opsi Getting Started opsi-Version 4.1
Contents

1 Copyright 1

2 Introduction 2

3 Conventions of this document 3

4 Requirements 4

  4.1 Supported distributions for server 4
  4.2 Hardware requirements 5
  4.3 Configuration requirements 5
    4.3.1 Valid DNS domain name 6
    4.3.2 Valid DNS hostname 6
    4.3.3 Correct name resolution for the server 6
    4.3.4 Localization settings 7
  4.4 Needed network ports 7

5 opsi-server Base Installation 8

  5.1 opsi-server Basic installation 8
    5.1.1 Starting up the uib preconfigured Virtual Machine 8
      5.1.1.1 First Start 8
      5.1.1.2 Language selection 9
      5.1.1.3 First boot 9
      5.1.1.4 Second Start 11
      5.1.1.5 Terminal Window 12
      5.1.1.6 Check the Network Connection 13
      5.1.1.7 Update the opsi-Server 13
      5.1.1.8 Install the standard opsi-products 14
      5.1.1.9 Starting opsi-Server Interface 14
    5.1.2 Installation on a Debian / Ubuntu 14
    5.1.3 Installation on a Univention Corporate Server (UCS) 16
      5.1.3.1 Installation through Univention App-Center 17
5.1.3.2 Upgrading an existing opsi-Installation from UCS 3 to UCS 4 (over the App-Center) ........................................ 17
5.1.3.3 Manual opsi-installation on UCS (without App-Center) ................................................................. 17
5.1.3.4 Hints about installing opsi on an UCS server with the role member ......................................................... 20
5.1.3.5 PXE-Boot configuration for operating system installation ................................................................. 20
5.1.3.6 Synchronising data from LDAP to opsi .................................................................................. 20
5.1.4 Installation on openSUSE or Suse Linux Enterprise Server (SLES) ............................................................. 21
5.1.5 Installation on CentOS or RedHat Enterprise Linux (RHEL) ................................................................. 22
5.2 Update and Configuration of the opsi-server .................................................................................. 24
5.2.1 Backend Configuration .................................................................................................................. 24
5.2.2 Set Samba Configuration and Change Passwords ............................................................................. 26
5.2.3 Create Users and administrate the groups opsiadmin / pcpatch ..................................................... 27
5.2.4 DHCP Configuration .................................................................................................................. 28
5.2.4.1 Using a DHCP Server at the opsi-server .................................................................................. 28
5.2.4.2 Using an External DHCP Server .......................................................................................... 28
5.2.4.3 Checking the Backend Configuration for DHCP Entries ........................................................ 29
5.2.5 Configure how the opsi-server gets the Client’s IP-Address ............................................................. 29
5.3 Install the minimal opsi-products ...................................................................................................... 30
5.3.1 Automatic import of the minimal opsi products ........................................................................... 30
5.3.2 Manual Installation of opsi-products ............................................................................................. 31
6 Management interface opsi-configed ...................................................................................................... 32
6.1 Installation of the management interface opsi-configed ........................................................................ 32
6.2 Start of the management interface opsi-configed ............................................................................... 32
7 Add clients to opsi ................................................................................................................................. 33
7.1 Creation of a new opsi client ............................................................................................................ 33
7.1.1 Creating a new opsi client via the graphical management interface .................................................. 33
7.1.2 Creating a new opsi client via the command line .......................................................................... 34
7.1.3 Create a New Client using the opsi-client-bootcd ........................................................................... 34
7.2 Integration of Existing Clients .......................................................................................................... 36
7.2.1 Usage of the service_setup.cmd on Windows NT6 ........................................................................... 36
7.2.2 Usage of the service_setup_NT5.cmd on Windows NT5 .............................................................. 37
7.2.3 Usage of the opsi-deploy-client-agent .......................................................................................... 37
8 Rollout existing products ..................................................................................................................... 40
8.1 Usage of opsi standard products: opsi-configed ............................................................................ 40
8.2 Hard- and Software Inventory with the Products hwaudit and swaudit ........................................... 40
8.3 Hardware Inventory with the Netboot Product hwinvent .................................................................. 41
9 Installation of a new Windows PC through opsi (OS Installations) 42

9.1 OS-Installation: Complete the Base Package for Windows 42

9.2 NT6 family: as of Win7 / 2008R2 42

9.2.1 Creating a PE 43

9.2.1.1 Automated PE creation using opsi 43

9.2.1.2 Manual PE creation for Windows 10 & Windows 8 (ADK) 43

9.2.1.3 Manual PE creation for Windows 7 (WAIK) 44

9.2.2 Extending a PE 45

9.2.3 unattend.xml 46

9.2.4 Driver Integration 46

9.2.5 Providing the Installation Files 46

9.2.6 Installation Log files 46

9.3 Windows Product Key 46

9.4 Start the Windows Installation 47

9.5 Structure of the Unattended Installation Products 48

9.5.1 Directory Tree Overview 48

9.5.2 File Descriptions 48

9.5.3 Directory installfiles / winpe 48

9.5.4 Directories opsi and custom 49

9.5.5 Directory drivers 49

9.6 Simplified Driver Integration during the unattended Windows Installation 49

9.6.1 General Driver Packages 49

9.6.2 Preferred Drivers 50

9.6.3 Drivers that will be manually assigned to clients 50

9.6.4 Drivers which will be Automatically Assigned to the Clients using the Fields <vendor>/<model> 50

9.6.5 Structure of the Driver Directory and Driver Files 51

9.6.6 Processing of the Different Levels of Driver Integration 51

9.6.7 Driver Addition and Checking 51

10 Integration of New Software Packages into the opsi Server 55

10.1 A Brief Tutorial: How to write a opsi-winst Script 55

10.1.1 Introduction 55

10.1.2 Methods of Non-Interactive Installation 55

10.1.3 Structure of a opsi-script / opsi-winst Script 56

10.1.4 Primary Sections 56

10.1.5 Important Kinds of Secondary Sections 57

10.1.6 Global Constants 58

10.1.7 Second Example: tightvnc 58

10.1.8 Elementary Commands for Primary Sections 59
Chapter 1

Copyright

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

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

A German description can be found here: https://creativecommons.org/licenses/by-sa/3.0/de/

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

The English description can be found here: https://creativecommons.org/licenses/by-sa/3.0/

The English license can be found here: https://creativecommons.org/licenses/by-sa/3.0/legalcode

Most parts of the opsi software are open source.
The parts of opsi that are not open source are still under a co-funded development. Information about these parts can be found here: opsi cofunding projects

All the open source code is published under the AGPLv3.

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

Some information around the AGPL: http://www.gnu.org/licenses/agpl-3.0.en.html

For licenses to use opsi in the context of closed software please contact theuib gmbh.

The names opsi, opsi.org, open pc server integration and the opsi logo are registered trade marks ofuib gmbh.
Chapter 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 perform.

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.
Chapter 3

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.
Chapter 4

Requirements

Subsequently the requirements for the installation from opsi in a server will be described.

4.1 Supported distributions for server

As of 30.08.2019

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Opsi 4.1</th>
<th>Opsi 4.0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian 10 <em>Buster</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debian 9 <em>Stretch</em></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 8 <em>Jessie</em></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 7 <em>Wheezy</em></td>
<td>x</td>
<td>▲</td>
</tr>
<tr>
<td>Ubuntu 18.04 LTS <em>Bionic Beaver</em></td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Ubuntu 16.04 LTS <em>Xenial Xerus</em></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu 14.04 LTS <em>Trusty Tahir</em></td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu 12.04 LTS <em>Precise Pangolin</em></td>
<td>x</td>
<td>▲</td>
</tr>
<tr>
<td>RHEL 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHEL 6</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>CentOS 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CentOS 6</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>SLES 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLES 12SP4</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLES 12SP3</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLES 12SP2</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLES 12SP1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLES 12</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
4.2 Hardware requirements

For a opsi-server the following hardware is recommended:

- Intel-x86-compatible PC
- 2GB RAM or higher
- a hard disk with 60 GB capacity or more
  - An opsi-server should have at least a minimum free space of 16 GB in the directory `/var/lib/opsi`

The requirements of the server are moderate in testing environments. In the case of production environments it is recommended to increase the capabilities of the host system.

We recommend in the case of testing with a Virtual machine, that the host computer should have at least a dual core processor and at least 4GB of RAM. For testing purposes, a test client can be run as another Virtual machine on the same host computer.

4.3 Configuration requirements

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

Your DNS domain name must consist of at least one domain and one toplevel domain. In other words: the fully qualified domain name must contain at least one point. Furthermore, the toplevel 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:

4.3.2 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 domain name, in which at least come two dots, e.g. `opiserver.domain.local`:

```
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

4.3.3 Correct name resolution for the server

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

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

The result should look like the following example:

```
192.168.1.1 server.domain.tld server
```

Dabei sollte die IP-Adresse der Netzwerkschnittstelle aufgeführt sein, zu der sich die Clients später verbinden sollen.

Sich das Ergebnis nicht so aus (enthält z.B. `127.0.0.1`, `127.0.0.2` oder `localhost`), dann müssen Sie die Datei `/etc/hosts` oder Ihre Namensauflösung korrigieren.

The result has the scheme:

```
<IP-Number> <full qualified hostname> <hostname>
```

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).

Note

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.

Note

Opsi does not required an Active Directory or similar. Integrating opsi is possible but not required.
4.3.4 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:

```
test -e /etc/default/locale && echo "ok" || (echo "Check locales:" && locale)
```

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 no or an undesired value is set:

```
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 manuals of your Linux distribution.

4.4 Needed network ports

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

- **opsi-server web service**: TCP 4447
  Client to server, depot to server (bidirectional, connections via localhost).
- **opsi-client web service**: TCP 4441
  Server to client, connection from client to itself via localhost.
- **opsi-client web service**: TCP 4442
  Connection from client to itself via localhost.
- **opsi-client Notifier**: TCP 45000 - 65536
  Connection from client to itself via localhost.
  A random port from the given range is selected.
- **TFTP**: UDP 69
  Client to server.
- **CIFS/SMB**: UDP 137 / UDP 138 (netbios) / TCP 139 / TCP 445
  Client to server (bidirectional).
  Depends on the version of the client operating system.
- **WEBDAV**: TCP 80
- **WINEXE**: UDP 137 / UDP 138 (netbios) / TCP 139 / TCP 445
  Server to client (bidirectional).
  Depends on the version of the client operating system.
- **SSH (optional)**: TCP 22
- **DNS**: TCP 53
- **WakeOnLan (WOL)**: UDP 7 / UDP 9 / UDP 12287
  Server to Client. These ports are configurable.
- **HTTP**: TCP 80
  E.g. To download server updates from [http://download.opensuse.org/](http://download.opensuse.org/)
- **HTTPS**: TCP 443
  To download updates from [https://download.uib.de](https://download.uib.de) (opsi-package-updater)
Chapter 5

opsi-server Base 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.

5.1 opsi-server Basic installation

In this section different variants of the installation of an opsi-servers are shown. You’ll 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.

Please set the apropriate environment variables http_proxy and https_proxy on your opsi-server.

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

5.1.1 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 ESXi.

5.1.1.1 First Start

VMware

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

• Download the opsi-ServerVM from opsi.org

• Unzip the file and a directory opsivm will be generated.

• Start the VMware player. Open ‘Open a Virtual Machine’ and search for the file opsivm.ovf in the opsivm directory. You can import the server with an new name. The virtual machine will still boot.

ESXi-Server
• Download the opsi-ServerVM from opsi.org
• Unzip the file and a directory `opsivm` will be generated.
• Start the vSphere Client.
  Install a new client with `File / Deploy OVF Template...` and answer the following questions.

**Virtualbox**

• Download the opsi-ServerVM from opsi.org
• Unzip the file and a directory `opsivm` will be generated.
• Start the 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.

5.1.1.2 **Language selection**

The first step is to choose the preferred language:

![Language selection](image)

> Figure 5.1: Language selection

5.1.1.3 **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.

⚠️ **Warning**

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

The log file of `1stboot.py` is located at `/var/lib/1stboot/1stboot.log`. 
Figure 5.2: Startup mask

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

Figure 5.3: 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**
Net mask of this server e.g. **255.255.255.0**

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

**country**
For the creation of the SSL-certificate: Identification of the nation (2 capital letter) e.g. **DE**

**state**
For the creation of the SSL-certificate: Identification of the federal state e.g. **RPL**

**city**
For the creation of the SSL-certificate: Identification of the city e.g. **Mainz**
organization
For the creation of the SSL-certificate: Identification of the company e.g. uib gmbh

organizational unit
For the creation of the SSL-certificate: Identification of the bureau (optional)

e-mail address
For the creation of the SSL-certificate: mail address (optional)

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

proxy
If required for the 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 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.

5.1.1.4 Second Start

After the reboot, login as adminuser with your password.

The graphical user interface of the opsi-server should have already started (implemented as a sustainable window manager). A 'Firefox' browser window might appear at startup, and display further instructions and information. This information can serve as a reference to the getting started document (the document you are currently reading).

If you get a message that there is no network connection, try rebooting the server. This might solve the problem.
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.

### 5.1.1.5 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 Section 3.1.1.4 of the last chapter)
• 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 many working applications 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`.

5.1.1.6 Check the Network Connection

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

If the Internet connection is not working, then you have to open a terminal window (maybe remote access isn’t possible, except using the server terminal window) 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 Section 5.2.

5.1.1.7 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 introduce the current password for the adminuser.

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
```
5.1.1.8 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 introduce the current password for the adminuser. Through this installation the actual stock of opsi-products, incl. templates for OS deployments, will be downloaded from the opsi repositories and installed on the server.

For more information see Section 5.3.

5.1.1.9 Starting opsi-Server Interface

For a description of the Server Interface check Section 6.2.

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

- Section 7.2
- Chapter 9

5.1.2 Installation on a Debian / Ubuntu

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

Important
Please check the requirements.

We recommend the following software installations:

```
apt install wget host pigz
```

opsi needs samba, which can be installed as:

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

Now install the MySQL-Server which is used for inventory- and license management as follows:

```
apt install mysql-server
```

Check the opsi-server entry in /etc/hosts, or the output of

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

The result should look like the following example:

```
192.168.1.1 server.domain.tld server
```

The result has the scheme:

```
<IP-Number> <full qualified hostname> <hostname>
```

If the result looks different than 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).

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

Ubuntu 18.04 LTS Bionic Beaver:

```
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1/stable/xUbuntu_18.04/ /" > /etc/apt/sources\ .list.d/opsi.list
```
Ubuntu 16.04 LTS Xenial Xerus:

```
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/xUbuntu_16.04/ /" > /etc/apt/sources.list.d/opsi.list
```

Debian 10 Buster:

```
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/Debian_10/ /" > /etc/apt/sources.list.d/opsi.list
```

Debian 9 Stretch:

```
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/Debian_9.0/ /" > /etc/apt/sources.list.d/opsi.list
```

Debian 8 Jessie:

```
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/Debian_8.0/ /" > /etc/apt/sources.list.d/opsi.list
```

Execute the following command in order to import the signature key of the repository:

Ubuntu 18.04 LTS Bionic Beaver:

```
apt-key add - < Release.key
```

Ubuntu 16.04 LTS Xenial Xerus:

```
sudo apt-key add - < Release.key
```

Debian 10 Buster:

```
apt-key add - < Release.key
```

Debian 9 Stretch:

```
apt-key add - < Release.key
```

Debian 8 Jessie:

```
apt-key add - < Release.key
```

All:
Check for key import success:

```
apt-key list
```

should contain the output:
```
```

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
```

Execute the following commands in order to install opsi at your server:
apt update
apt remove tftp
# Only needed when a tftp line is present in the inetd configuration
update-inetd --remove tftp
apt install opsi-tftpd-hpa
apt install opsi-server
apt install opsi-windows-support

If you are asked for the tftp directory during the tftpd-installation answer with /tftpboot. During the installation of the opsiconfd, you will be asked for information for the creation of a local SSL certificate.

During the opsi-server installation, you have to allow the patching of the file smb.conf. Answer the question with yes. Also, you will be asked for a password for the user pcpatch. Set a new password, and please remember this password when continuing with the following sections.

**Debian 8 (Jessie) specialities**: The bootimage has issues to mount the opsi_depot-Share over mount.cifs. To avoid these problems you can either configure ID mapping in Samba or disable winbind. If you do not rely on winbind we recommend to disable the daemon.

Disable starting of winbindd:

```bash
systemctl disable winbind
```

or

```bash
insserv -r winbind
```

For a sane ID mapping configuration specify a configuration with limited mapping range in smb.conf and then restart Samba.

To configure ID mapping you can insert the following into the [global] section of smb.conf.

```plaintext
idmap config * : range = 1000-1999999
```

Assuming all of the above steps completed successfully, we can assume that the network is properly configured. Next continue on with Section 5.2

### 5.1.3 Installation on a Univention Corporate Server (UCS)

**Important**
Please check the **configuration requirements**!

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

Both are equally supported methods of installations. We recommend using only one method per server. The difference is that an installation of opsi on a *member* server is not possible if the App Center is used. If new packages for an operating system are released they are available right away if the repositories maintained by uib are used. If the installation is made through the App Center the change to a newer UCS version (i.e. from UCS 4.2 to UCS 4.3) will be blocked until all installed apps are supported on the new version of the operating system.

**Note**
Installations on a system with 32bit and 64bit are currently both supported.
In the future the support for 32bit will be dropped. Therefore we recommend to only use opsi on a system with a 64bit architecture.
With all installation methods the package `opsi4ucs` will be installed. This will prepare the use of opsi by using a join script. This preparation includes adding required users, groups and shares.

The first opsi server in a domain will have it’s backend configured to make use of the installed MySQL server. All following servers will be registered as depots in opsi.

When updating from opsi 4.0 the join script will execute the migrations of the backends.

### 5.1.3.1 Installation through Univention App-Center

In the Univention App-Center an automatic installation of the opsi-Server is available. The installation-app for opsi can be found in the UCS-Management-Webinterface in the category `System`. Over the App-Center opsi can be installed on server roles `master`, `backup` and `slave`.

If you want to update a existing opsi4ucs-installation please check the next chapter for further information.

Additional packages will be installed: `opsi-tftpd-hpa`, `opsi-windows-support`, `univention-mariadb` respectively `univention-mysql`.

If an existing opsi-server is detected `opsi-package-updater` will be configured to retrieve packages from this server. On installation `opsi-package-updater` will be called to download and install opsi packages. This will not be done after an update. On an major or minor update package updates will installed automatically.

Please note that no automatic transfer of clients to opsi takes place. More information at Section 5.1.3.6.

The opsi installation on a UCS server via the Univention App Center is now complete. Continue with Section 7.2

### 5.1.3.2 Upgrading an existing opsi-Installation from UCS 3 to UCS 4 (over the App-Center)

Since opsi 4.0.5 the group `opsifileadmins` replaces the group `pcpatch` in UCS. This group has already been introduced with the support of UCS 3.0, but only on installations that had Samba 4 and the Univention Directory Services (Samba4-AD). In all other variants and roles the group continues to be, as it was before `pcpatch`.

Since this situation represents a problem not only on the installation, but could also lead to potential problems with migrations (especially of Samba3 on Samba4) since the release of 4.0.5 the group `pcpatch` will be created as `opsifileadmins`.

---

**Warning**

To implement the integration package in a clean way, an already existing group `pcpatch` will be renamed automatically to `opsifileadmins`. This is done via the join script. If your config server is run on role `master` or `backup` the join script will be executed automatic.

The main reason for this drastic measure is that the manual rename of this group is not trivial, because it is a primary group. Therefore it is recommended before installing this update to make sure that your group is still named `pcpatch`. If so, the update should be started with the config server and soon afterwards on the depot servers as well. Otherwise the operation on multi depot-environments could lead to issues. This should not be the case, if your group is already named `opsifileadmins`. Nevertheless, it is recommended after importing the update to check every opsi server to verify complete functionality.

---

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

---

**Important**

Please check the requirements.

---

Necessary preparations:
• The command

```bash
classic_installation_user:pcpatch
```

must return a full qualified domain name containing two dots, e.g. `opsiserver.domain.local`

• The command

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

has to show the IP-address of the network interface which the client has to connect with. If the command shows the address `127.0.0.1` or `127.0.0.2`, you have to correct your name resolution in `/etc/hosts` file.

• Samba has to be configured. For the use of 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 daemon `dhcpd` has to be configured and should be running.

The installation of opsi is possible on the roles `master`, `backup`, `slave` and `member`. For the installation on a `member` you need to read Section 5.1.3.4!

The following documentation describes an installation on a `master` with Samba4.

---

**Caution**

When installing on role `slave` the server must be already joined and Samba 4 has to be installed first. UCS configuration is usually done on the `master` while the installation and configuration of opsi happen on the `slave`.

---

The classic installation with the user: `pcpatch` in the primary group: `pcpatch` does not work with UCS. Samba4 has placed fundamental restrictions on the Active-Directory, so groups with the same name as a user are no longer allowed. For this reason the configuration file `/etc/opsi/opsi.conf` has been introduced. This file controls how the group used for Samba access will be named. More specifically for UCS the group name `pcpatch` will be renamed to `opsifileadmins`. This means that users that need clearance for opsi (opsi package builder for example) can’t be members of the group `pcpatch` but must be member of `opsifileadmins`. This peculiarity applies only to UCS and is different to other distributions and chapters in the opsi-documentation. Furthermore, since UCS 3 the user `pcpatch` is created as an domain user. For more information about this new configuration file please refer the opsi-manual.

• Next add the opsi4ucs repository:*  

**UCS 4.4:**

```bash
echo "deb http://download.opensuse.org/repositori\nes/home:/uibmz:/opsi:/4.1:/stable/Univention_4.4/ " > /etc/apt/

```

**UCS 4.3:**

```bash
echo "deb http://download.opensuse.org/repositori\nes/home:/uibmz:/opsi:/4.1:/stable/Univention_4.3/ " > /etc/apt/

```

Now import the key to the repository system with the following command:

**UCS 4.4:**

```bash
wget -nv https://download.opensuse.org/repositori\nes/home:/uibmz:/opsi:/4.1:/stable/Univention_4.4/Release.key -O Release.\n
```

```
apt-key add - < Release.key
```
UCS 4.3:

```
apt-key add - < Release.key
```

For the installation the following commands must be entered:

UCS 4.4:

```
univention-install univention-mariadb
univention-install opsi-tftpd-hpa opsi4ucs
univention-install opsi-windows-support
```

UCS 4.3:

```
univention-install opsi-tftpd-hpa
univention-install univention-mariadb
univention-install opsi4ucs
univention-install opsi-windows-support
```

If the role of the target systems different than master or backup then the following commands 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 configurations 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 the opsi_depot release point must be released in UDM. To realize this settings set the link to yes under Advanced Settings → Advanced Samba Settings: follow symlinks. The same should be done for the opsi_depot_rw, so the driver integration will run without problems. If the directory `/var/lib/opsi/depot` was found 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 samba4 will not be automatically restarted, since it is a important service with potential dependencies. You have to restart it manually.

Since UCS 3 there is no direct link between the Univention LDAP backend and opsi 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 the LDAP clients in Univention does not mean that the client was also deleted under opsi and vice versa. This problem is further discussed in Section 5.1.3.6.

Since opsi was installed on an already running server we assume that the network configuration is correct. Continue with the installation be skipping forward to Section 5.2.

---

⚠️ **Warning**
The Unix commands used in the following chapters are for Debian systems. You may have to change them to match your Linux system.
5.1.3.4 Hints about installing opsi on an UCS server with the role member

Warning
Running opsi on a member server is affected by certain limitations. Therefore we recommend beginners to run their opsi systems on a different role.

Installing opsi on a server with the role member is possible. However an automated installation through the Univention App Center is currently not possible.

After an installation you need to make sure that the user that will be used to access the depot is set with the current domain. Control 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.

5.1.3.5 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 the configuration already exists.

- The opsi-tftpd-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 pxelinux.0 will be replaced with linux/pxelinux.0 instead.

You have to set guidelines to realize the mentioned configurations in the UCS-System. These guidelines are dependent on existent guidelines and have to be realized appropriately. If opsi was installed on an UCS-test system without existing guidelines you need to install DHCP-service at first. If the DHCP-service is already installed the easiest way to create guidelines in the UMC-web interface (Univention Management Console) is from UCS-server. Therefore choose the category "Domain" and subcategory the module DHCP-server. Next you have to choose the service (in a testing system you find only one entry usually). In the following detailed view choose guidelines in the menu. The needed guideline is a DHCP-Boot guideline. During the guideline configuration choose cn=default-settings as default entry (should be the only entry) and choose edit. Under the basic settings DHCP-boot for the option Bootserver enter the IP from opsi-server and insert as boot-filename pxelinux.0.

Warning
If the guideline is configured like mentioned above, this configuration Section 5.2.1 affects every device whose IP is served by the DHCP from this server. So once again this instruction should only be used for evaluation purposes which will be also testing not only opsi but also UCS. In a productive UCS environment you should not configure the guideline as described previously.

Optional you can run the udm-commands at the console. You can find more informations in the UCS-documentation.

5.1.3.6 Synchronising data from LDAP to opsi

In an opsi-ucs installation Windows-Clients have to be created in the UDM first and in a second step they have to be created in opsi-configed. 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 a wrong MAC address.

A solution for this is the extension opsi-directory-connector. Please consult the manual for more information.
5.1.4 Installation on openSUSE or Suse Linux Enterprise Server (SLES)

Important
Please check the requirements.

Necessary preparations:

- The command
  
  ```
  hostname -f
  ```

  has to return a full qualified domain name containing two dots, e.g. `opsidemo.domain.local`

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

  has to return the IP-address of the network interface the clients should connect with. If the command returns the result `127.0.0.1` or `127.0.0.2` then file `/etc/hosts` has to be corrected.

- Samba has to be installed and configured.

- mariadb-server has to be installed.

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

You can use zypper to add the opsi repositories:

openSUSE Leap 42.3:

```
```

SLES 12SP4:

```
```

SLES 12SP3:

```
```

SLES 12SP2:

```
```

SLES 12SP1:

```
```

SLES 12:

```
```

After adding the repository, you may start the opsi installation:
zypper refresh
Do you want to (r) eject the Key, (t)emporary or (a) always trust? [r/t/a/?] (a): a
zypper -v install opsi-server
zypper -v install opsi-windows-support

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

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

In case you used a tool like yast or autoyast to help you with your network configuration it’s possible the tool may have created an entry in your /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, where the server can be reached.

Please continue with Section 5.2.

---

**Warning**
The unix commands used in the following chapters are working on Debian systems. You may have to change them to match your linux system.

---

### 5.1.5 Installation on CentOS or RedHat Enterprise Linux (RHEL)

The installation from opsi on CentOS or Red Hat Enterprise Linux (RHEL) differs only on the applied repository.

---

**Important**
Please check the [configuration requirements](#).

---

Necessary preparations:

- The command
  
  `hostname -f`

  returns a fully qualified domain name containing two dots, e.g. `opsidemo.domain.local`

- The command

  `getent hosts $(hostname -f)`

  has to return the IP address of the network interface the clients should connect to. If the command returns the result `127.0.0.1` or `127.0.0.2` then file `/etc/hosts` has to be corrected.

- Install Samba and a database server:

  `yum install mariadb-server samba samba-client`

- Configure samba and database server:
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 daemon dhcpd has to be configured and be running.
- In the case of Red Hat Enterprise Linux, you must register with the Red Hat Network to have access to all required packages in Red Hat repositories:

```
rhn_register
```

Add the CentOS Repository:

**CentOS 7:**

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

Add the opsi RHEL Repository:

**RHEL 7:**

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

After adding the repository you may start the opsi installation:

```
yum remove tftp-server
yum install opsi-server
yum install opsi-windows-support
```

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:/4.1:/stable/CentOS_7/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 above steps were completed successfully we can assume that the network is properly configured. Next continue on with Section 5.2

---

⚠️ **Warning**
The unix commands used in the following chapters are working on Debian systems. You may have to change them to match your CentOS- /RHEL system.
5.2 Update and Configuration of the opsi-server

In this chapter, the installed opsi-server is set up.

5.2.1 Backend Configuration

opsi supports different backends for data management. The most important backends are:

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

Besides these main backends there are some special backends:

- **opsipxeconfd** - the service used for network boots with opsi
- **dhcpd** - used for configuring and restarting the local dhcp service at opsi-server
- **jsonrpc** - redirects all calls to another server via JSON-RPC

The default is to use the mysql backend for inventory data. The usage of the file backend for inventory data is possible but noticeably slower and therefore not recommended.

**Note**
The use of the mysql backend for inventory data is free and does not require activation. More information about the activation of co-financed modules can be found on the opsi manual.

**Note**
Some distributions use MariaDB instead of MySQL. The mysql backend will also work with MariaDB.

**Caution**
Since MySQL server version 5.7 the strict mode is enabled by default. This mode prevents the command `opsi-setup --configure-mysql` from finishing properly. To disable the strict mode please edit the file `/etc/mysql/mysql.conf.d/mysqld.cnf`. In the `[mysqld]` section add the following line underneath the section name:

```
sql_mode=NO_ENGINE_SUBSTITUTION
```

Now the service `mysql` has to be restarted: `systemctl restart mysql.service`

We will now configure the mysql backend. It is assumed that a MySQL server is installed and configured. We require the credentials for a database administrator. For specific information on installation and configuration of your 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 information to database access and then use the provided credentials to create a database and an user with appropriate rights to access that database for opsi.

The following screen shots show example parameters for a MySQL configuration setup:

![MySQL Configuration Interface](image)

Figure 5.5: 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 5.6: 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 has to be involved. Therefore, the different method calls can be used by more than one backend. These method-to-backend(s) calls are configured in the file `/etc/opsi/backendManager/dispatch.conf`.

Here an example:

```
# = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
# = 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

You will find explanations and examples at the top of this file. At 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 proved to determine which backend have to be 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.

⚠️ **Caution**
Make sure that every backend used is listed in the line starting with backend_.*.

Whenever you changed the file dispatch.conf you should execute the following commands. Even if you have not changed the file during the initial setup execute these commands now.

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

## 5.2.2 Set Samba Configuration and Change Passwords

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

```bash
opsi-setup --auto-configure-samba
```

Please restart the samba services using the following commands:

```bash
systemctl restart smbd.service
systemctl restart nmbd.service
```

**Tip**

If the server update asks if the file smb.conf should be overwritten, you have to confirm this.

If the smb.conf has been changed before, you should keep the default and synchronize 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.

A pcpatch user is created on the system. This user can install software on a client PC. The pcpatch user allows access to the configuration data on the host shares. The user pcpatch needs to get a correct password - once as system user, as samba user and as opsi user.

In a terminal window the program opsi-admin should be called which will set the pcpatch-password for opsi, unix and samba.
opsi Getting Started opsi-Version 4.1

opsi-admin -d task setPcpatchPassword

After sending the command enter the password.

5.2.3 Create Users and administrate the groups opsiadmin / pcpatch

The opsi administration is only allowed for members of the UNIX-group opsiadmin.
In the following example, we create the user adminuser, which is a similar procedure to creating an account for yourself.
Let’s create the user:

useradd -m -s /bin/bash adminuser

now set the unix password:

passwd adminuser

and now the samba password:

smbpasswd -a adminuser

Caution
Do not use the char § as part of the passwords. It becomes impossible to login at the opsi web service.

Create and test the group membership:

usermod -aG opsiadmin adminuser
getent group opsiadmin

the getent command should have a result like:

opsiadmin:x:1001:opsiconfd,adminuser

Note
In you want root to use the opsi administration commands, then root has to be a member of the group opsiadmin.

To perform everyday tasks on your opsi server, usually it is not necessary to be logged in as root. Our suggestion is to use a normal user and make use of 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 have to also be in the group pcpatch:

usermod -aG pcpatch adminuser

test the results by entering:

getent group pcpatch

The result should look like

pcpatch:x:992:adminuser

To make sudo opsi-set-rights available for users of the group pcpatch, please execute:
opsi-setup --patch-sudoers-file

opsi-set-rights, which does the same as opsi-setup --set-rights, can then be executed not only as root, but also with sudo from members of the group pcpatch (or opsi-file-admins):

Example:

```
sudo opsi-set-rights .
```

## 5.2.4 DHCP Configuration

It is essential for opsi that the DNS in combination with DHCP can be used for address lookups. To simplify the setup the opsi-server VM is delivered with a running DHCP server. In the situation where a DHCP server already exists, it should be configured to work with opsi. Both alternatives are described below.

### 5.2.4.1 Using a DHCP Server at the opsi-server

**Using the opsi-Server VM:** The opsi server VM has an installed DHCP server. The DHCP server on the opsi-server VM is configured with no free leases, so no unknown clients will get an IP-Number from this DHCP server.

If you create a client at the opsi-server using the opsi-configed, it will also create a dhcp entry for this client in the `/etc/dhcp/dhcpd.conf` and the DHCP server task will be restarted. Therefore you have to supply the IP-number and the MAC-address.

**Your own installation:** If you want to use the opsi server as DHCP server, you have to install the DHCP server package.

For example:

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

After the installation you must configure the dhcp configuration for opsi. This is done by the following command:

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

To execute the command to restart the DHCP server from opsiconfd listed in the configuration file `/etc/opsi/backends/dhcpd.conf`, an entry under `/etc/sudoers` is required. This is created using the command

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

If deemed necessary, the permissions of the DHCPD configuration file can be checked, which should look like this, for example:

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

### 5.2.4.2 Using an External DHCP Server

**Using the opsi-Server VM:** If you use an external DHCP server, then you may want to uninstall the DHCP server at the opsi-server, which is done by entering:

```
apt remove isc-dhcp-server
```

**Your own installation:** You have to configure your external DHCP server, and provide the client information to the external DHCP server, such that clients know that our opsi-server is now the boot server. 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 server, then the corresponding entries may be bootserver or startserver and bootfile or startfile (Options 66 / 67).
If you create a client at the opsi-server, then you only have to supply the MAC-address, but not the IP-number.

5.2.4.3 Checking the Backend Configuration for DHCP Entries

Regardless of whether or not you use the dhcp of the opsi-server, you have to configure the opsi-server.
The file /etc/opsi/backendManager/dispatch.conf defines which opsi method uses which backends (i.e. file, mysql).
The lines with backend_.* and host_.* entries configure how changes at host entries are handled. If you are using the DHCP server on the opsi-server, then the backend dhcpd has to be added here.
The according entry using the file backend is (e.g.):

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

If the local DHCP isn’t used, then the backend dhcpd is not required:

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

After adapting the correct backend configuration, you should execute:
```
opsi-setup --init-current-config
opsi-set-rights
systemctl restart opsiconfd.service
systemctl restart opsipxeconfd.service
```

5.2.5 Configure how the opsi-server gets the Client’s IP-Address

In the default method of the opsi software deployment, only the client must know how to contact the opsi-server.
However, if you would like to use one of the opsi push features (like send messages to the client, or fire on_demand events, or get session information, or start remote control software), then the server needs to know how to get the IP-Address of the client.

How the opsi server does this, depends on your DNS/DHCP configuration and policy. There are a large number of possible configurations. So here we show two different example configurations:

1. The clients are not known by the DNS (only by netbios), and they get dynamically assigned frequently changing IP-Numbers by the DHCP.
2. The DNS always provides the correct IP-Address of a client.

To configure the opsi server to your situation, you may change the following parameters:

- The entry resolveHostAddress in the file /etc/opsi/backends/hostcontrol.conf
  This option controls whether the name resolution of a opsi-client address is primarily done by the opsi database or by the name resolution of the operating system of the opsi-server.
  If this option is True, the opsi-server first tries to get the IP-Address of an opsi-client using the name resolution from the operating system (DNS, /etc/hosts). If the operating system DNS name resolution fails, then the opsi database is used.
  To use the opsi database during the first attempt, you have to set this option to False.
• The entry `update ip` at the file `/etc/opsi/opsiconfd.conf`
  If this entry is `yes`, then the opsi-server will update its own IP database whenever the opsi-server gets a client
  IP-Address (e.g. at every web service contact of a client). The default is `yes`.

If you are running configuration example 1, then you should probably set `resolveHostAddress` to `False` and `update ip` to `yes`.
If you are running configuration example 2, then the best configuration is to set `resolveHostAddress` to `True` and `update ip` to `no`.
You should decide for yourself which combination fits the needs of your situation.
If you changed anything while configuring your environment, then you should reload the opsiconfd:

```bash
systemctl restart opsiconfd.service
```

### 5.3 Install the minimal opsi-products

To deploy software with opsi prepared packages exist. They contain amongst other things the agent (`opsi-client-agent`),
which is required for managing clients.

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

#### 5.3.1 Automatic import of the minimal opsi products

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

You should now download and install the opsi products with the command:

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

If the `opsi-package-updater` fails, it may be necessary to add a required proxy to the `.repo` configuration files
under `/etc/opsi/package-updater.repos.d/`. Since opsi-utils version 4.1.1.33 a global proxy can be configured in
`/etc/opsi/opsi-package-updater.conf`.

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

If you want to update installed packages sometime later you can use the following command:

```
opsi-package-updater -v update
```

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

**Note**

Please notice that the OS-Installation products like win10 aren’t ready for action after installation. The installation has
to be supplemented by the installation files from the corresponding installation media (i.e. DVD, see Section 9.1).
5.3.2 Manual Installation of opsi-products

There is also the possibility to manually download and install the packages.

Download the current opsi packages in the .opsi package format. The packages can be found at https://download.uib.de/opsi4.1/stable/packages/windows in the directories netboot/, localboot/ and for Linux-clients also in https://download.uib.de/opsi4.1/stable/packages/linux.

We recommend storing the .opsi-files at /var/lib/opsi/repository. To make sure the access rights allow opsiconfd to access these files run opsi-set-rights /var/lib/opsi/repository.

After the download you must to manually 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
```
Chapter 6

Management interface opsi-configed

Opsi offers with the opsi-configed 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.

**Tip**
When using a virtual machine, make sure that the virtual screen has a sufficient screen resolution. For the opsi-configed a resolution of 1024x768 pixels (at least) 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.

### 6.1 Installation of the management interface opsi-configed

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

Alternatively you can find corresponding installer under https://download.uib.de/opsi4.1/misc/helper/.

**Important**
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 a PC is equipped with the management interface, further PCs can have the interface installed Section 8.1, if the opsi agent is already installed on the PC.

### 6.2 Start of the management interface opsi-configed

Start opsi-configed from the shortcut in your Start menu. Login 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.

**Note**
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 *Reload data* button.
Chapter 7

Add clients to opsi

To be able to manage computers with opsi, they must be known to the system. Besides, the agent must run on these computers so that communication between server and client is possible. No administration is possible without the client agent.

Depending on the environment in which opsi is to be used, there are different procedures. If clients already exist in the environment with an installed operating system that are to be managed with opsi immediately, they can be integrated in different ways Section 7.2.

The alternative to this is that the computers to be managed by opsi are equipped with a new operating system. Within the scope of this operating system installation, opsi installs the required agent at the same time, but removes all previously existing software (incl. operating system). With this procedure you first add a Client to opsi and then do a Operating system installation.

7.1 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 a later administration. 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 chapter integration of existing Clients.

7.1.1 Creating a new opsi client via the graphical management interface

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

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

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

The client will be created in opsi. If the client is configured as a PXE-client, then it will also be configured in the DHCP on the opsi-server if the opsi-server also is DHCP-server.

To see all created clients in opsi-configed choose the tab Clients, which puts opsi-configed in the mode Configuration of clients, and reload the data by pressing F5 or use the context menu.
7.1.2 Creating a new opsi client via the command line

A client can be added to opsi 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 will usually resort to a default - for most this will be 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 `00:53:00:00:00:00` and the IP address `192.0.2.1`.

```
opsi-admin -d method host_createOpsiClient testclient.domain.local "null" "Testclient" "" 00:53:00:00:00:00 192.0.2.1
```

7.1.3 Create a New 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/opsi4.1/boot-cd/`.

Download the latest and burn it to a CD.

Start the computer from the CD.

You should see the following picture:

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

Figure 7.1: Start image opsi-client-boot-cd
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. You have to complete the missing data. You must at least give the hostname.

![Figure 7.2: bootimage/boot-cd configuration screen](image)

Confirm with *OK*.

![Figure 7.3: bootimage/boot-cd: Choose how to create Client](image)

Choosing *Admin account* the client will register itself at the opsi-server using provided credentials.

![Figure 7.4: bootimage / boot-cd: Authenticate as member of opsiadmin group](image)

Therefore you will get a login window, where you should authenticate yourself as a member of the opsiadmin group. If the authorization is successful, then the client gives its data to the server, at which point the client will be created at the server. In the next step, the server provides a list of Netboot products to the client.
7.2 Integration of Existing Clients

To include existing Windows clients in opsi, the opsi-client-agent must be installed on them. This can be done in several ways. After you have installed the opsi-client-agent as described below, the client will also appear in the client list of the opsi-configed. This can be realized in several ways. After you have installed the opsi-client-agent as described below, the client will also appear in the client list of the opsi-configed, unless you haven’t done that already.

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

Executing the installation directly on the client is suitable for individual computers. For a mass agent rollout have a look at opsi-deploy-client-agent. If there is other way to distribute software already, 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, existing opsi products can be installed on these clients.

7.2.1 Usage of the service_setup.cmd on Windows NT6

1. logon to the Windows client with administrative privileges
2. mount the shared directory on the opsi server at `\<opsiserver>\opsi_depot` to a drive letter
3. on the drive letter 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 may have 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 (winst32.exe) elevated in order to do the installation. Therefore you may see here an UAC Message.
5. The script connects to the opsi-webservice in order to create the client on the server side and get the pckey. The first connection is established with the user/password combination provided in the config.ini. If the connection fails, a login window will pop up, where the user can type in a Service-URL (opsi-config-server), with a user and a password. The provided user needs to be member of the group `opsiadmin`. It is also possible to provide a user which has only the right to call the method `host_createOpsiClient`.

Figure 7.5: bootimage/boot-cd: netboot product list

Now you may choose the operating system that you would like to install (or e.g. hwinvent for testing).
Caution
During installation the client reboots without notice!

7.2.2 Usage of the service_setup_NT5.cmd on Windows NT5

1. logon to the Windows client with administrative privileges
2. mount the shared directory on the opsi server at `\<opsiserver>\opsi_depot` to a drive letter
3. on the drive letter from the previous step, start the script `opsi-client-agent\service_setup_NT5.cmd`
4. The script copies the needed files to a temporary local directory and starts from there the opsi-script (`winst32.exe`) in order to do the installation.
5. The script connects to the opsi-webservice in order to create the client on the server side and get the pckey. The first connection is established with the user/password combination provided in the config.ini. If the connection fails, a login window will pop up, where the user can type in a Service-URL (opsi-config-server), with a user and a password. The provided user needs to be member of the group `opsiadmin`.

Warning
During installation, the client reboots without notice!

7.2.3 Usage of the opsi-deploy-client-agent

The `opsi-deploy-client-agent` script installs the opsi-client-agent directly from the opsi-server to the clients. It is now easier to integrate a large number of clients from the server into an opsi environment. Requirements for the clients are:

- an open C$ share
- an open admin$ share
- an administrative account
- a non interfering solution when using the `winexe`

On the server side we require the program `winexe`. This one is part of the package `opsi-windows-support`. 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.

Two copy methods are possible. The first method will use the `mount`-command to locally mount the C$ share of the client on the server. The second variant will use `smbclient` for mounting. After mounting the client the files will be copied.

The script can work with IP addresses, hostnames or FQDNs. It will automatically detect what type of address it is processing.
With the `opsi-deploy-client-agent` script you can batch install a list of clients. Therefore you can give the FQDNs of multiple clients as last argument or you give with the option `-f` the name of a text file which has one FQDN per line.

The script itself is located in `/var/lib/opsi/depot/opsi-client-agent`.

Run this script with `root` privileges.

It may be possible that you have to make the script executable with:

```
```

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

```
bonifax:/home/uib/oertel# cd /var/lib/opsi/depot/opsi-client-agent
bonifax:/var/lib/opsi/depot/opsi-linux-client-agent#./opsi-deploy-client-agent --help
```

```
```

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
- `--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
- `--smbclient`, `-s`
  - 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.
Chapter 8

Rollout existing products

For the rollout of software on clients the `opsi-client-agent` must be installed. This can be rolled out on existing computers. If operating system installation via opsi, `opsi-client-agent` will be installed automatically.

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

8.1 Usage of opsi standard products: `opsi-configed`

One of the opsi standard products is the product `opsi-configed`. This product installs the opsi Management Interface. This Application is a Java Program therefore the Java Runtime Engine is bundled within the product.

Using `opsi-configed`, choose the client by pressing the tab `Clients`, which puts `opsi-configed` in the mode `Configuration of clients`.

If you haven’t done so yet, reload all the data by clicking the reload button at the top left corner of the `opsi-configed` interface (or use the `File` menu).

Switch to the tab `Product configuration`, look for the line with the product-id `opsi-configed`. Go to the column `Requested Action`, and select the action `setup` using a left mouse click. Finally, save the new action with a click on the checkmark button at the top (or by right clicking the mouse and selecting `save`).

Restart the client.

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

8.2 Hard- and Software Inventory with the Products `hwaudit` and `swaudit`

Using `opsi-configed`, choose the client by pressing the tab `Clients`, which puts `opsi-configed` in the mode `Configuration of clients`.

If you haven’t done so yet, reload all the data by clicking the reload button at the top left corner of the `opsi-configed` interface (or use the `File` menu).

Switch to the tab `Product configuration`, look for the lines that audit the software and hardware of the system (`hwaudit` and/or `swaudit`). Go to the column `Requested Action`, and select the action `setup` using a left mouse click. Finally, save the new action with a click on the checkmark button at the top (or by right clicking the mouse and selecting `save`).

Now reboot the client, `opsi-client-agent` executes the `hwaudit` and/or `swaudit` which should automatically start. The client scans the hardware and/or software inventory and sends the results back to the server.

To see the changes at the `opsi-configed` management interface, select reload with the button at the top or with a right mouse click. You may see the update after selecting the tabs `Hardware information` and/or `Software inventory`. 
8.3 Hardware Inventory with the Netboot Product hwinvent

If the product hwinvent is already installed on your opsi server and you have added a client Section 7.1 which is configured to boot over the network, you can do something else useful: Hardware inventory without an existing operating system installed.

Using opsi-configed, choose the client by pressing the tab Clients, which puts opsi-configed in the mode Configuration of clients.

If you haven’t done so yet, reload all the data by clicking the reload button at the top left corner of the opsi-configed interface (or use the File menu).

Switch to the tab Netboot products, look for the line that has hwinvent. Go to the column Requested Action, and select the action setup. Finally, save the new action with a click on the checkmark button at the top (or by right clicking the mouse and selecting save).

Then reboot the client. It should now use a PXE Linux image over the network, to scan the hardware of the PC and then reboot it (if the computer was not already set up, the message that no operating system is installed on the disk will be displayed).

To see the changes at the opsi-configed management interface, select reload with the button at the top or with the mouse. You may see the update after selecting the tab Hardware information.

---

**Note**

In case the screen only turns black after booting the bootimage or a network error occurs, the startparameters of the bootimage have to be adjusted.

This can be achieved using the opsi-configed in the tab Hostparameter by editing the entry opsi-linux-bootimage.append.
Chapter 9

Installation of a new Windows PC through opsi (OS Installations)

The following describes how a computer with no operating system can get a Windows OS installed via 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 loading boot systems via the network. The network boot has to be activated in the BIOS menu or moved to the first position of the boot options.

Virtual hardware is usually well supported by the Windows standard drivers if you later 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 maps an empty machine and can run in VMware Workstation Player.

For the following chapter you should create the corresponding client in opsi Section 7.1. Take a look here through opsi-configed.

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

9.1 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.

If you want to make use of fully automated OS installations of Windows 7/8.1/10, you have to complete these packages as described below.

9.2 NT6 family: as of Win7 / 2008R2

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 do it all yourself manually following the steps described. Generally speaking, the Windows-Version of the PE does not matter with regard to the Windows OS version being installed. Still, working drivers should be present for at least disk and network devices. Microsoft recommends 32-Bit PE for x86 installations, and 64-Bit PE for x64 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."

Regardless of how you want to create your PE, you’ll need an installed 'Assessment and Deployment Kit' (ADK, Win8.1 bzw 10), or its predecessor 'Windows Automated Installation Kit' (Windows AIK; since Windows 7):

- **Windows 10 / 8.1 ADK**

It’s sufficient to install *Windows PE add-on for the ADK* and the dependencies automatically selected. Stick with the suggested install path in *Program Files (x86)*.

- **link: WAIK Windows 7**

This site provides you with an ISO file, which may then be burnt to a CD or mounted. The content of this CD must be installed in an OS mentioned in the previous system requirements.

### 9.2.1 Creating a PE

The easiest available method for tweaking your PE requires a computer that has opsi-client-agent installed, as well as a Windows ADK (Win8.1, Win10). Doing all steps manually is being described below at Section 9.2.1.2.

#### 9.2.1.1 Automated PE creation using opsi

- Using the opsi-configed set the localboot-product opsi-winpe to once for the client you intend to use, if desired adjust the produkt properties to x86 instead of x64 at the lower right side, and save (right click > save)
- in case the opsi-product opsi-winpe is missing, install it onto your opsi server via CLI opsi-package-updater -v install opsi-winpe
- launch an installation event for the client (right click > on-demand, or reboot)
- after successful run of this action move or copy the contents of the now existing folder on your client C:\winpe_<ARCH>\media\ into the folder (already existing) within the OS folder you want to use at \\opsiserver\opsi_depot_rw\<OS>\winpe\
- finally run the following command on the CLI of your opsi server. Done.

```bash
opsi-set-rights
```

#### 9.2.1.2 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 set to 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 with required environment variables set.

- Copy the WinPE

```bash
copy.exe.cmd <ARCH> C:\winpe
```

- Mount the Image

```bash
dism /Mount-Wim /WinFile:C:\winpe\media\sources\boot.wim /index:1 /MountDir:c:\winpe\mount
```
replace startnet.cmd

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

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

Unmount the Image

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

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

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

### 9.2.1.3 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 set to either `x86`, `amd64` or `ia64`.

Start a terminal as Administrator with elevated rights (Start ⇒ Programs ⇒ Accessories ⇒ right click on 'Command Prompt' ⇒ 'Run as' ⇒ Administrator)

Copy the WinPE

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

Mount Image:

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

replace startnet.cmd

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

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

Unmount the Image

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

Move the WinPE now. From the target dir more files will be moved to the server.

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

Copy the contents of C:\winpe\ISO to /var/lib/opsi/depot/win7/winpe (or /var/lib/opsi/depot/win2008/winpe).
Adjust the file access rights by entering e.g.:

```bash
opsi-set-rights /var/lib/opsi/depot/<productid>/winpe
```
9.2.2 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 for use in 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 chapter "Creating a PE".

Note
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 7zip 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.

- Use dism to scan the image, in order to determine the required index number. Normally a PE-image is a one-image-file, so you can use the index 1, but it is better to check at first. Start a terminal as administrator (Start ⇒ Programs ⇒ Accessories ⇒ right click on "Command Prompt" ⇒ "Run as" ⇒ (Administrator) and run the following command:

```bash
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.

- The next command mounts the image for modification:

```bash
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:

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

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

Note
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. With the option `/ForceUnsigned` it is possible to integrate unsigned drivers to the image.

- For the changes to be committed, the images must be unmounted:

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

- Copy the directory `C:\winpe\ISO` with the target name `winpe` to `/var/lib/opsi/depot/win7/` (or `/var/lib/opsi/depot/win2008`).

Adjust the file access rights by entering(e.g.):

```bash
opsi-set-rights /var/lib/opsi/depot/win7/winpe
```
9.2.3 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`. If you would like to make any modifications to this file, then do it in this directory and not in the `opsi` directory.

The file `unattend.xml` that comes with the opsi package, contains links to the Netboot-Productproperties which are responsible for activating the Administrator account with the password `nt123`.

Documentation of the file `unattend.xml` can be found (after the installing WAIK) in the directory `c:\Program Files\Windows\Waik\docs\chms`.

9.2.4 Driver Integration


Please keep in mind that only signed drivers are accepted. Therefore, if you want to use driver packs like the driver packs from driverpacks.net, be sure to use only the Windows 7/8.1/10 versions.

9.2.5 Providing the Installation Files

Copy the complete installation DVD to

`/var/lib/opsi/depot/<productid>/installfiles` Adjust the file access rights by entering:

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

9.2.6 Installation Log files

- `c:\Windows\Panther\setupact.log`:
  Logs until the end of setup phase 4 (running under WinPE)

- `c:\Windows\Panther\setupact.err`:
  Error log including the end of setup phase 4 (running under WinPE)

- `c:\Windows\Panther\UnattendGC\setupact.log`:
  Logs a specialize phase

- `c:\Windows\Panther\UnattendGC\setupact.err`:
  Error log for a specialize phase

- `c:\Windows\System32\Winevt\Logs\*`

- `c:\Windows\ntbtlog.txt` (only when the startup protocol is activated)

9.3 Windows Product Key

If you are using the opsi license management module, then you may administrate your Windows product keys using the license management software. Information on how to do this can be found in the opsi manual.

If you don’t want to use the license management module, then the product key can simply be made up using the product properties.

While creating a client, you can use the opsi management interface to enter the product key:

- choose a client
• change to the tab *netboot products*
• select the product (e.g. win7-x64)
• change to the product property *productkey* (on the right lower corner of the opsi management interface)
• type in your key
• leave the input field and save the changes

Another possibility is to use the command line. While working with an opsi server, you can read and/or change the server defaults. To read the server default use (you may need to modify the productId and you must change `<opsiserver.domain.local>` with the fqdn from your opsiserver. Be sure that you enter the commands in one line):

```bash
opsi-admin -d method productPropertyState_getObjects [] '{"productId":"win10-x64","objectId":"opsiserver.domain.local"}';
```

The easiest way to modify the defaults, is to modify the file, and then update the objects with the modified file. The first step would be to view the contents of an actual configuration file (you may need to modify the productId and you must change `<opsiserver.domain.local>` with the fqdn from your opsiserver. Be sure that you enter the commands in one line):

```bash
opsi-admin -d method productPropertyState_getObjects [] '{"productId":"win10-x64","objectId":"opsiserver.domain.local"}'; > /tmp/property_config.json
```

The second step would be to modify the file `/tmp/property_config.json`, and change the entries and values. Finally, you must update the objects using this modified file (enter this command in one line):

```bash
opsi-admin -d method productPropertyState_updateObjects < /tmp/property_config.json
```

You can check that the modifications were successful using the following command (you may need to modify the productId and you must change `<opsiserver.domain.local>` with the fqdn from your opsiserver. Be sure that you enter the commands in one line):

```bash
opsi-admin -d method productPropertyState_getObjects [] '{"productId":"win10-x64","objectId":"opsiserver.domain.local"}';
```

### 9.4 Start the Windows Installation

To start a Windows installation:

• choose a client
• change to the tab *netboot products*
• select the product (e.g. win7-x64)
• set the *action request* to *setup*
• save the changes by clicking the red check mark (which then changes to green)

Now the client should load the opsi-linux-bootimage via the network and start it up. Before the Windows installation starts, you might have to confirm.

---

**Caution**

This refers to clients with a hard drive larger than 2 terabyte. On a non UEFI-system the possible largest partition size is 2 terabyte. When you have a larger partition scheme the installation will fail. This is a technical restriction. You have to configure the system partition with a maximal size of 2 terabyte and therefore configure two partitions. This can be done with the product-properties. On the other hand the UEFI-module bypasses this restriction by using another partition table.
9.5 Structure of the Unattended Installation Products

This chapter describes the Windows netboot products.

9.5.1 Directory Tree Overview

```
<productid>-
| |-i386/ NT5 only: Installations files
| |-installfiles/ NT6 only: Installations files
| |-winpe/ NT6 only
| |-oem$/ NT5 only: $oem$ according to MS
| | |-postinst.d/ scripts after OS-install by opsi.org
| | |-unattend.(txt/xml).template Template by opsi.org
| |-custom/ scripts and templates by customer
| | |-oem$/ NT5 only: $oem$ according to M$ customer
| | |-postinst.d/ scripts after OS-install by customer
| | |-unattend.(txt/xml) unattend.txt by customer
| |-drivers/ drivers directory
| | |-drivers/ drivers directory
| | |-pciids/ symbolic links to drivers
| | |-vendors/ symbolic links to drivers
| | |-classes/ symbolic links to drivers
| | |-usbids/ symbolic links to drivers
| | |-hdaudioids/ symbolic links to drivers
| | |-pci.ids PCI-IDs DB
| | |-usb.ids USB-IDs DB
| |-setup.py installation script
| |-<productid>_<version>.control meta data (only for info)
| |-<productid>.files file list (created automatically)
| |-create_driver_links.py driver management script
| |-show_drivers.py driver management script
| |-download_driver_pack.py driver management script
| |-extract_driver_pack.py driver management script
```

9.5.2 File Descriptions

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

- **<productid>_<version>.control**
  Contains the meta data of the product as prepared from the package maintainer. These files are here for information purposes only. There will be no effect after changing these files.

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

- **create_driver_links.py**
  show_drivers.py
  download_driver_pack.py
  extract_driver_pack.py
  These are scripts for the simplified driver integration, which is described in its own chapter ("Simplified driver integration for the automatic OS installation").

9.5.3 Directory installfiles / winpe

- **installfiles**
  This directory contains all files from the windows installation DVD (NT6 = Windows 7 and above).

- **winpe**
  This directory contains a bootable winpe image among other files.
9.5.4 Directories opsi and custom

Both directories contain scripts and configuration files for the OS installation. During the installation process, the directories work together in such a way that they give the priority usage to the files in the custom directories.

The opsi directory contains files and templates that are maintained by opsi.org, and maybe replaced during the next update. So it's not a good idea to make specific (or customized) changes to these files in this location. Please use the custom directory for this purpose, because that directory is not subject to any changes by opsi.org.

The subdirectory `postinst.d` contains scripts which are executed after the OS installation is completed by the `postinst.cmd` program. These scripts are needed to install the opsi-client-agent, among other software. The scripts will be executed in alphabetic order. To make it easier to see the order in which the scripts will be executed, the name always starts with a 2 digit number (e.g. `10_dhcp.cmd`). If you want to make extensions, then please do so in the custom/postinst.d directory and start numbers between the 10, 20, 30,... (e.g. `13_myscript.cmd`). The starting numbers 10, 20, 30,... are reserved for use by opsi org / uib gmbh. The script `99_cleanup.cmd` is the last one and initiates a reboot.

9.5.5 Directory drivers

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

9.6 Simplified Driver Integration during the unattended Windows Installation

If a pool of computers needs drivers that are not part of the Windows default installation, it’s best to integrate their drivers during installation time.

Opsi supports the automatic integration of drivers into the installation, and therefore simplifies driver deployment. In order to integrate drivers with opsi, the drivers simply need to be placed into the correct directory. When the installation script is called it parses through these directories and creates a catalog. The boot image automatically uses this catalog to embed the correct drivers. Opsi supports the automatic installation of standard drivers, USB drivers, HD audio drivers, and disk controller drivers (text-mode drivers).

In order for a driver to be immediately installed with the Windows installation, you must place the drivers on the server in a specific format. The drivers must be placed in the drivers directory, with the format `*.inf`, where the file name describes the driver for the Windows setup program. Drivers packed like `setup.exe` or `*.zip` are not used here. If you have a computer that already has the drivers installed, then you can extract the appropriate drivers using the program double driver (http://www.boozet.org/dd.htm).

There are many levels of driver integration:

- **General driver packages**
- **Preferred drivers that belong to your hardware, but are not assigned to specific computers**
- **Drivers that will be manually assigned to computers**
- **Drivers that will be automatically assigned to the computers using the fields <vendor>/<model>**

Below is a detailed discussion about how to include each of these drivers

9.6.1 General Driver Packages

When the hardware configuration across your pc-pool is very heterogeneous, then it may be reasonable to work with general driver packages.

General drivers can be placed under `./drivers/drivers`. You can find such general driver packages here http://driverpacks.net/ .

Download the appropriate driver package to a temporary directory, and then unpack the driver package using the opsi script `extract_driver_pack.py` as such:
This will unpack and store the drivers in the directory `.drivers/drivers/`. It may be the case that the drivers found by opsi in this location do not necessarily work with your hardware. For the drivers which are found in `.drivers/drivers/`, the driver will be matched to the corresponding hardware using the PCI IDs (i.e. USB- or HD_Audio-ID) in the description file, and then integrated into the Windows setup as needed.

### 9.6.2 Preferred Drivers

In case you have to support special hardware, where additional drivers are provided by the manufacturer, then you can use the following procedure to include them in the installation.

Place the additional drivers in their own directory under:

`.drivers/drivers/preferred`

(the naming and depth of the directory structure is not important). Drivers that are found in the directory `.drivers/drivers/preferred` will be integrated into the Windows setup, assuming that opsi finds a suitable match for the hardware based on the PCI IDs (i.e. USB or HD_Audio-ID) in the description file of the driver. Problems can occur when the same PCI ID can be found in the description file of multiple drivers in preferred. In this case, a direct mapping of the drivers to the client is needed.

### 9.6.3 Drivers that will be manually assigned to clients

When installing additional drivers not based on the PCI-IDs or USB-IDs, they should be installed under the directory `.drivers/drivers/additional` (where name and depth of the directory structure is not important). You can map one or more drivers to a client using the Product-Property `additional_drivers` by supplying one or multiple paths of driver directories under `.drivers/drivers/additional` as value. The directories specified by `additional_drivers` are searched recursively and all located drivers will be installed. The usage of symbolic links is also permitted, this can be used to create a specific directory based on the client type (i.e. dell-optiplex-815).

When a driver is found in one of the driver directories that is specified by `additional_drivers` and matches the PCI identifier, then other drivers in `.drivers/preferred` or `.drivers/` will not be used (`additional_drivers` can be thought of as super-preferred). Therefore the drivers under `additional_drivers` are installed even when the corresponding devices are not found.

### 9.6.4 Drivers which will be Automatically Assigned to the Clients using the Fields `<vendor>/<model>`

The previously described mechanism that directly maps drivers to devices is automated since the 4.0.2 Release 2 of opsi. The opsi-linux-bootimage will search the directory `.drivers/drivers/additional/byAudit` for a directory name that matches the field `Vendor` that was given in the Hardware Inventory. This `Vendor` directory will be searched for a `Model` directory that corresponds to what is seen in Hardware Inventory. If this directory is found, it will be treated as if it was manually assigned to 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 one can use opsi-configed for uploading the drivers Automatic driver upload (see opsi-manual ‘Automatic driver upload’)

The opsi-linux-bootimage looks for drivers successively in the following directories

- `<vendor>/<model> (<sku>)`
- `<vendor>/<model>`
- `<vendor>/<motherboard-model>`

Some vendors use model identifiers containing special characters. Opsi replaces `<,>?,\;,:\|/ and * with _ therefore a model identifier like "5000/6000/7000" will match a directory identifier like "5000_6000_7000".
9.6.5 Structure of the Driver Directory and Driver Files

/\var/
   /\-lib/
      /\-opsi/depot/
         /\-<productid>/
            /\-drivers
               /\-classes/ (Links to driver device classes)
               /\-hdaudioids/ (Links to HD-Audio drivers)
               /\-pci.ids/ (PCI database)
               /\-pciids/ (Links to PCI-ID drivers)
               /\-usb.ids/ (USB database)
               /\-usbids/ (Links to USB-ID drivers)
               /\-vendors/ (Links to manufacturer drivers)
               /\-additional/ (place for general driver packages)
                  /\-byAudit/ Model-specific drivers that will be assigned by the Hardware Inventory
                  /\-<vendor> will be assigned by the Hardware Inventory
                  /\-<model> the Hardware Inventory
               /\-buildin/ (data for the i386 version)
               /\-preferred/ (certified drivers)
               /\-exclude/ (excluded drivers)
               /\-mydriverpacks/ (example driver packages)

9.6.6 Processing of the Different Levels of Driver Integration

The top priority is given to drivers which are found using the property additional_drivers or using the inventory data in ./drivers/drivers/additional/byAudit. When installing a client these drivers will be prefered. If a device is not matched by a driver the following methods are used in order to find a matching driver.

For devices where no matching driver was found using additional_drivers or byAudit, opsi will search for and integrate an appropriate driver based on the PCI-ID (respectively 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 will search for the drivers in c:\drv\, this is specified in the unattended file.

9.6.7 Driver Addition and Checking

After any changes in the directory ./drivers/drivers have been made, call the following command from the Netboot Product root directory in order to set the permissions:

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

Then call the script ./create_driver_links.py. The script searches through the directories under ./drivers/drivers and generates a list of links using PCI-IDs, USB-IDs, HD-Audio-IDs. This list is used to match drivers and devices. The script will prioritize the drivers in the preferred directories.

The script setup.py examines the hardware of the installed computers and identifies the necessary drivers. These will be copied to the disk and the file unattended.xml will be patched. The script create_driver_links.py examines the NT5 products one at a time in the i386 tree and extracts the .inf files of the necessary drivers into windows_builtin. If you make a change to the i386 directory tree (i.e. after installing a service pack), then delete that directory and run create_driver_links.py again. The recognized drivers for NT6 products are found in WinPE at windows_builtin.

With the following command, matching drivers for a existing hardware inventory for a client can be shown:

```
./show_drivers.py <clientname>
```
Furthermore devices without a matching driver will be shown.

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

It is possible that manufacturers include different drivers for different operating systems (i.e. Windows 7/8.1/10) or different configurations (i.e. SATA vs. SATA RAID) in one directory. The `create_driver_links.py` cannot make this distinction. If you think the wrong driver will be used, you can move the driver to the `drivers/exclude` directory and then call `create_driver_links.py` again. Drivers in the directory `drivers/exclude` are not used during the integration.

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

```
./show_drivers.py pcdummy

PCI-Devices

[(Standard-Schnittstellen), PCI Standard-PCI-zu-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-Zweikanal-PCI-IDE-Controller]
/var/lib/opsi/depot/<productid>/drivers/drivers/D/W/1129

[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-konformer Hostcontroller-Hersteller, OHCI-konformer 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_eb7xx

[(Standard-USB-Hostcontroller), Standard OpenRCD 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-Verbundgeräte]
/var/lib/opsi/depot/<productid>/drivers/drivers/preferred/brother_844x_pGerb

[Microsoft, USB-Druckerunterstützung]

/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`:

```
./show_drivers.py e5800

Manually selected drivers (additional)

[hp_e5800]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPCI13.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPCI11.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPCI12.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/autorum.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/libxHDMI/IntcDaud.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/HDMI/IntcHdmi.inf]

[href/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp54284/Realtek 64bit/hp64win7.inf]

PCI-Devices

[8086:27C8] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27C8

[8086:27DA] Intel : Intel(R) N10/ICH7 Family SMBus Controller - 27DA

[8086:27C9] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27C9

[8086:27DF] Intel : Intel(R) ICH7 Family Ultra ATA Storage Controllers - 27DF
```

Example output of a `show_drivers.py` call for a client with `additional_drivers`.
Example with `byAudit`:

```bash
./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)/atiilhag.inf]
```

PCI-Devices

```text
[1002:5B70] ATI Technologies Inc. : Radeon X300/X550/X1050 Series Secondary (Microsoft Corporation - WDDM)
```

USB-Devices

```text
[0461:0010] (Standardstystemgeräte) : USB-Eingabegerät
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
[0461:4D20] (Standardstystemgeräte) : USB-Eingabegerät
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
[058F:6366] (Kompatibles USB-Speichergerät) : USB-Massenspeichergerät
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/058F' not found
[0461:0010] (Standard-USB-Hostcontroller) : USB-Verbinder
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
```

HD-Audio-Devices

```text
[10EC:0662] Realtek High Definition Audio
Manually selected [hp_e5800] /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64
```

Example with `byAudit`:

```bash
./show_drivers.py pctry5detlef
Manually selected drivers (additional)
```

PCI-Devices

```text
[1022:1100] AMD : AMD HyperTransport(tm)-Konfiguration
Using build-in windows driver
```
TIPS

- Directory names NDIS1 contain Vista-Drivers; NDIS2 contain Win7-Drivers
- 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 were to be placed in additional_drivers (or byAudit), then other drivers in the preferred subdirectory will be ignored.
- SATA drivers and SATA-RAID drivers refer to the same PCI ID. A SATA-RAID driver will not function with a single-disk system.
- Check the output of ./show_drivers.py carefully!
Chapter 10

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-winst 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.

10.1 A Brief Tutorial: How to write a opsi-winst Script

10.1.1 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-winst reference card and opsi-winst manual


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

10.1.2 Methods of Non-Interactive Installation

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-winst:**
The actions executed by the setup-routine when installing manually are recorded and the opsi-winst is used to reproduce them.

**Note**
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.

### 10.1.3 Structure of a opsi-script / opsi-winst Script

An example of a simple opsi-winst script:

```plaintext
[Actions]
WinBatch_7z_silent_install

[WinBatch_7z_silent_install]
"%ScriptPath%/7z.exe" /S
```

An opsi-winst script contains a **primary** and **secondary** sections. The section headers are in square brackets, similar to what you may have seen in ini-files. The primary section is noted by the identifier `[Actions]`, and the secondary section is noted by the identifier `[WinBatch_...`.

The core tasks, like starting programs or copying files, are specified in the secondary sections, not in the primary sections. These secondary sections are topic specific, and have a specific syntax that relates to their specific topic.

The name of a secondary section starts with a reserved word identifying the type of the secondary section followed by a free identifier.

In the above example, the primary section `[Actions]` calls a secondary section `[WinBatch_7z_silent_install]`. This secondary section has the type `WinBatch`. The content of the secondary sections, of type `WinBatch`, are executed by the Windows API. In this case, the binary `7z.exe` will be started with the parameter `/S`.

### 10.1.4 Primary Sections

**Actions**
The section `[Actions]` is the main program.

Any part of the code that is called more then one time can be placed in sub sections.

**Sub-sections**
Primary sections which may be called multiple times or have their code in external files.

The primary sections are the main program which control the program flow. There you will find:

- Variables: strings and string lists
- if else endif statements
- for loops that traverse string lists
• Functions

![Diagram of setup and uninstall processes](image)

**Figure 10.1:** double code for deinstallation

![Diagram showing avoidance of double code](image)

**Figure 10.2:** avoid double code by using sub sections

### 10.1.5 Important Kinds of Secondary Sections

#### Files

File operations include

- copying (regarding the internal version information, recursive, …)
- deleting files or directories
- creating directories

#### WinBatch

It’s used for calling programs using the Windows API. For example, WinBatch calls the setup programs in the silent mode.
DosBatch/DosInAnIcon
The content of these sections are interpreted by the cmd.exe like normal batch files.
A variant of DosBatch is DosInAnIcon which is run in a minimized window.

ExecWith
A program is given as a parameter, and then that program interprets the content of this section (e.g. AutoIt).

Registry
The Registry sections are used for registry manipulations.

Linkfolder
Link folder sections are used for the manipulation of start menus and desktop icons.

10.1.6 Global Constants
Global constants are placeholders which can be used in primary and secondary sections. These placeholders are replaced by their values at runtime.

Examples:

%ProgramFiles32Dir%
c:\program files

%Systemroot%
c:\windows

%System%
c:\windows\system32

%Systemdrive%
c:\

%Scriptpath%
<path to the running script>

10.1.7 Second Example: tightvnc

The following example shows a simple script that is used for a tightvnc installation. This script should contain only the winbatch call for the silent installation. If you call the sub-section silent installation more the one time, a confirmation window appears (which is a bug in the installer). This confirmation window will be closed by a autoit script if it appears.

tightvnc.ins:

[Actions]
Message "Install 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")
10.1.8 Elementary Commands for Primary Sections

10.1.8.1 String Variable

Declaration of a variable
DefVar <variable name>

Setting a value
Set <variable name> = <value>

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

---

Important
The use of string variables is different in primary versus secondary sections. In the primary section, the string variables are handled as independent objects. String variables can only be declared and set to values in primary sections. Therefore you have to use a operator (+) to concatenate variables and strings in a string expression. Example: "Installing "+$ProductId$+" ..."
In secondary sections string variables are used as a placeholder for their values. Example: "Installing $ProductId$ ...
You should keep this in mind if you cut and paste string expressions between primary and secondary sections.
The advantage of handling string variables in this format is that it is possible to use these variables in secondary sections that are interpreted by other programs (DosBatch / Execwith).

---

10.1.8.2 Message / showbitmap

Displaying text during runtime:
Message <string>

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

Displaying a picture during installation:
ShowBitmap [<file name>] [<sub title>]

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

---

10.1.8.3 if [else] endif

Syntax:

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

HasMinimumSpace
   Check for free space on the hard disk

FileExists
   Check for the existence of a file or directory

10.1.8.5 Error, Logging and Comments

comment char ;
   Lines starting with the ; char are simply ignored.

comment
   writes a comment to the log file

LogError
   writes error messages to the log file

isFatalError
   aborts the script, and return the installation state failed to the server.

10.1.8.6 Requirements

requiredWinstVersion
   Minimum required version of opsi-winst

10.1.9 Third example: The Generic Template opsi-template

This third template should be used as a rough guide whenever you create your own opsi product. Do not cut-and-paste from this manual, but instead look at http://download.uib.de for a new version of the opsi-template product package. Using the opsi-package-manager command you may install opsi-template (-i) or extract (-x) at your server and then grab the scripts.

setup32.opsiscript: installation script

; Copyright (c)uib gmbh (www.uib.de)
; This sourcecode is owned by uib
; and published under the Terms of the General Public License.
; credits: http://www.opsi.org/en/credits/

[Actions]
requiredWinstVersion >= "4.11.4.6"
ScriptErrorMessages=off

DefVar $MsiId$
DefVar $UninstallProgram$
DefVar $LogDir$
DefVar $ProductId$
DefVar $MinimumSpace$
DefVar $InstallDir$
DefVar $ExitCode$
DefVar $LicenseRequired$
DefVar $LicenseKey$
DefVar $LicensePool$
DefVar $displayName32$
DefVar $displayName64$
DefStringlist $msilist$

Set $LogDir$ = "%opsiLogDir%"

; - Please edit the following values -

; $ProductId$ should be the name of the product in opsi
; therefore please: only lower letters, no umlauts,
; no white space use '-' as a separator
Set $ProductId$ = "opsi-template"
Set $MinimumSpace$ = "1 MB"
Set $InstallDir$ = "%ProgramFiles32Dir%\<path to the product>"
Set $LicenseRequired$ = "false"
Set $LicensePool$ = "p_" + $ProductId$

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
else
    comment "Show product picture"
    ShowBitmap "%ScriptPath%\" + $ProductId$ + ".png" $ProductId$
    if FileExists("%ScriptPath%\delsub32.opsiscript")
        comment "Start uninstall sub section"
        Sub "%ScriptPath%\delsub32.opsiscript"
    endif
    Message "Installing " + $ProductId$ + " ...
    if $LicenseRequired$ = "true"
        comment "Licensing required, reserve license and get license key"
        Sub_get_licensekey
    endif
    comment "Start setup program"
    ChangeDirectory "%SCRIPTPATH%"
    Winbatch_install
    Sub_check_exitcode
    comment "Copy files"
    Files_install /32Bit
    comment "Patch Registry"
    Registry_install /32Bit
    comment "Create shortcuts"
    LinkFolder_install
endif
### Getting Started

opsi Version 4.1

[Winbatch_install]

; Choose one of the following examples as basis for your installation
; 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* "$LogDir$\$ProductId$.install_log.txt" /qb-! ALLUSERS=1
; REBOOT=ReallySuppress
;
:== InstallShield + MSI
=================================================================================

; Attention: The path to the log file should not contain any whitespaces
; "%ScriptPath%\setup.exe" /s /v" /i $LogDir$\$Product%.install_log /qb-! ALLUSERS=1
; REBOOT=ReallySuppress
; "%ScriptPath%\setup.exe" /s /v" /qb-! ALLUSERS=1 REBOOT=ReallySuppress"
;
:== InstallShield
=================================================================================

; Create setup.iss answer file by running: setup.exe /r /f1"c:\setup.iss"
; You may use an answer file by the parameter /f1"c:\setup.iss"
; "%ScriptPath%\setup.exe" /s /sms /f2$LogDir$\$Product%.install_log /qb-! ALLUSERS=1
;
:== Inno Setup
=================================================================================

; http://unattended.sourceforge.net/InnoSetup_SwitchesExitCodes.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 a registry key:
;
; openkey [HKEY_LOCAL_MACHINE\Software\$Product%]
; set "name1" = "some string value"
; set "name2" = REG_DWORD:0001
; set "name3" = REG_BINARY:00 af 99 cd

[LinkFolder_install]

; Example of deleting a folder from AllUsers startmenu:
;
; set_basefolder common_programs
; delete_subfolder $ProductId$
;
; Example of creating an 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
; set_subfolder ""
;
; set_link
; name: $ProductId$
; target: <path to the program>
; parameters: <some_param>
; working_dir: $InstallDir$
; icon_file: <path to icon file>
; icon_index: 2
; end_link

[Sub_get_licensekey]
if opsiLicenseManagementEnabled
    comment "License management is enabled and will be used"

    comment "Trying to get a license key"
    Set $LicenseKey$ = demandLicenseKey ($LicensePool$)
    ; If there is an assignment of exactly one license pool to the product the following call
    ; is possible:
    ; Set $LicenseKey$ = demandLicenseKey ("", $ProductId$)
    ;
    ; If there is an assignment of a license pool to a windows software id, it is possible to
    ; use:
    ; DefVar $WindowsSoftwareId$
    ; $WindowsSoftwareId$ = "...
    ; Set $LicenseKey$ = demandLicenseKey ("", ", $WindowsSoftwareId$)

    DefVar $ServiceErrorClass$
set $ServiceErrorClass$ = getLastServiceErrorClass
comment "Error class: " + $ServiceErrorClass$
if $ServiceErrorClass$ = "None"
    comment "Everything fine, we got the license key " + $LicenseKey$ + ""
else
    if $ServiceErrorClass$ = "LicenseConfigurationError"
        LogError "Fatal: license configuration must be corrected"
        LogError getLastServiceErrorMessage
        isFatalError
    else
        if $ServiceErrorClass$ = "LicenseMissingError"
            LogError "Fatal: required license is not supplied"
            isFatalError
    endif
endif
endif
endif
else
    LogError "Fatal: license required, but license management not enabled"
    isFatalError
endif

[Sub_check_exitcode]
comment "Test for installation success via exit code"
set $ExitCode$ = getLastExitCode
; informations to exit codes see
if ($ExitCode$ = "0")
    comment "Looks good: setup program gives exitcode zero"
else
    comment "Setup program gives a exitcode unequal zero: " + $ExitCode$
    if ($ExitCode$ = "1605")
        comment "ERROR_UNKNOWN_PRODUCT 1605 This action is only valid for products
        that are currently installed."
        comment "Uninstall of a not installed product failed - no problem"
    else
        if ($ExitCode$ = "1641")
            comment "looks good: setup program gives exitcode 1641"
            comment "ERROR_SUCCESS_REBOOT_INITIATED 1641 The installer has
            initiated a restart. This message is indicative of a success."
        else
            if ($ExitCode$ = "3010")
                comment "looks good: setup program gives exitcode 3010"
                comment "ERROR_SUCCESS_REBOOT_REQUIRED 3010 A restart is
                required to complete the install. This message is indicative of a success."
            else
                logError "Fatal: Setup program gives an unknown exitcode unequal
                zero: " + $ExitCode$
                isFatalError
            endif
        endif
    endif
endif
endif

delsub32.opsiscript: external deinstallation sub section

; Copyright (c) uib gmbh (www.uib.de)
; This sourcede is owned by uib gmbh
; and published under the Terms of the General Public License.
; credits: http://www.opsi.org/en/credits/
Set $MsiId$ = '{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}'
Set $UninstallProgram$ = "$InstallDir$\uninstall.exe"
Message "Uninstalling " + $ProductId$ + " ...
if FileExists($UninstallProgram$)
    comment "Uninstall program found, starting uninstall"
    Winbatch_uninstall
    sub_check_exitcode
endif

if not (GetRegistryStringValue32("[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall]" + $MsiId$ + "] DisplayName") = "")
    comment "MSI id " + $MsiId$ + " found in registry, starting msiexec to uninstall"
    Winbatch_uninstall_msi
    sub_check_exitcode
endif

comment "Delete files"
Files_uninstall /32Bit

comment "Cleanup registry"
Registry_uninstall /32Bit

comment "Delete program shortcuts"
LinkFolder_uninstall

[Winbatch_uninstall]
; Choose one of the following examples as basis for program uninstall
;
; === Nullsoft Scriptable Install System
; ---------------------------------------------------------------
; maybe better called as
; Winbatch_uninstall /WaitforProcessending "Au_.exe" /Timeoutseconds 10
; "$UninstallProgram$" /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]
uninstall32.opsiscript: deinstallation script

; Copyright (c) uib gmbh (www.uib.de)
; This sourcecode is owned by uib
; and published under the Terms of the General Public License.
; credits: http://www.opsi.org/en/credits/

[Actions]
requiredWinstVersion >= "4.11.4.6"
ScriptErrorMessages=off

DefVar $MsiId$
DefVar $UninstallProgram$
DefVar $LogDir$
DefVar $ExitCode$
DefVar $ProductId$
DefVar $InstallDir$
DefVar $LicenseRequired$
DefVar $LicensePool$

Set $LogDir$ = "%opsiLogDir%"

; -----------------------------------------
; - Please edit the following values -
; -----------------------------------------
Set $ProductId$ = "opsi-template"
Set $InstallDir$ = "%ProgramFiles32Dir%\<path to the product>"
Set $LicenseRequired$ = "false"
Set $LicensePool$ = "p_" + $ProductId$

; -----------------------------------------

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

Message "Uninstalling " + $ProductId$ + " ..."

if FileExists("%ScriptPath%\delsub32.opsiscript")
    comment "Start uninstall sub section"
    Sub "%ScriptPath%\delsub32.opsiscript"
endif

if $LicenseRequired$ = "true"
    comment "Licensing required, free license used"
    Sub_free_license
endif

[Sub_free_license]
comment "License management is enabled and will be used"

comment "Trying to free license used for the product"
DefVar $result$
Set $result$ = FreeLicense($LicensePool$)
; If there is an assignment of a license pool to the product, it is possible to use
; Set $result$ = FreeLicense("", $ProductId$)
10.1.10  Interactive Creation and Testing of a opsi-winst Script

It is possible to interactively adapt and test your own opsi-winst script using winst32.exe.

Start by creating a directory where you will build and test your script (e.g. c:\test), and then copy the template scripts from the opsi-template (setup.ins, delsub.ins und uninstall.ins) to this directory.

Start the opsi-winst (winst32.exe) program via a double mouse click. (On Windows 7 Clients, you must right-click on the mouse button and select "run as Administrator"). If the opsi-client-agent is installed on your computer you will find the opsi-winst at the directory C:\program files\opsi.org\opsi-client-agent\opsi-winst. If the {opsi-client} agent is not installed you will find the {opsi-winst} at the share \<opsiserver\opsi_depot_rw' in the directory `install\opsi-winst\files.

After starting opsi-winst, you will see the following window:
Figure 10.3: opsi-winst Started in Interactive Mode

- **Select Script** is used to choose the script that you want to execute.
- **Start** will start the execution of the selected script.
- **View Log** is used to read the log file from the script that was run most recently.

Select the `setup.ins` script and run it.
• Look at the log file to see how opsi-winst interpreted the script.

• After figuring out which setup.exe that you will use to install software, copy setup.exe to the directory where the scripts are located (e.g. c:\test).

• Open the setup.ins script with a editor. You may use any text editor you like. We suggest the jEdit with syntax highlighting for opsi-winst which is part of the essential opsi-products.
Figure 10.5: jEdit with a opsi script

- You may now change the script using the editor. Save the changes (keep the editor open).
- Now switch to the opsi-winst and start the script again. (You don’t have to reselect the script. Just press the start button).
- Just have a look at the log again and see how the program flow changed according to your script changes.
You can interactively develop a script until it fits your needs by performing these steps in this order:

- Change the script and save
- run the script
- review the log

The next chapter contains some hints about handling any problems that may arise while building a opsi-winst script. Section 10.2.1.1 describes how to create an opsi-product from your scripts, and how to install the products on the opsi-server.

### 10.1.11 Suggestions on How to Solve Problems with opsi-winst Scripts

#### 10.1.11.1 Search for Unattend or Silent Switches

For an unattended or silent setup, the original setup will be switched to an unattended non-interactive mode using the proper command line arguments.

The problem is to find the correct arguments.

**Look on the internet:** Before you start integrating a new package, you’d better first have a look online to see if somebody has already done that job for you:

Ready to run opsi-winst scripts, built by the community, can be found at the opsi wiki.

A collection of links to web sites with switch collections can be found at opsi wiki: Software integration web-links.

**Search the software producer’s site:** Many software manufacturers are aware of the needs of unattended software distribution, so there are often some hints and instructions in the product documentation or on the software producer’s website.

**Identify the manufacturer of the setup program:** Most setup programs are built using frameworks like Inno, NSIS, Installshield or Wise. Each one of these setup frameworks has their own switch. The following method can be used to determine the framework and other necessary information: The input strings can be determined using the command line program strings given the setup program setup.exe, and the output framework names can be found using grep or findstr.

The Linux commands looks like this (change <mysetup.exe> to the name of your setup.exe):

```bash
strings <mysetup.exe> | grep -i -E "(inno|nsis|installshield|wise)"
```

Windows does not have a native strings command, so you will have to install it. You can download a strings.exe program from here: http://technet.microsoft.com/en-us/sysinternals/bb897439

To use this program, enter these commands at the command line interface (change <mysetup.exe> to the name of your setup.exe):

```bash
strings.exe <mysetup.exe> | findstr /i /r "inno installshield nsis wise"
```

The same method is used in the opsi-setup-detector. See the example below:
This GUI program can be called from the Windows context menu Explore.

The *opsi setup detector* is part of the Windows package repositories and can be obtained through them. At the *opsi wiki: Software integration web-links* you will find links to websites that give hints on how to detect the manufacturer of the setup program.

### 10.1.11.2 Some Important opsi-winst Commands

A short overview of the opsi-winst commands can be found in the *opsi-script reference card*. All syntax details are described in the *opsi-script manual*.

Here are some hints regarding important methods:

**Stringlisten** String lists can be powerful tools to review the output from other programs. Read the opsi-winst manual for details.

**ExitWindows**

- `ExitWindows /Reboot` Reboot after the script is finished
- ExitWindows /ImmediateReboot
  Reboot now
- ExitWindows /ImmediateLogout Exit the opsi-winst now

**Product Properties** For some products it is important to know which product properties can modify the installation in order to make a client-specific installation. Creating these properties is described below in "Creating an opsi package".

To evaluate these properties, opsi-winst provides the function `GetProductProperty`

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

### 10.1.11.3 Installation When the User is Logged on

Before we begin, we assume that you have tried an unattended installation using an opsi-winst 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:

```c
[Actions]
DefVar $MsiLogFile$
Set $MsiLogFile$ = %opsiLogDir% + "\myproduct.log"
winbatch_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-winst 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-winst manual chapter 8.3 Script for installation in the context of a local administrator and use the template `opsi-template-with-admin`.

### 10.1.11.4 Working with MSI-packages

With Windows 2000, Microsoft launched its own installation concept based on the Microsoft Installer Service "MSI". Since then, many setup programs have become MSI compliant.

To be MSI compliant means to provide a package with installation instructions for the MSI. Usually this is a file named `product.msi`.

In practice, the setup.exe of a product contains a `product.msi` file and an additional control program for the installation. The control program unpacks the `product.msi` and pops up a window that asks if it is allowed to start the installation. If installation has been approved, then the control program checks whether or not MSI is installed, and if so passes `product.msi` to MSI. If no MSI is found, then the control program tries to install MSI.

If you were to interrupt the installation at that point, you will often find the unpacked MSI-package in a temporary directory.

For example, this package can be used for an unattended installation with the statement:
**10.1.11.5 Customization after a silent/unattended Installation**

After a successful silent installation, some customizing might be useful. The opsi-winst 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 Section 10.1.11.7. Some other tools can be found here:

Some other often used tools are:

- sysinternals
- regshort

**10.1.11.6 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-winst scripts:

- winwait()
- winactivate()
- Send()

Because these commands are so widely used, we provide substitutes: winwait() should be replaced by the function

opsiwinwait($title, $text, $maxseconds, $logname)

which is defined as:
Func opsinwinwait($title, $text, $maxseconds, $logname)
    Local $exists = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($exists = 0)
        $exists = WinExists($title , $text)
        FileWriteLine($mylog,"win: " & $title & " ; " & $text & " exists result (1=exists): " & $exists)
        $seconds = $seconds + 1
        sleep(1000)
    WEnd
    FileClose($mylog)
EndFunc

The parameters are:

- **$title** the title of the window
- **$text** a part of the readable text in the window
- **$maxseconds** the timeout in seconds
- **$logname** the name of the log file

Send()
should be replaced by the function
opsiControlClick($title, $text, $id, $maxseconds, $logname)
respectively by
opsiControlSetText($title, $text, $id,$sendtext, $maxseconds, $logname)
which are defined as:

Func opsiControlClick($title, $text, $id, $maxseconds, $logname)
    Local $result = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($result = 0)
        $result = ControlClick($title , $text,$id)
        FileWriteLine($mylog,"answer for " & $title & " ; " & $text & " id: " & $id & " sended: result (1=success) : " & $result)
        $seconds = $seconds + 1
        sleep(500)
    WEnd
    FileClose($mylog)
EndFunc

Func opsiControlSetText($title, $text, $id,$sendtext, $maxseconds, $logname)
    Local $result = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($result = 0)
        $result = ControlSetText ($title , $text,$id, $sendtext)
        FileWriteLine($mylog,"answer for " & $title & " ; " & $text & " id: " & $id & " set: " & $sendtext & " sended: result (1=success) : " & $result)
        $seconds = $seconds + 1
        sleep(500)
    WEnd
    FileClose($mylog)
EndFunc

The parameters are:

- **$title** the title of the window
- $text a part of the readable text in the window
- $id the numerical ControlId of the button or edit field
- $sendtext the text to insert to a edit field
- $maxseconds the timeout in seconds
- $logname the name of the log file

Therefore, you should 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:

Below is an example from a script.
In this script we produce a log file from the autoit activities, which may be integrated in the opsi-winst log file with the following commands:

```
includelog %opsiLogDir% + "\au3.log" "500"
```

Example:

```autoit
[ExecWith_autoit_confirm]
Func opsiWinwait($title, $text, $maxseconds, $logname)
    Local $exists = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($exists = 0)
        $exists = WinExists($title, $text)
        FileWriteLine($mylog, "win: " & $title & " ; " & $text & " exists result (1=exists): " & $exists)
        $seconds = $seconds + 1
        sleep(1000)
    WEnd
    FileClose($mylog)
EndFunc

Func opsiControlClick($title, $text, $id, $maxseconds, $logname)
    Local $result = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($result = 0)
        $result = ControlClick($title, $text, $id)
        FileWriteLine($mylog, "answer for " & $title & " ; " & $text & " id: " & $id & " set: " & $sendtext & " sended: result (1=success) : " & $result)
        $seconds = $seconds + 1
        sleep(500)
    WEnd
    FileClose($mylog)
EndFunc

Func opsiControlSetText($title, $text, $id, $sendtext, $maxseconds, $logname)
    Local $result = 0
    Local $seconds = 0
    Local $mylog
    $mylog = FileOpen($logname, 1)
    While ($seconds <= $maxseconds) and ($result = 0)
        $result = ControlSetText($title, $text, $id, $sendtext)
        FileWriteLine($mylog, "answer for " & $title & " ; " & $text & " id: " & $id & " set: " & $sendtext & " sended: result (1=success) : " & $result)
        $seconds = $seconds + 1
        sleep(500)
    WEnd
    FileClose($mylog)
EndFunc

; exact title match
Opt("WinTitleMatchMode", 3)
$mylog = FileOpen("%opsiLogDir%\au3.log", 2)
FileWriteLine($mylog, "auto-it started - waiting for the window")
FileClose($mylog)
```
opsiwinwait("InstallShield Wizard", "Wollen Sie wirklich", 200, "%opsiLogDir%\au3.log")
opsiControlClick("InstallShield Wizard", "Wollen Sie wirklich", 6, 5, "%opsiLogDir%\au3.log")
opsiwinwait("InstallShield Wizard", "Deinstallation ist abgeschlossen", 400, "%opsiLogDir%\au3.log")
opsiControlClick("InstallShield Wizard", "Deinstallation ist abgeschlossen", 1, 5, "%opsiLogDir%\au3.log")

Sleep(500)
; and good bye
Exit

see also:
http://www.autoitscript.com/wiki/FAQ#Why_doesn.27t_my_script_work_on_a_locked_workstation.3F
http://www.autoitscript.com/autoit3/docs/
http://www.autoitscript.com/autoit3/docs/intro/controls.htm
http://www.autoitscript.com/autoit3/docs/functions.htm

10.1.11.7 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:

- InstallWatch Pro
- appdeploy-repackager

10.1.11.8 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-winst provides several commands for software deletion:

Using an uninstall routine If the product manufacturer provides an option for software deletion, you must checked 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-winst 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-winst script as described below:

**Useful opsi-winst commands for uninstall** If a product has been installed by opsi-winst functions, or if there is no uninstall routine for the product, the complete uninstall has to be done by an opsi-winst script. opsi-winst 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-winst handbook.

Basic uninstall means deleting one or more files from the file system. This command can be executed from an opsi-winst files section:

```
delete -f <file name>
```

or to delete a directory including subdirectories:

```
delete -sf <dir name>\n```

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-winst 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-winst 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-winst command DeleteKey:

```
DeleteKey [HKLM\Software\Macromedia]
```

### 10.1.11.9 Known Issues with the 64 Bit Support

The opsi installer opsi-winst is a 32 bit program. There are no known problems when installing 32 bit software on a 64 bit system using opsi-winst. For the installation of 64 bit software, some constants (like `%ProgramFilesDir%`) give wrong values.

New versions of opsi-winst have special commands to handle these problems. So read the opsi-script manual for these issues.
10.2 Creating an opsi Package

In opsi, the new software is integrated into the system as a package. This package contains the installation files, the opsi-winst installation script, and any meta data.

The advantages of this format are essentially:

- Simplified menu driven handling using the program `opsi-newprod`.
- Holding all meta data in one file, which is easy to edit.
- Optional menu driven installation of the package, with optional default overriding.
- Information about the package will be saved; including product version, package version, and customer extensions. The package information is stored in the installation directory, and all the information can be seen in the package name and the opsi-configeditor. This means that different package versions can be easily handled (product life cycle management).
- For creating and unpacking products, no root privileges are required. Privileges of the group `pcpatch` are sufficient.

The package itself is merely a Gzip compressed cpio archive. This archive includes three directories:

- **CLIENT_DATA**
  holds the files which are to be copied into the product directory (`/var/lib/opsi/depot/<productid>`).
- **OPSI**
  The file named `control` holds the product meta data (like the product dependencies). The files `preinst` and `postinst` will be executed before and after the installation. Any customer extensions might be added here.

10.2.1 Create, Pack, and Unpack a New Product

In order to create a new opsi package, you must login to the server and do some things at the command line. To be able to do this from windows you may use putty.exe: (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

The essential commands to create and install packages are:

- `opsi-newprod`
- `opsi-makepackage`
- `opsi-package-manager -i <opsi-product-file>`

The privileges of the group `pcpatch` are required to create a new product.

Opsi makes use of parallel compression provided by `pigz` if installed. This requires a minimum version 2.2.3 or any higher version. If a sufficient version is installed, opsi will automatically use it for (de-)compression of products. Please keep in mind that archives created by `gzip` or `pigz` can profit from the bandwidth preserving synchronization via `rsync` but they are not bit-compatible. This will become relevant if you have been using `gzip` before to create your packages and synchronized these packages to other depots. If you now use `pigz` for compression an sync, it will transmit more than the expected differences. This is the case for the first synchronization after a switch of the used compression program. Any further synchronization will then again only transmit the differences. It is possible to explicitly disable the usage of `pigz` on your server by setting the value for `use_pigz` under the section `packages` in the file `/etc/opsi/opsi.conf` to `False` as shown below:

```plaintext
[packages]
use_pigz = False
```

You should create products in the directory `/var/lib/opsi/workbench`. This directory is also available as share `opsi_workbench`. The group `pcpatch` has to be owner of the directory and the directory permissions are 2770 (set `group ID` bit is set for group `pcpatch`).
10.2.1.1 Create with opsi-newprod

⚠️ Warning
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-winst`. The product type `netboot` is used for products which are activated as a bootimage (like OS installation).

![Image](image.png)

Figure 10.8: 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 10.9: 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-winst` 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:
After editing the product information you should mention the script you want to use for different activities.

Usually the **Setup script** is named *setup.ins*

Usually the **Uninstall script** is named *uninstall.ins*

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>always</td>
</tr>
<tr>
<td>always</td>
<td>not_installed</td>
<td>none</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.*
Figure 10.11: Create product dependency: No/Yes

To create a product dependency, enter the following data (help is available at the top of the window):

**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.

**Note**
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.

**Note**
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.
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-winst* 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*.

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.

**Note**
If a values contains a backslash \ it has to be doubled.
An example showing how a path would be defined: C:\\temp

---

![Create property for product mytest](image1)

**Figure 10.14:** Description of the product properties

Next, you can decide if the product property has a default value (switch).

![Open PC ServerIntegration](image2)

**Figure 10.15:** Default value of the product property
If you choose **boolean** as the data type, then the description will contain only the *Property name* and *Property description*.

![Figure 10.16: Description of a boolean property](image)

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.

![Figure 10.17: Input of the maintainer data](image)

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.

**Example of a control file:**

```plaintext
[Package]
version: 1
depends:
incremental: False

[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
```
For the next step, you will have to copy the product `opsi-winst` script, and any necessary data files (i.e. program-installation-executable.exe), into the `CLIENT_DATA` folder.

So if the script you have written is currently at `c:\test`, just mount the share `\<opsiserver\opsi_workbench` e.g. to `w:`, and then copy the complete content of `c:\test` to the directory `CLIENT_DATA`.

### 10.2.1.2 Build the Package with `opsi-makepackage`

Now you may build the package. Change to the root directory of the product (maybe `/var/lib/opsi/workbench/myproduct/`, and enter `opsi-makepackage`. The product package will be built. The package `<package name>` will be a file that has a format similar to `/var/lib/opsi/workbench/<myproduct>/<myproduct_ProductVersion-PackageVersion>.opsi`.

Finally, install the package. The resulting package can be installed on the opsi-server with the command `opsi-package-manager -i <package name>`.

`opsi-makepackage` can be started with different options:

```
$ opsi-makepackage --help
  [--log-level {0,1,2,3,4,5,6,7,8,9}] [--no-compression]
  [--archive-format {cpio,tar}] [--follow-symlinks]
  [--temp-directory directory] [--md5 | --no-md5]
  [--zsync | --no-zsync] [--no-pigz] [--keep-versions]
  [--package-version packageversion]
  [--product-version productversion]
  [source directory]

Provides an opsi package from a package source directory. If no source directory is supplied, the current directory will be used.

positional arguments:
  source directory

optional arguments:
  --help, -h   Show help.
  --version, -V show program’s version number and exit
  --quiet, -q  do not show progress
  --verbose, -v verbose
  --log-level {0,1,2,3,4,5,6,7,8,9}, -l {0,1,2,3,4,5,6,7,8,9}
  Set log-level (0..9)
  --no-compression, -n Do not compress
  --archive-format {cpio,tar}, -F {cpio,tar}
  Archive format to use. Default: cpio
  --follow-symlinks, -h follow symlinks
  --custom-name custom name, -i custom name
  custom name (add custom files)
  --custom-only custom name, -c custom name
```
It is recommended to create the packages with a corresponding md5 checksum file. This file is used amongst others by `opsi-package-updater` to check after a file transfer to ensure package integrity. This file is created by default but its creation can be suppressed for special purposes.

When transferring packages to opsi-depot-server zsync can be used to only transfer differences between different packages. To be able to use this method a special `.zsync` file is required. This file is created by default but its creation can be suppressed for special purposes.

If you are running into the problem that the creation of a package fails because of insufficient free space in `/tmp` you can use the option `--temp-directory` to specify a different temporary folder.

If there is already a package file with the same version information, opsi-makepackage will ask for overwrite confirmation:

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

Choosing o will overwrite, c abort, and n will ask for new version information.

The created opsi-package can be installed at the opsi-server with the command:

```
opsi-package-manager --install <packagefile>
```

More information about the opsi-package-manager can be found in the opsi-manual.
Chapter 11

More Information

The opsi manual contains a wide array of additional information that is important for productive operations. If you are using your opsi server productively, we recommend that you familiarize yourself with the opsi-backup tool in order to create a backup of your data.

If you do not find what you are looking for or need help, please contact the opsi community.

For productive installations we recommend professional support by uib with a maintenance and support contract.