drivers/exclude' are not used during driver integration.

Example output of `show_drivers.py` for a client:

```bash
./show_drivers.py pcdummy

PCI-Devices

[(Standardsystemgeräte), Standard PCI to PCI bridge]
  No driver - device directory /var/lib/opsi/depot/<productid>/drivers/pciids/1022/9602 not found
[ATI Technologies Inc., Rage Fury Pro (Microsoft Corporation)]
  Using build-in windows driver
[(Standard-IDE-ATA/ATAPI-Controller), Standard-Dual-Channel-PCI-IDE-Controller]
  /var/lib/opsi/depot/<productid>/drivers/drivers/D/M/N/123
[Realtek Semiconductor Corp., Realtek RTL8168C(P)/8111C(P) PCI-E Gigabit Ethernet NIC]
  /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/realtek_gigabit_net_8111_8168b
[IEEE 1394 OHCI-conform Hostcontroller-Manufacturer, OHCI-conform IEEE 1394-Hostcontroller]
  No driver - device directory /var/lib/opsi/depot/<productid>/drivers/pciids/197B/2380' not found
[Advanced Micro Devices, Inc., AMD AHCI Compatible RAID Controller]
  /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/ati_raid_sb7xx
[(Standard-USB-Hostcontroller), Standard OpenHCD USB-Hostcontroller]
  No driver - device directory /var/lib/opsi/depot/<productid>/drivers/pciids/1002/4397' not found
[ATI Technologies Inc, ATI SMBus]
  /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/ati_smbus

USB-Devices

[(Standard-USB-Hostcontroller), USB-Connection device]
  /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/brother_844x_pGerb
[Microsoft, USB-Printersupport]
  /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/brother_844x_pGerb

Additional drivers

[ati_hdaudio_azalia]
  /var/lib/opsi/depot/<productid>/drivers/drivers/additional/ati_hdaudio_azalia
```

Example for a client with 'additional_drivers':

```bash
[ati_hdaudio_azalia]
  /var/lib/opsi/depot/<productid>/drivers/drivers/additional/ati_hdaudio_azalia
```
8. Installation of a new Windows PC with opsi (OS Installation)

./show_drivers.py e5800
Manually selected drivers (additional)

[hp_e5800]
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPA13.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPA11.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPA12.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPA11.inf

[hp_e5800]
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/autorun.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/ibxHDMI/IntcDAud.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/HDMI/IntcHdmi.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/Graphics/kit24890.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/IIPS/Impcd.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp54284/Realtek 64bit/hp64win7.inf

PC-Devices

[8086:27C8] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27C8
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27DA] Intel : Intel(R) N10/ICH7 Family SMBus Controller - 27DA
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27C9] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27C9
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27DF] Intel : Intel(R) ICH7 Family Ultra ATA Storage Controllers - 27DF
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27CA] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27CA
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27CB] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27CB
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27D0] Intel : Intel(R) N10/ICH7 Family PCI Express Root Port - 27D0
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27B8] Intel : Intel(R) ICH7 Family LPC Interface Controller - 27B8
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27D2] Intel : Intel(R) N10/ICH7 Family PCI Express Root Port - 27D2
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27C0] Intel : Intel(R) N10/ICH7 Family Serial ATA Storage Controller - 27C0
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27D8] Microsoft : High Definition Audio-Controller

No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/8086/27D8' not found

Manually selected [hp_e5800] /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp54284/Realtek 64bit/rtl8102e.inf

USB-Devices

[0461:0018] (Standardsystemgeräte) : USB-Eingabegeräte
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461/0018' not found
[0461:0028] (Standardsystemgeräte) : USB-Eingabegeräte
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461/0028' not found
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/058F/6366' not found
[0461:0018] (Standard-USB-Hostcontroller) : USB-Verbundgeräte
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461/0018' not found

HD-Audio-Devices

[10EC:8662] Realtek High Definition Audio
Manually selected [hp_e5800] /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64
8. Installation of a new Windows PC with opsi (OS Installation)

Example for a client with 'byAudit':

```plaintext
./show_drivers.py pctry5detlef
Manually selected drivers (additional)
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series Secondary (Microsoft Corporation - WDDM)/atilhag.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series (Microsoft Corporation - WDDM)/atilhag.inf
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/MEDIA/Realtek AC'97 Audio/oem21.inf

PCI-Devices
[1002:5B70] ATI Technologies Inc. : Radeon X300/X550/X1050 Series Secondary (Microsoft Corporation - WDDM)
Manually selected
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series Secondary (Microsoft Corporation - WDDM)
Multiple selected
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series (Microsoft Corporation - WDDM)
[100E:0053] (Standard-IDE-ATA/ATAPI-Controller) : Standard-Zweikanal-PCI-IDE-Controller
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/0053' not found
[100E:0050] (Standardsystemgeräte) : PCI Standard-PCI zu PCI-Brücke
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/0050' not found
[1022:1100] AMD : K8 [Athlon64/Opteron] HyperTransport Technology Configuration
Using build-in windows driver
[100E:0054] (Standard-ATA/ATAPI-Controller) : Standard-Zweikanal-PCI-IDE-Controller
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/fsc__esprimo_p625/FTS_NVIDIASATAAHCI_DRIVER_Vista64V103042MCP78__1026963/NVIDIA_SATA_AHCI_DRIVER_Vista64_V10.3.0.42_MCP78 (textmode capable)
[1022:1101] AMD : K8 [Athlon64/Opteron] Address Map
Using build-in windows driver
[100E:0055] (Standard-IDE-ATA/ATAPI-Controller) : Standard-Zweikanal-PCI-IDE-Controller
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/fsc__esprimo_p625/FTS_NVIDIASATAAHCI_DRIVER_Vista64V103042MCP78__1026963/NVIDIA_SATA_AHCI_DRIVER_Vista64_V10.3.0.42_MCP78 (textmode capable)
[1022:1102] AMD : K8 [Athlon64/Opteron] DRAM Controller
Using build-in windows driver
[100E:0057] NVIDIA : CK804 Ethernet Controller
Using build-in windows driver
[1022:1103] AMD : K8 [Athlon64/Opteron] Miscellaneous Control
Using build-in windows driver
[100E:0059] Realtek : Realtek AC'97 Audio
Manually selected
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/MEDIA/Realtek AC'97 Audio
[100E:005E] NVIDIA : CK804 Memory Controller
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/ga-ma78-pcbon4/chipset_win7-64/SMBUS
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/104C/8025' not found
[100E:005A] (Standard-USB-Hostcontroller) : Standard OpenHCD USB-Hostcontroller
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/005A' not found
[100E:0058] (Standardsystemgeräte) : PCI Standard-ISA-Brücke
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/0058' not found
[100E:005B] (Standard-USB-Hostcontroller) : Standard PCI zu USB erweiterter Hostcontroller
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/005B' not found
[100E:0060] ATI Technologies Inc. : Radeon X300/X550/X1050 Series (Microsoft Corporation - WDDM)
Manually selected
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series Secondary (Microsoft Corporation - WDDM)
Multiple selected
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi
/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon X300-X550-X1050 Series (Microsoft Corporation - WDDM)
[100E:0052] NVIDIA : CK804 SMBus
Using build-in windows driver
[100E:005C] (Standardsystemgeräte) : Standard PCI to PCI Bridge
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/100E/005C' not found
```
USB-Devices

[1241:1111] (Standardsystemgeräte) : USB-Eingabegerät

No driver - vendor directory /var/lib/opsi/depot/<productid>/drivers/usbsids/1241' not found

HD-Audio-Devices

No devices installed

TIPS

• Directory names such as NDIS1 contain Vista drivers; NDIS2 contain Win7 drivers

• NDIS versions:

NDIS 6.0: Windows Vista
NDIS 6.20: Windows 7, Server 2008 R2
NDIS 6.30: Windows 8, Windows Server 2012
NDIS 6.50: Windows 10, version 1507
NDIS 6.70: Windows 10, version 1703
NDIS 6.80: Windows 10, version 1709
NDIS 6.91: Windows 10, version 1803
NDIS 6.82: Windows 10, version 1809 and Windows Server 2019
NDIS 6.83: Windows 10, version 1903

• Some chipset drivers contain description files, which specify hardware without actually providing drivers. An example would be the cougar.inf or ibexahci.inf from Intel. If such a 'pseudo driver' directory is assigned via 'additional_drivers' (or 'byAudit'), this means that the hardware listed here is excluded from further searches for drivers in the 'preferred' directory.

• SATA drivers and SATA-RAID drivers refer to the same PCI ID. However, a SATA RAID driver will not function with a single-disk system.

• Check the output of ./show_drivers.py carefully!
9. Integration of New Software Packages into the opsi Server

The primary objective of software distribution is to accomplish automatic software installation without user interaction. Software installation and user activity should be strictly separated. In most cases, the installation process requires administrative privileges which the user usually doesn't have. So the installation process has to be done independently from the user. This way, the user can neither interfere nor be affected by the software installation process.

In order to do this, you have to write a script for the script driven installer, which is called an 'opsi-script' script. This script in addition to the installfiles and some metadata can be packed as a opsi-product, which in turn can be installed on a opsi-server.

9.1. A Brief Tutorial: How to write a opsi-script Script

9.1.1. Introduction

This tutorial merely helps you getting started with opsi. It can't replace professional training (which you may order through uib), or thoroughly studying the complete opsi manuals (which might be time consuming and partially error prone if you lack background knowledge). uib now offers training in English, too.

Training and Support:
Get Training by uib gmbh in Europe or possibly Northern America:
https://uib.de/en/support-training/support/

Manuals:
The opsi Manuals can be found at: https://uib.de/en/opsi-documentation/documentation/ important for scripting:
opsi-script reference card and opsi-script manual

Wiki (Scripts, Tips, Links):
https://forum.opsi.org/wiki

Support Forum (fast and free vendor support):
https://forum.opsi.org

9.1.2. Methods of Non-Interactive Installation for Windows

Regardless of whether or not you are using opsi or another management solution, there are three different ways to install software without user interaction:

1. Unattended or Silent Installation
   The original setup binary from the software manufacturer can be executed with command line arguments which enable a 'silent' or 'unattended' mode. It depends on whether or not the program supports a silent installation mode. A special case of this method is the unattended installation of MSI packages. "silent" Installation of a MSI-Package: A MSI-Package can be installed using the "quiet" Option.

2. Interactive Setup with recorded Answers
   The actions executed by the administrator while running the manufacturer's setup program during a manual installation are automatically recorded using the free tools 'Autoit' or 'Autohotkey'. This generates an autoIt script which in turn can be used for an unattended installation.

3. Recreate the setup-routine with opsi-script:
   The actions executed by the setup-routine when installing manually are recorded and the opsi-script is used to reproduce them.
opsi supports all of these variants. Usually a combination of these methods in one script provides the best result. For example, performing the basic installation using the original setup if available, and then doing some customizing by patching the registry or the file based configuration.

9.1.3. Structure of an opsi-script script

In the subsequent chapters the basic elements of a opsi-script script will be described with examples for Windows.

First an example for a simple opsi-script script:

```plaintext
[Actions]
WinBatch_tightvnc_silent_install

[WinBatch_tightvnc_silent_install]
"%ScriptPath\tightvnc-1.3.9-setup.exe" /silent
```

An opsi-script script consists of primary and secondary sections. Sections are introduced with a section name in square brackets, as known from the ini files. The true software installation work takes place in the secondary sections, which are called by the primary sections.

The secondary sections are "theme specific" and each has a special syntax. The section name of a secondary section starts with its type, followed by a freely determinable name.

In the shown example, the primary section [Actions] calls a secondary section [WinBatch_tightvnc_silent_install]. The secondary section is of the type WinBatch. The content of a WinBatch section is executed via the Windows API. So in this case the setup program tightvnc-1.3.9-setup.exe is started with the parameter /silent.

9.1.4. Primary Sections

**Actions**

The [Actions] section is the actual main program. This is where the script processing begins.

**Sub-sections**

Program sections that are required constantly can be redistributed to sub sections (subroutines). It's possible to source sub sections to external files.

The primary sections are the main program in which the script flow is controlled. For this there are:

- Variables: Strings and string lists
- if elseif else endif statements
- for loops over string lists
- Functions
9.1.5. Important secondary sections

**Files**

File operations, such as:

- copy (with version control, recursive ...).
- delete
- create directories
- ...

**WinBatch**

Used to call programs via the Windows API. For example, invocations to setup programs in silent mode are made in these sections.

**ShellInAnIcon**

The content of this section is passed to the operating system on the typical shell for execution. This shell is the `cmd.exe` for Windows, for Linux and for macOS the `bash`. As such, normal batch scripts can be stored here. Name variants of `ShellInAnIcon` with identical behavior are `Shellbatch`, `DOSBatch` and `DOSInAnIcon`.

**ExecWith**

The contents of these sections are passed to an external program such as an (interpreter) for execution. For example, `ExecWith` can be used to integrate AutoIt scripts [http://www.autoitscript.com](http://www.autoitscript.com) directly into the opsi-script script.

**Registry**

The `Registry sections` are used to edit the registry.

**LinkFolder**

`LinkFolder sections` are used to create and remove shortcuts. For example, such shortcuts can be created on the desktop or in the start menu.

9.1.6. Global constants

Global constants are text placeholders that can be used in primary and secondary sections and are textually replaced by their values at runtime.

The use of placeholders can be utilized to ensure that paths are set correctly in different environments (in the case of systems with different languages or operating system versions for example).
Examples:

```plaintext
%ProgramFiles32Dir%
c:\Program Files (x86)

%Systemroot%
c:\windows

%System%
c:\windows\system32

%opsiTmpDir%
c:\

%Scriptpath%
<path to running script>
```

9.1.7. Second example: tightvnc

For clarification purposes, now a simple script for the installation of `tightvnc`. As a matter of fact this script would get on with the call of the silent installation in the `Winbatch section`. However, during a repeated installation an interactive dialog appears here (because of the restart of the running service). This dialog window is closed (if it appears) with the help of `AutoIt`.

```
[Actions]
Message "Installing tightvnc 1.3.9 ..."
ExecWith_autoit_confirm "%ScriptPath%\autoit3.exe" WINST /letThemGo
WinBatch_tightvnc_silent_install
KillTask "autoit3.exe"

[WinBatch_tightvnc_silent_install]
"%ScriptPath%\tightvnc-1.3.9-setup.exe" /silent

[ExecWith_autoit_confirm]
; Wait for the confirm dialog which only appears if tightvnc was installed before as service
; Waiting for the window to appear
WinWait("Confirm")
; Activate (move focus to) window
WinActivate("Confirm")
; Choose answer no
Send("N")
```

9.1.8. Elementary commands for primary sections

**String-Variable**

**Variable declaration**

```
'DefVar <variable name> [= <initial value>]'
```

**Variable assignment**

```
'Set <variable name> = <value>'
```

**Example:**
DefVar $ProductId$
Set $ProductId$ = "firefox"
or
DefVar $ProductId$ = "firefox"

String variables are handled differently in primary and secondary sections. In primary sections, string variables are independent objects. Only here they can be declared and assigned values. Accordingly, the connection of variables and strings to a string expression is to be performed with a "+" operator.

Example: "Installing "+$ProductId$" ..."
In secondary sections, string variables are replaced with the contents of the variable before the section is executed.
For example: "Installing $ProductId$ ..."
This should be taken into consideration when the corresponding string expressions are cut and pasted in the script.
The advantage of this construction is that in sections that are executed outside the 'opsi-script' (DosBatch / Execwith) opsi-script variables can be manipulated at ease.

Message / ShowBitmap

Text output during installation:
Message <string>

Example:
Message "Installing "$ProductId" ...

To output a graphic during the installation:
ShowBitmap <filename> <subtitle>

Example:
ShowBitmap "%ScriptPath%/python.png" "Python"

if [elseif] [else] endif

Syntax:

if <condition>
  ;statement(s)
[elseif <condition>
  ;statement(s)]
[
  else
  ;statement(s)
] endif
Functions

**HasMinimumSpace**
Checks for free space on the hard disk.

**FileExists**
Checks for the existence of a file or directory.

Errors, logging and comments

**Comment characters ‘;’**
Lines that start with a semicolon (’;’) are not interpreted.

**Comment**
Writes a comment message to the log file.

**LogError**
Writes an error message to the log file.

**IsFatalError**
Cancels the execution of the running script and reports the installation as failed.

Condition for execution

**requiredWinstVersion**
 specifies the (minimum) required opsi-script version.

Other important opsi-script functions

An overview of the opsi-script functions is given by the reference card

A detailed documentation can be found in the opsi-script manual:

Here are a few more notes on particularly important elements:

**Stringlists:**
String lists are very powerful, especially for evaluating output from external programs. Read the opsi-script docs for more information.

**ExitWindows:**
Reboot/Shutdown the system and finish the opsi-script.

- **ExitWindows /Reboot**
  Computer restart after completion of the running script.
- **ExitWindows /ImmediateReboot**
  Immediate reboot.
- **ExitWindows /ImmediateLogout**
  Immediately stop script editing and terminate opsi-script.

**Product-Properties:**
For some products it’s necessary to provide options. These are specifically evaluated per client at runtime. How such properties are created is described in the chapter Creation of opsi product packages
The access to the values of the properties is done via the function `GetProductProperty`:

```plaintext
if GetProductProperty("example-property", "no") = "yes"
    Files_copy_extra_files
endif
```

**Encoding:**

Write your scripts in UTF-8 encoding and set the line `encoding=utf8` at the beginning of the file.

**Special commands for Windows**

- `getOS : string // returns: Linux or Windows_NT or MacOS [W/L/M]
- `getMsVersionInfo : string // Windows Version Information [W]
- `GetMsVersionName //Windows marketing Version //since 4.12.4.35 [W]
- `getMSVersionMap : stringlist [W]

### 9.1.9. Example: Windows-Template 'opsi-template'

This template can be created with `opsi-setup-detector`.

**Listing 1. setup.opsiscript: Installationsscript**

```plaintext
; ----------------------------------------------------------------
; This is a opsi-script file.
; See https://opsi.org  https://uib.de
; This code was originally created by opsi-setup-detector 4.2.0.10
; ----------------------------------------------------------------
encoding=utf8

[Actions]
requiredOpsiVersion >= "4.12.4.23"

importlib "uib_exitcode.opsiscript"
importlib "osd-lib.opsiscript"

DefVar $ProductId$
DefVar $InstallDir$
DefVar $MinimumSpace$
DefVar $ExitCode$
DefVar $ErrorString$
DefVar $LicenseRequired$
DefVar $LicenseKey$
DefVar $LicensePool$
DefVar $OS$
DefVar $MsiId$
DefVar $UninstallProgram$
```
; Please edit the following values:
; $ProductId$ is the name of the product in opsi, only lower letters, no umlauts, no white spaces, use '-' as a seperator
Set $ProductId$ = "opsi-template"
Set $MinimumSpace$ = "1 MB"
; the path were we find the product after the installation
; Set $InstallDir$ = "%ProgramFiles32Dir%\<path to the product>"
Set $InstallDir$ = "unknown"
Set $LicenseRequired$ = "false"
Set $LicensePool$ = ""
;
set $OS$ = GetOS
if not(($OS$ = "Windows_NT"))
    logError "Installation aborted: wrong OS version: only Windows"
    isFatalError "wrong OS"
endif
if not(HasMinimumSpace("%SystemDrive%", $MinimumSpace$))
    LogError "Not enough space on %SystemDrive%, " + $MinimumSpace$ + " on drive %SystemDrive% needed for " + $ProductId$
    isFatalError "No Space"
    ; Stop process and set installation status to failed
endif

comment "Show product picture"
ShowBitmap "%ScriptPath%" + $ProductId$ + ".png" $ProductId$

if FileExists("%ScriptPath%\delsub.opsiscript")
    comment "Start uninstall sub section"
    Sub "%ScriptPath%\delsub.opsiscript"
endif

Message "Installing " + $ProductId$ + " ...

if $LicenseRequired$ = "true"
    comment "Licensing required, reserve license and get license key"
    set $LicenseKey$ = get_licensekey_byPool($LicensePool$)
endif

comment "Start setup program"
ChangeDirectory "%SCRIPTPATH%"
;----------------------------------------------
Winbatch_install
;----------------------------------------------
set $ExitCode$ = getlastexitcode
if "true" = isMsiExitcodeFatal_short($exitcode$, "true", $ErrorString$ )
    LogError $ErrorString$
    isFatalerror $ErrorString$
else
    Comment $ErrorString$
endif

comment "Copy files"
Files_install /32Bit

comment "Patch Registry"
Registry_install /32Bit
comment "Create shortcuts"

LinkFolder_install

[Winbatch_install]
; This is a comment
; You can use $LicenseKey$ var to pass a license key to the installer
;
; === Nullsoft Scriptable Install System
"%ScriptPath%\Setup.exe" /S
;
; === MSI package
You may use the parameter PIDKEY=$Licensekey$
msiexec /i "%ScriptPath%\some.msi" /l* "%opsiLogDir%\$ProductId$.install_log.txt" /qb-! ALLUSERS=1 REBOOT=ReallySuppress
;
; === InstallShield + MSI
; You can use $LicenseKey$ var to pass a license key to the installer
"%ScriptPath%\Setup.exe" /s /v" /l* "%opsiLogDir%\$ProductId$.install_log.txt" /qb-! ALLUSERS=1 REBOOT=ReallySuppress"
;
; === Inno Setup
http://unattended.sourceforge.net/InnoSetup_Switches_ExitCodes.html
You may create setup answer file by: setup.exe /SAVEINF="filename"
You may use an answer file by the parameter /LOADINF="filename"
"%ScriptPath%\setup.exe" /sp- /silent /norestart /nocancel /SUPPRESSMSGBOXES

[Files_install]
; Example of recursively copying some files into the installation directory:
;
; copy -s "%ScriptPath%\files\*.*" "$InstallDir$"

[Registry_install]
; Example of setting some values of an registry key:
;
; openkey [HKEY_LOCAL_MACHINE\Software\$ProductId$]
; set "name1" = "some string value"
; set "name2" = REG_DWORD:0001
; set "name3" = REG_BINARY:00af99cd

[LinkFolder_install]
; Example of deleting a folder from AllUsers startmenu:
;
; set_basefolder common_programs
; delete_subfolder $ProductId$
;
; Example of creating a shortcut to the installed exe in AllUsers startmenu:
;
; set_basefolder common_programs
; set_subfolder $ProductId$
;
; set_link
; name: $ProductId$
; target: <path to the program>
; parameters:
; working_dir: $InstallDir$
; icon_file:
; icon_index:
; end_link
;
; Example of creating a shortcut to the installed exe on AllUsers desktop:
;
; set_basefolder common_desktopdirectory
Listing 2. delsub.opsiscript: Deinstallations-SubSkript

; This is a opsi-script file.
; See https://opsi.org https://uib.de
; This code was originally created by opsi-setup-detector 4.2.0.10
; encoding=utf8

Message "Check for existing installation of " + $ProductId$ + " ...

Set $MsiId$ = '{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}'+
Set $UninstallProgram$ = $InstallDir$ + "\uninstall.exe"

if FileExists($UninstallProgram$)
  comment "Uninstall program found, starting uninstall"
  Winbatch_uninstall
  Sub_check_exitcode_del
endif

if not (getRegistryValue("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" + $MsiId$ ,
"DisplayName","32bit") = "")
  comment "MSI id " + $MsiId$ + " found in registry, starting msiexec to uninstall"
  Winbatch_uninstall_msi
  Sub_check_exitcode_del
endif

comment "Delete files"
if not(($InstallDir$ = '') or ($InstallDir$ = 'unknown'))
  Files_uninstall
endif

comment "Cleanup registry"
Registry_uninstall

comment "Delete program shortcuts"
LinkFolder_uninstall

[Winbatch_uninstall]
; Choose one of the following examples as basis for program uninstall
;
; === Nullsoft Scriptable Install System  9. Integration of New Software Packages into the opsi Server
70 / 102
; "$UninstallProgram$" /S
;
; === Inno Setup ==
; "$UninstallProgram$" /silent /norestart /SUPPRESSMSGBOXES /nocancel

[Winbatch_uninstall_msi]
msiexec /x $MsiId$ /qb-! REBOOT=ReallySuppress

[Files_uninstall]
; Example for recursively deleting the installation directory:
;
; del -sf "$InstallDir\"

[Registry_uninstall]
; Example of deleting a registry key:
;
; deletekey [HKEY_LOCAL_MACHINE\Software\$ProductId$]

[LinkFolder_uninstall]
; Example of deleting a folder from AllUsers startmenu:
;
; set_basefolder common_programs
; delete_subfolder $ProductId$
;
; Example of deleting a shortcut from AllUsers desktop:
;
; set_basefolder common_desktopdirectory
; set_subfolder ""
; delete_element $ProductId$

[Sub_check_exitcode_del]
set $ExitCode$ = getlastexitcode
if "true" = isMsiExitcodeFatal_short($exitcode$, "true", $ErrorString$ )
    LogError $ErrorString$
    isfatalerror $ErrorString$
else
    Comment $ErrorString$
endif

;-----------------------------------------------------
Listing 3. `uninstall.opsiscript`: Deinstallations-Skript

```plaintext
; This is a opsi-script file.
; See https://opsi.org https://uib.de
; This code was originally created by opsi-setup-detector 4.2.0.10
; ----------------------------------------------------------------
encoding=utf8

[Actions]
requiredOpiscriptVersion >= "4.12.4.23"
importlib "uib_exitcode.opsiscript"
importlib "osd-lib.opsiscript"

DefVar $ProductId$
DefVar $InstallDir$
DefVar $MinimumSpace$
DefVar $ExitCode$
DefVar $ErrorString$
DefVar $LicenseRequired$
DefVar $LicenseKey$
DefVar $LicensePool$
DefVar $OS$
DefVar $MsiId$
DefVar $UninstallProgram$

; Please edit the following values:
; ----------------------------------------------------------------
; $ProductId$ is the name of the product in opsi, only lower letters, no umlauts, no white spaces, use '-' as a seperator
Set $ProductId$ = "opsi-template"
; the path were we find the product after the installation
;Set $InstallDir$ = "%ProgramFiles32Dir%\<path to the product>"
Set $InstallDir$ = "unknown"
Set $LicenseRequired$ = "false"
Set $LicensePool$ = ""
; ----------------------------------------------------------------

set $OS$ = GetOS
if not(("$OS$ = "Windows_NT"))
    logError "Installation aborted: wrong OS version: only Windows"
    isFatalError "wrong OS"
endif

comment "Show product picture"
ShowBitmap "%ScriptPath%\" + $ProductId$ + ".png" $ProductId$

Message "Uninstalling " + $ProductId$ + " ...
if FileExists("%ScriptPath%\delsub.opsiscript")
    comment "Start uninstall sub section"
    Sub "%ScriptPath%\delsub.opsiscript"
endif
if $LicenseRequired$ = "true"
    comment "Licensing required, free license used"
    Set $tmpstr$ = FreeLicense($LicensePool$)
endif
```

9. Integration of New Software Packages into the opsi Server

72 / 102
9.2. Creating an opsi product

9.2.1. Installation of opsi-setup-detector, opsi PackageBuilder and opsi-logviewer

Installation of the opsi-package-builder

The opsi-package-builder is currently available for Windows and Linux and MacOS.

The installation files / packages of the opsi-package-builder can be found here:

There you will find in the first lines of the post links to the installation files for Windows and Linux and MacOS. The opsi-package-builder is not made by 'uib' but by Holger Pandel from the opsi-community (thank you!).

The opsi-package-builder open source license:
https://github.com/pandel/opsiPackageBuilder/blob/master/LICENSE_DE

The opsi-package-builder has its own documentation, that is part of the installation.

You can install the opsi-package-builder also via opsi:

The package opsipackagebuilder_wlm belongs to the opsi standard products and should be installed on your opsi-server. If not, use:

```bash
opsi-package-updater install opsipackagebuilder_wlm
```

to install it on the opsi-server.

Installation of the opsi-setup-detector

The opsi-setup-detector is currently available for Windows and Linux and MacOS.

You can install the opsi-setup-detector via opsi:

The package opsi-setup-detector belongs to the opsi standard products and should be installed on your opsi-server. If not, use:

```bash
opsi-package-updater install opsi-setup-detector
```

to install it on the opsi-server.

A setup program to install the opsi-setup-detector without opsi can be found at:
https://download.uib.de/opsi4.2/misc/helper/

The base functionality of the opsi-setup-detector is the same on all supported platforms. While analyzing a installer file some helper programs will be called, that may not available or runnable.

- Inno-Setups will be analyzed with innounpack.exe at Windows.
- wix-setups will be analyzed with dark.exe at Windows.
- .deb or .rpm files will be analyzed with the Linux command line tools.

The opsi product opsi-setup-detector has a dependency on the opsi product opsi-package-builder_wlm. The opsi-setup-detector uses the opsi-package-builder if available, but can for the most part also work without it. The installation of the opsi-package-builder is recommended.
Installation of the opsi-logviewer

The opsi-logviewer is currently available for Windows, Linux and MacOS.

You can install the opsi-logviewer via opsi:

The package `opsi-logviewer` is part of the opsi standard products and should be installed on your opsi-server. If not, with:

```
opsi-package-updater install opsi-logviewer
```

You can install it on the opsi-server.

A setup program to install the opsi-setup-detector on Windows even without opsi can be found at:
https://download.uib.de/opsi4.2/misc/helper/

The opsi product `opsi-logviewer` has a stated dependency to the opsi product `javavm`.

### 9.2.2. The opsi-setup-detector program for creating a Windows script

**Opsi-setup-detector Start and necessary configurations**

The opsi-setup-detector can be started from the programs menu and can be found there under `opsi.org`. Under Windows the opsi-setup-detector is also integrated into the context menu of the explorer in order to call the setup program directly for analysis with the right mouse button.

![opsi-setup-detector](image)

*Figure 18. opsi-setup-detector Necessary configuration on first startup*

After the first start of the opsi-setup-detector a configuration mask appears. The following information is required here:

- **fullname**: (Used for entries in the changelog.txt)
- **email_address**: (Used for entries in the changelog.txt)
- **workbench_path**: Path to the directory in which the opsi packages are going to be created. This is ideally the path to the place
where the `opsi_workbench` of your opsi-server is mounted.

After all needed configurations has be done and saved, you will see the startpage.

![opsi Setup Detector](image)

**Figure 19. opsi-setup-detector Start**

On the main window, select the desired task and follow the dialogs or select the 'Next step' button.

The offered tasks are grouped by:

- OS independent
- Windows
- Linux
- MacOS
- multi platform

The offered tasks for Windows:

1. **Analyze file and create opsi package**
   Here, a setup file is analyzed and the entire process is run until an opsi package is created. This process is described in the next chapter.

2. **Analyze 2 files (32 / 64 bit) and create opsi package**
   The procedure is the same as in point 1 above with the following differences:
   Two setup programs for the 32 and 64 bit architectures are queried and analyzed. The product gets an additional property: `install_architecture` with the possible values: `32bitonly`, `64bitonly`, `both`, `systemspecific`.

3. **Only analyze single file**
   runs, similarly to point 1 above, only that is aborted after the analysis of the setup program.

4. **Create an opsi package template**
   This point does not ask for a installer file, but creates a opsi template product for Windows with the information from the product configuration is already taken over.
opsi-setup-detector: Analyze file and create opsi package

The workflow is here described using *Analyze file and create opsi package* as an example.

![Figure 20. opsi-setup-detector Start](image)

After you selected the task, you will get a file selection dialog. Select now the setup file that has to be analyzed. The analysis will start directly after the selection is done.

opsi-setup-detector: Analyze

![Figure 21. opsi-setup-detector analysis](image)
If the analyze found no result, you will get here a *Sorry unknown Installer*.

![Sorry Unknown Installer dialog]

In this dialog you may choose to abort the create process. You may also choose to continue based on the pattern of a choosable installer type.

If we have a successful analyze, you will see the result page.

![opsi-setup-detector Result of the analysis]

**Figure 22. opsi-setup-detector Result of the analysis**

- **Detected Setup Type**: Type of detected Installer
- **MST allowed**: 
- **Link** with information about the installer
- **Setup file**: Path and name of the analyzed setup file
- **MST file**: For MSI installers or installers which contain MSI, an MST file can be specified here which will be integrated into the MSI call.
- **MsId**: For MSI installers or installers that contain MSI in the form of product code
- **Software version**: The version of the software to be installed if determinable.
- **Setup file size MB**: Size of the setup file in MB
• **Required space MB**: This value is an estimate of six times the size of the setup file and can be adjusted if necessary.

• **InstallDir**: As far as detected the directory where the software will be installed.

• **Unattended installation command**: The determined command for a non-interactive installation.

• **Unattended deinstallation command**: The determined command for a non-interactive deinstallation.

• **Deinstallations program**: The determined deinstallations program

The values determined here can now be corrected or supplemented if necessary. The button **Next Step** leads to the first page of the product configuration. The metadata of the opsi product to be created is entered here.

The values determined here can be incorrect and are probably incomplete! After an initial installation, you should definitely check the values of **InstallDir**, **deinstallation program** and **software version** and adjust them in your script if necessary.

**opsi-setup-detector: Product configuration 1**

![opsi-setup-detector Product configuration 1](image)

**Figure 23. opsi-setup-detector Product configuration 1**

• **opsi Product ID**: this is the name of the opsi package to be generated and is generated from the product name below, where spaces and other invalid characters are replaced by a ‘-’. The proposed opsi Product ID can of course be changed.

• **Product Name**: the name of the software to install. This may have to be corrected manually.

• **Product Version**: The version number determined from the name of the setup file must probably be corrected manually. It may only contain numbers and periods, since it's used for the versioning of the opsi package.

• **Description**: In this field the product name is given as default and should be completed with further hints, which are then set as product description of the opsi package.

• **License required**: If this checkbox is set, ‘$LicenseRequired$=true’ will be set when patching the opsiscrpt.

**opsi-setup-detector: Priority and dependencies**
For normal application software you don’t have to do anything here, due to the fact that the default settings ‘fit’. You can press the Next Step button.

Otherwise, here is an explanation of the settings that are possible:

Priority
affects the installation order. **Recommended for application software: 0**
Possible values are between 100 (at the very beginning) and -100 (at the very end). If product dependencies also exist, these will also additionally influence the installation sequence.

**Dependencies**
Here you can define the dependencies between products. If the configuration contains the connection data for your opsi-server, the connection will here be started. If the configuration does not contain the password (for security reasons) you will be asked for the password:
Figure 25. opsi-setup-detector Dependency Editor

**Productid**

Productid (identifier) of the product to which a dependency exists.

If there is a connection to the opsi-server, this will be noticed in green letters and you may select the `productid` in the DropDownBox. If there is no connection, this will be noticed in red letters and you have to write the `productid` in the input field.

**Require Mode**

You can either request the **Action** setup or (see below) the **State** *(installed)*.

**Action or State**

For **State**: State that the product to which a dependency corresponds, should have *(installed)*. If there is another status, the product is set to **setup**.

For **Action**: Action request, which should be set on the product, whereupon there is a dependency *(setup)*.

This control is disabled while creating a *Meta Product* to avoid sense less changes.

**Requirement Type**

Installation order. If the product for which there is a dependency must be installed before the installation of the current product, then this is **before**. If it must be installed after the current product, this is **after**. If the order doesn't matter then nothing has to be entered here.

This control is disabled while creating a *Meta Product* to avoid sense less changes.

**Hint:**

Unfortunately there is currently no generic mechanism for uninstalling product dependencies. The Product Dependency mechanism is only reliable for action: setup and the (before- or after-) setup actions to be triggered and installed status, in the case of a requiredAction: uninstall this leads unfortunately to errors.

Another hint:

The actual installation order is determined by a combination of product dependencies and product prioritization. Details about this can be found in the opsi manual in the chapter 'Manipulating the installation sequence by priorities and dependencies'.

**opsi-setup-detector: Properties**

Here, editable properties (product variables) can be defined for the product.
### Field / Function

<table>
<thead>
<tr>
<th>Field / Function</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Name</td>
<td>Name of the product variable</td>
<td>This identifier is displayed in the product configuration in opsi-configed and can be read within the scripts with the function <code>GetProductProperty</code>.</td>
</tr>
<tr>
<td>Property Type</td>
<td>Variable type</td>
<td>Possible values: <code>Text</code> / <code>bool</code></td>
</tr>
<tr>
<td>Multivalue</td>
<td>Determines whether the product variable can take only exactly one or multiple values</td>
<td>Only available for type <code>Text</code></td>
</tr>
<tr>
<td>Editable</td>
<td>Determines whether the default values can be overwritten with new or additional values or not</td>
<td>Only available for type <code>Text</code></td>
</tr>
<tr>
<td>Description</td>
<td>Variable function description</td>
<td>Displayed as tooltip in opsi-configed</td>
</tr>
<tr>
<td>Possible values</td>
<td>Comma separated list of possible input values</td>
<td>If editable is set to “True”, the list can be added later within opsi-configed. Only available for type <code>Text</code></td>
</tr>
<tr>
<td>Default value</td>
<td>Default value</td>
<td>Selection list; Only available for type <code>text</code>: Free text field. Only available for type <code>Multivalue</code>: Multiple selection</td>
</tr>
</tbody>
</table>

**opsi-setup-detector: Product Icon**
Here you can select an icon to be displayed during the installation or you can accept the default icon (cogwheel) with Next step and switch to the next tab...

To select another icon, use the button Open icon directory to select the directory in which you expect to find icons. As a preselection you get a directory of 'open source' icons: 128x128, supplied with the opsi-setup-detector. Select a subdirectory and the icons will be displayed.

Now you can select an icon from the display.

After the product configuration is performed, the product can be created.

opsi-setup-detector: Create product
Figure 28. opsi-setup-detector create product

- **Path to opsi-workbench** is a drive letter or UNC path where the share `opsi_workbench` of your opsi-server is mounted.

- To the left of the button **Create opsi package** there are three possible options, which refer to the function of the button:
  - **Create Mode** is a selection area where you can specify what happens when creating the package:
    - **Create opsi product files** creates the directory tree for the new opsi package on the selected `opsi workbench` if it does not already exist. The files required for the package will be created or copied.
    - **Create opsi product files and build package** performs the operations described in the first point. Additionally, the **opsi Package Builder** is called to create the opsi package from the created directory tree. The exact processes are determined by the selection field **Build Mode**:
      - **Only build** starts the **opsi Package Builder** without interactive GUI, creates an opsi package from the directory tree via server command `opsi-makepackage` and terminates the **opsi Package Builder** after work is done.
      - **build and install** starts the **opsi Package Builder** without interactive GUI, creates from the directory tree via server command `opsi-makepackage` an opsi package installs the package via server command `opsi-package-manager` and finishes the **opsi Package Builder** after the work is done.
  - **Create opsi product files and start interactive package builder** performs the operations listed in the first item. Additionally the **opsi Package Builder** is called interactively. You have to quit it yourself to return to the **opsi-setup-detector**. For installation, configuration and operation of the community project **opsi Package Builder** check [https://forum.opsi.org/viewforum.php?f=22](https://forum.opsi.org/viewforum.php?f=22)
  - **Create opsi package** is the button that initiates the package creation. If a package with this name already exists, you will be asked if the files in the existing directory should be backed up or deleted:
More details about `opsi-setup-detector` can be found in the opsi manual:

9.2.3. The opsi PackageBuilder program to modify a script.

At the first start after the installation the opsi PackageBuilder starts in offline mode, because important configuration data for the connection with the opsi-server is missing.

![Figure 29. opsi PackageBuilder  First Start: Offline Mode](image)

If the startup does not work this way and the start menu does not respond (observed under Linux / KDE), try it from the command line by specifying a path and confirm the error message that the path was not found:

```
opsipackagebuilder --path /home
```

**Initial configuration of the opsi PackageBuilder**

To enter the missing configuration data open the `Settings`. 
In the General tab please enter the following settings:

- **configserver**: full name (FQDN) of your opsi-configserver (e.g. opsi.mycompany.org).
- **opsiadmin user**: username of a member of the group opsiadmin (preferably your username)
- **opsiadmin password**: the password of the user specified above. This will not be displayed and is stored encrypted. It's necessary for the opsi PackageBuilder to communicate with the opsi-server.
- **opsi Server Version**: opsi 4.1 or higher
- **opsi Workbench**: /var/lib/opsi/workbench
- **command execution compatibility**: opsi 4.0.4 or newer / Sudo without password
- **User**: your full name (used in changelogs)
- **Email**: your email address (used in changelogs)

In the Program tab please enter the following settings:

- **Use existing network drive**: Check the box.
- **Development folder**: Path to the directory where the opsi packages should be created. This is ideally the path to where the opsi_workbench of your opsi server is mounted.
- **script editor**: The script editor of the opsi PackageBuilder is only available for Windows unfortunately.
  - Under Windows leave it with the default settings.
- Under Linux: External editor: `/usr/local/bin/jedit`
  Command line options: (empty)
- On MacOS: External editor: `/Applications/jedit`
  Command line options: (empty)

**Figure 32. opsi PackageBuilder Settings: Management**

In the Administration tab, we recommend the following setting, deviating from the default

- **Package**: `opsi-makepackage -v`.

Save the settings and restart the opsi PackageBuilder. The opsi PackageBuilder should now no longer report *Offline mode*.

**Install, modify and pack packages with the opsi PackageBuilder.**

**Figure 33. opsi PackageBuilder Start**

Use Open package (F2) and select the directory in which you have created with the `opsi-setup-detector` a package. (e.g.: `w:\newprod2`)

The product window opens with different tabs. The default tab is Package.

**Figure 34. opsi PackageBuilder Package Tab**
In this tab you see on the left side the general metadata of the opsi product as you have already been explained in Section 9.2.2.4, “opsi-setup-detector: Product configuration 1”.

On the right side you see the script files and next to it the button:

Figure 35. opsi PackageBuilder Edit button

With the button you can invoke the file in the script editor specified in the configuration and modify the script. On Windows this is the script editor of the opsi PackageBuilder.

Figure 36. opsi PackageBuilder Script editor under Windows

Key features:

- Color syntax highlighting.
- “Folding” of source code (optional: compact, with comments)
- Lexical definition customizable (to do this, the editor must be invoked via start menu entry)
- Autocomplete for syntax elements and variables
- Freely definable and reusable code blocks (“snippets”)

The core component of the editor is the module Scintilla, which is also used in other well known editors, such as Notepad++. The lexical elements (syntax highlighting and folding) for the representation of the script language valid for opsi are however completely written in AutoIt, since Scintilla does not supply its own representation module for opsi scripts. Because AutoIt is an interpreter language, it’s slower than other editors and is therefore only conditionally suitable for editing very large scripts, especially when source code convolution is switched on. In the settings, however, it’s possible to specify whether the editor is invoke with these functions or not, provided that the call is made directly via the script tree. If the editor is open via the link in the start menu, syntax highlighting and convolution are generally switched off at startup and can be activated via the editor menu “View”.

(The editor can also be open via the command line. More information about the possible command line parameters can be check with the “-help” option).
9. Integration of New Software Packages into the opsi Server

9.2.4. Testing and improving an opsi-script script

For testing and improving a script / product there are two different variants:

- Testing the created script as 'standalone' i.e. without installing it on the opsi-server and deploying it from there to the client.
- 'Integrated' testing of the complete product with installation on the server and deployment on a client.

In both cases it will be assumed here that you have created a project with the opsi-setup-detector.
'Standalone' tests

Start the application opsi-script-gui: with double click.

- Windows: Double-click the file `opsi-script.exe`.
  (When starting the program on a Windows 7 / 10 client, "run as administrator" must be used with the right mouse button). If the `opsi-client-agent` is already installed on your computer, you will find it in `C:\Program files (x86)\opsi.org\opsi-client-agent\opsi-script\opsi-script.exe`. If not, copy from the share `\<opsiserver\opsi_depot`, from the `opsi-script\windows\x86` directory the content of this directory.

- Linux: start file `/usr/bin/opsi-script`.

- MacOS: Start the application `/Applications/opsi-script`.

You'll see the following window:

![Image of opsi-script-gui in interactive mode]

**Figure 42. opsi-script-gui in interactive mode**

- With **Select Script** you can select the script you want to run.
- With **Start** you can start the script. With it, the script will be executed on this computer.

Now open the log file with the `opsi-logviewer` to see how the opsi-script interprets the script.
Make sure that you can adjust the displayed log level with the slider in the lower right hand corner.

- Open the script `setup.opsiscript` in an editor and make the desired changes (do not forget to save). There are several ways to do this:
  - Open the project in `opsi PackageBuilder` and open the editor from there.
  - In principle you can use any other editor of your choice.
  
We recommend the editor **jEdit** with opsi-script syntax highlighting, that you can find in the basic opsi products.
You can now customize and save the script in the editor (you can leave the editor open). Switch to the opsi-script window and start the script again with the Start button (the script does not have to be selected again). View the log modified based on your changes in the script with the opsi-logviewer. (Don’t forget reload via context menu or toolbar button).

In this way, i.e. by repeating the points:
- Customize the script and save it
- Execute script
- Check log
you can gradually tailor your scripts to do what you need.

Hints for solving detailed problems can be found in the next chapter. The chapter after the following explains how to create an opsi product from the scripts created in this manner, which you can install on the opsi-server.

'Integrated' tests

With the 'integrated tests' the whole project is always executed by opsi on a test client. Proceed as follows:

- Open the script setup.opsiscript in an editor and make desired changes (do not forget to save). There are several ways to do this:
  - Open the project in opsi PackageBuilder and open the editor from there.
  - In principle you can also use any other editor. We recommend the editor jEdit with opsi-script syntax highlighting, that you can find in the basic opsi products.
- Product Packing
  - Variant 1: Open the project in the opsi PackageBuilder and start the packing via the button Pack.
  - Variant 2: Login via terminal (e.g. Putty) to the opsi-server and change to the project directory on the workbench. Pack the product with the command opsi-makepackage.
9. Integration of New Software Packages into the opsi Server

- Install the product on the opsi-server.
  - Variant 1: Start the install in the opsi PackageBuilder with the button install.
  - Variant 2: Start the install in the terminal in the project directory with the command `opsi-package-manager -i <myproctid_version.opsi>`. Where `<myproctid_version.opsi>` is the filename that was output in the previous step when packing.

- Select and start product via `opsi-configed`
  1. Select the test client in the tab Clients
  2. In the tab Product configuration select the product. If the product is not visible (which is normal after the first installation) reload the data via the menu File / Reload all data or the button on the very left of the toolbar.
  3. For the selected product set the action request setup and save.
  4. Start the client or start it via context menu on_demand if the client is running.
  5. Wait until the product has run through on the client.
     - In the tab Logfiles / instlog inspect the log file to see how the opsi-script interprets the script. Note that you can adjust the log level displayed here with the slider in the lower right hand corner.

- In this way, repetition of the points mentioned:
  - Adaptation of the script and saving
  - Pack product
  - Install product on the server
  - Run product on the client
  - check log
    you can gradually customize your scripts to do what you need.

9.2.5. Packing with `opsi-makepackage`

Afterwards you can pack the product. To do this, go to the root directory of the product and execute `opsi-makepackage`. Now the product will be packed.

It’s recommended to create the packages immediately with an associated md5 checksum file. This file is used by `opsi-package-updater` among others to ensure package integrity after package transfer. Such a file is created automatically, but for special usage scenarios its creation can be avoided.

When transferring packages on the opsi-depotserver, ‘zsync’ can be used to transfer only differences between different packages. In order to use this method, a special `.zsync` file is needed. Such a file is created automatically, but for special usage scenarios the creation can be avoided.

If there are space problems in the temporary directory `/tmp` when creating large packages, it’s possible to specify an alternate temporary directory using `--temp-directory`.

If a package of this version already exists, `opsi-makepackage` will show a query:

```
Press <O> to overwrite, <C> to abort or <N> to specify a new version:
```

With `o` you can choose to overwrite, with `c` you cancel the process and with `n` you can choose to be asked for a new product or package version.

You can install the packed package on the server with `opsi-package-manager --install <package-file>`

More details about the `opsi-makepackage` can be found in the opsi-manual:
9.2.6. Installing with opsi-package-manager

To install the packed product there is a command `opsi-package-manager`. To do this, go to the root directory of the product and execute the following command.

```
opsi-package-manager -i <myproductid_version.opsi>
```

More details about the `opsi-package-manager` can be found in the opsi-manual:

9.2.7. Example of a 'control' file

```
[Package]
version: 1
depends:

[Product]
type: localboot
id: mytest
name: My Test
description: A test product
advice:
  version: 3.14
  priority: 10
  licenseRequired: False
  productClasses:
  setupScript: setup.ins
  uninstallScript:
  updateScript:
  alwaysScript:
  onceScript:
  customScript:
  userLoginScript:

[ProductDependency]
  action: setup
  requiredProduct: javavm
  requiredStatus: installed

[ProductProperty]
type: unicode
name: mytextprop
multivalue: False
editable: True
description: hint
values: ["off", "on"]
default: ["off"]

[ProductProperty]
type: bool
name: myboolprop
description: yes or no
default: False

[Changelog]
mytest (3.14-1) testing; urgency=low

* Initial package

-- jane doe <j.doe@opsi.org> Mi, 14 Jul 2010 12:47:53 +0000
```
9.2.8. Create opsi-package with CLI tool opsi-newprod

- Do not use any country-specific symbols (umlaut), since the actual country code might vary for different code tables.

To start creating a new product, change directories to the product directory, and start the creation of the new product by entering the command `opsi-newprod`. The next question will ask you about the type of product you want to create. Choose the type `localboot` for products which should be installable by `opsi-client-agent/opsi-script`. The product type `netboot` is used for products which are activated as a bootimage (like OS installation).

**Figure 44. Choose the product type: localboot**

Confirm your choice with tab (or F12). Next, fill in the basic product parameters. At the top of the window there is an explanation for the current input field.

**Figure 45. Input of the product information**

**Product Id**
- is a distinct short name for the product, independent from the product version (we recommend to use only plain ASCII letters and '-', no white space, no special characters)

**Product name**
- is the full name of the product

**Description**
- is an additional description of the product.

**Advice**
- is some additional information on how to handle the product (a note).

**Product version**
- is the version of the packed software (max 32 chars).
Package Version

is the version of the package for the product version. For example, this helps to distinguish between packages with the same product version but with modified *opsi-script* scripts.

License required

is only relevant to netboot products.

Priority

controls the installation sequence. Possible Values are between 100 (at the very beginning) and -100 (at the end). Note: product dependencies also have influence on the installation sequence. See the opsi manual for more information.

After the product information is completed, fill in which action scripts should be provided:

![Figure 46. Input of the opsi-script script names for different actions](image)

After editing the product information you should mention the script you want to use for different activities.

Usually the **Setup script** is named `setup.opsiscript`

Usually the **Uninstall script** is named `uninstall.opsiscript`

An **Update-Script** will be used for minor changes on existing big installations. If this product is switched to the required action `setup`, then the update script will be automatically executed after the setup script.

An **Always-Script** will be executed at the beginning of every activity of *opsi-client-agent* (e.g. on every boot).

A **Once-Script** has the resulting state `not_installed`. It is a very special kind of script, and you should only use it if you really know what you are doing.

A **Custom-Script** doesn't change the resulting state. It is a very special kind of script, and you should only use it if you really know what you are doing.

A **userLoginScript** is used to modify the user's profile after the user logs into the system. It only works with the opsi extension *User Profile Management*, which is described at the *User Profile Management* chapter in the opsi-manual.

<table>
<thead>
<tr>
<th>Type</th>
<th>resulting state</th>
<th>resulting action</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup</td>
<td>installed</td>
<td>none</td>
</tr>
<tr>
<td>uninstall</td>
<td>not_installed</td>
<td>none</td>
</tr>
<tr>
<td>update</td>
<td>installed</td>
<td>none</td>
</tr>
<tr>
<td>always</td>
<td>installed</td>
<td>always</td>
</tr>
<tr>
<td>once</td>
<td>not_installed</td>
<td>none</td>
</tr>
<tr>
<td>custom</td>
<td>unchanged</td>
<td>unchanged</td>
</tr>
<tr>
<td>User login</td>
<td>unchanged</td>
<td>unchanged</td>
</tr>
</tbody>
</table>
The next step is to define one or more product dependencies. If there are no product dependencies, select No.

![Image of product dependency selection](image)

Figure 47. Create product dependency: No/Yes

To create a product dependency, enter the following data (help is available at the top of the window):

![Image of dependency creation interface](image)

Figure 48. Data needed to create a dependency

**Dependency for Action**
Which product action shall the dependency create, or when should the dependency be checked (only setup).

**Required product id**
Product id of the required product.

**Required action**
Select the required action (setup) for the required product. If no required action is set, a required installation status must be set.

**Required installation status**
Select the required status of the required product (installed). So the required product will be installed if it isn't installed on the client yet. If no required installation status is set, a required action must be set.

**Requirement type**
This is regarding the installation order. If the required product has to be installed before the installation of the actual product,
this is set to before. If it has to be installed after the actual product, set requirement type to after. Leave it blank if the installation order doesn't matter.

The possibility to define uninstall actions or dependencies is broken. After defining a product dependency, you will be asked if you want to create another product dependency. If you choose Yes, then the procedure for defining a product dependency is repeated. If you choose No, then you will be asked to define some product properties, which means defining additional switches for product customization.

The installation sequence results from a combination of product dependencies and product priorities. For details on how this is done, and what you can configure, see the opsi-manual.

![Image](https://via.placeholder.com/150)

**Figure 49. A(nother) product property to create?**

If you answer Yes, you will have to describe the product properties.

The product properties are client specific, and have names (keys) which can hold different values. These values can be evaluated by the opsi-script script, and result in installing different options at installation time.

First we have to decide if our property is a text value (unicode) or a logical value e.g. true/false (boolean). If you are not sure choose unicode.

![Image](https://via.placeholder.com/150)

**Figure 50. Choose the data type of the property**

Next, a description for the switch needs to be specified. This description will be shown in the opsi-configed as a help text. Next, you can define the set of values for the switch (separated by comma). If this is left blank, then any value is allowed for the switch.

If a values contains a backslash \ it has to be doubled. An example showing how a path would be defined: C:temp
Next, you can decide if the product property has a default value (switch).

If you choose `boolean` as the data type, then the description will contain only the `Property name` and `Property description`.

After defining a product property, you will be asked if you want to create another product property. If you choose Yes, then the procedure of defining a property will be repeated. If you choose No, then you will be asked for name and email of the product maintainer. This data will be written on the changelog.

Finally, the basic definitions for the new product are done.

Using the list command (`ls`), you can see the directory structure as described above. Change to the `OPSI` folder and list the content. The `control` file now contains the data you just defined, and you can load the file into an editor to view or change the entries.

### 9.3. Suggestions on How to Solve Problems with opsi-script Scripts

#### 9.3.1. Installation When the User is Logged on

Before we begin, we assume that you have tried an unattended installation using an opsi-script script, and the installation worked OK when the user had administrative privileges. However with some software products, you will see that the installation fails when started from within the opsi deployment software (opsi-client-agent). A possible reason for that difference might be that the installation process requires knowledge about the user environment or profile.
In the case of a MSI package, the option 'ALLUSERS=1' might help. Example:

```plaintext
[Actions]
DefVar $MsiLogFile$
Set $MsiLogFile$ = %opsiLogDir% + "\myproduct.log"
wmbatch_install_myproduct

[winbatch_install_myproduct]
msiexec /qb-! /l* $MsiLogFile$ /i "%ScriptPath%\files\myproduct.msi" ALLUSERS=1
```

Another possibility is that the installation starts a second process and stops before the second process is finished. So from the point of view of the opsi-script script, the task is finished while in fact the second process is still working (installing / uninstalling).

In this case, you may use the modifier /WaitSeconds <seconds>, or /WaitForProcessEnding "program.exe" /TimeOutSeconds "<seconds>", in the WinBatch section so that the script waits for the end of the second process.

Another more complex way to solve the problem is to create a temporary administrative user account and use this account for the program installation. For a detailed description on how to do this, please refer to the opsi-script manual chapter 8.3 'Script for installation in the context of a local administrator':

### 9.3.2. Customization after a silent/unattended Installation

After a successful silent installation, some customizing might be useful. The opsi-script is a powerful tool to do that job. First, find out what patches have to be applied. For example, that could mean analyzing which registry settings are affected by the GUI customizing tools.

You can use the tools shown in [opsi-getting-started-softwintegration-tutorial-analyse-and-repackage]. Some other tools can be found here:

Some other often used tools are:

- systernals
- regshot

### 9.3.3. Integration with Automated Answers for the setup Program

Another fast way of integration is to provide an automated answer file for the setup process. The answer file contains pre-defined answers. To be more precise, the answer file is used by a control tool, which waits for the setup to come up with the interactive windows. The control tool then passes input to these windows as defined in the answer file. As a control tool we recommend 'AutoIt'. The AutoIt program and the documentation can be found at: http://www.hiddensoft.com/autoit3.

AutoIt provides a lot of commands to control the setup process. Also, several error states can be handled (if known in advance) with the '[ADLIB]' section in the script.

There is, however, a fundamental challenge in using AutoIt:
The AutoIt script must provide input for every window that might pop up during installation. So if any unexpected window pops up, which isn't handled in the [ADLIB] section, AutoIt provides no input for this window and the installation stops at that point while waiting for input. This input could be done interactively by a user, and then the script can take over again and handle the next windows.

Another situation that may cause failure of an AutoIt installation:
The user can interfere with the installation if the mouse and keyboard are not disabled. Therefore we regard 'unattended' or 'silent' setup as a more stable solution.

A combination of both might do a good job:
The 'silent'-setup does the main installation and the AutoIt script handles special conditions that might occur.
If you use the opsi option of running the installation on another desktop than the current desktop, or if the current desktop is locked, then you will find that some autoit functions do not work properly under these conditions.

Therefore you should avoid using the following autoit commands in 'opsi-script' scripts:

- `winwait()`
- `winactivate()`
- `Send()`

These are the most widely used commands.

We recommend to use the `opsi-autoit-lib.au3` library, that provides replacements for this commands. You'll find this file at `C:\Program Files (x86)\opsi.org\opsi-client-agent\opsi-script\lib\opsi-autoit-lib.au3`.

This library provides substitutes:

**winwait()**

should be replaced by the function

```
opsiwinwait($title, $text, $maxseconds, $logname)
```

**Send()**

should be replaced by the function

```
opsiControlClick($title, $text, $id, $maxseconds, $logname)
```

respectively by

```
opsiControlSetText($title, $text, $id, $sendtext, $maxseconds, $logname)
```

It is always a good idea for the identification of controls to use the program `Au3info.exe` to get the 'ControlId' needed by these commands. Please use the numerical 'ControlId', because the other variants do not seem to work properly.

Example you will find in: `C:\Program Files (x86)\opsi.org\opsi-client-agent\opsi-script\lib\autoit_example_1.txt` and `autoit_example_1.txt`.

see also:

- [http://www.autoitscript.com/wiki/FAQ#Why_doesn.27t_my_script_work_on_a_locked_workstation.3F](http://www.autoitscript.com/wiki/FAQ#Why_doesn%27t_my_script_work_on_a_locked_workstation.3F)

### 9.3.4. Analyze and Repackage

When a software developer builds a setup for deployment, the developer usually knows about the required components of the software that have to be installed. But if somebody has a black box as a setup, then they need first to analyze what the setup does. This can be done by monitoring the setup activities with the appropriate tools (e.g. monitoring files and registry access) or by comparing the system states before and after installation.

To analyze the before or after states, there are several tools. For Example:

- `regshot`
- `appdeploy-repackager`

### 9.3.5. How to uninstall Products

To uninstall a software product from a computer, you need an 'uninstall' script to perform the deletion. The fundamental difficulty in software deletion is deciding what exactly has to be removed. Not all of the files that came with a software package can be deleted afterwards. Sometimes a package comes with standard modules that are also referred to by other programs. Often only the software manufacturer himself knows what parts have to be removed. The manufacturer's setup might offer an unattended uninstall option which can be embedded in the opsi uninstall script. Otherwise opsi-script provides several commands for software deletion:
Using an uninstall routine

If the product manufacturer provides an option for software deletion, you must check whether or not it can be run unattended (or in silent mode). If it requires some user interaction, an AutoIt script combined with the uninstall routine might do the job. The uninstall statement can be embedded in a [WinBatch] section of the opsi-script script:

```
[WinBatch_start_ThunderbirdUninstall]
"%SystemRoot%\UninstallThunderbird.exe" /ma
```

When using an uninstall program, always run a test to see if all of the files have been deleted and the computer is still in a stable state.

Products that are installed by MSI normally come with an uninstall option, which is usually the program `msiexec.exe` combined with the parameter `/x`. The parameter `/qb-!` is for the unattended mode (or without user interaction). So here is an example of an unattended uninstall command:

```
msiexec.exe /x some.msi /qb-! REBOOT=ReallySuppress
```

Instead of the package name, you could also use the GUID (Global Unique ID) with `msiexec.exe`. This GUID identifies the product in the system, which can be found in the registry directory `HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall`.

A request using the GUID looks like this:

```
msiexec.exe /x {003C5074-EB37-4A75-AC4B-F5394E08B4DD} /qb-!
```

If none of these methods are available or sufficient, the uninstall can be done using an opsi-script script as described below:

**Useful opsi-script commands for uninstall**

If a product has been installed by opsi-script functions, or if there is no uninstall routine for the product, the complete uninstall has to be done by an opsi-script script. opsi-script comes with some powerful uninstall functions. This chapter provides a brief overview of the uninstall functions, and more detailed information can be found in the opsi-script handbook.

Basic uninstall means deleting one or more files from the file system. This command can be executed from an opsi-script files section:

```
delete -f <file name>
```

or to delete a directory including sub directories:

```
delete -sf <dir name>\
```

The parameter 'f' means 'force' or to delete the files even if they are marked as 'read only' and the parameter 's' means including the 'subdirectories'. A file or directory can be deleted from all user profiles using the option '/AllNTUserProfiles' (see opsi-script manual for details).

Directories containing files with the attribute 'hidden' or 'system' can be deleted by using a 'DosInAnIcon'-section:

```
[DosInAnIcon_deleteDir]
rmdir /S /Q "<List>"
```
To stop a running process before deletion use the `killtask` command with the process' name (look at the task manager for process name):

```
KillTask "thunderbird.exe"
```

If the product or part of it, runs as a service, you will have to stop the service before deleting the files. One way to do so, is to set the service state to inactive in the registry and restart the computer. Or to stop the service by using the command 'net stop', which doesn't need a reboot:

```
net stop <servicename>
```

Deleting DLL files also requires special attention, since DLLs could also be used by other products. There is no general way of handling this.

To delete registry entries with opsi-script you can use the command `DeleteVar`. This command deletes entries from the currently open key:

```
DeleteVar <VarName>
```

To delete a registry key with all sub keys and registry variables, you can use the opsi-script command `DeleteKey`:

```
DeleteKey [HKLM\Software\Macromedia]
```

### 9.3.6. Known Issues with the 64 Bit Support

The opsi installer opsi-script is a 32 bit program. There are no known problems when installing 32 bit software on a 64 bit system using opsi-script. For the installation of 64 bit software, some constants (like `%ProgramFilesDir%`) give wrong values.

opsi-script have special commands to handle these problems. So read the [opsi-script manual](#) for these issues.
10. More Information

The opsi manual contains a wide array of additional information that is important for use in production. If you are using your opsi server in production, we recommend that you familiarize yourself with the 'opsi-backup' tool in order to create a backup of your data.

If you cannot find what you are looking for or need help, please visit the opsi community.

For production installations we recommend professional support by uib with a maintenance and support contract.