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
      4.2.1 Notes on determining hardware requirements ............ 5
         4.2.1.1 Memory requirements .................................... 6
         4.2.1.2 CPU ....................................................... 6
   4.3 Configuration requirements ..................................... 6
      4.3.1 Valid DNS domain name .................................... 6
      4.3.2 Valid DNS hostname ........................................ 6
      4.3.3 Correct name resolution for the server .................. 7
      4.3.4 Localization settings .................................... 7
   4.4 Needed network ports .......................................... 8

5 opsi-server Base Installation ...................................... 9
   5.1 opsi-server Basic installation ................................ 9
      5.1.1 Starting up the uib preconfigured Virtual Machine .... 9
         5.1.1.1 First Start .............................................. 9
         5.1.1.2 Language selection .................................... 10
         5.1.1.3 First boot ............................................. 10
         5.1.1.4 Second Start ......................................... 12
         5.1.1.5 Terminal Window ....................................... 13
         5.1.1.6 Check the Network Connection ......................... 14
         5.1.1.7 Update the opsi-Server ............................... 14
         5.1.1.8 Install the standard opsi-products ................... 15
         5.1.1.9 Starting opsi-Server Interface ....................... 15
5.1.2 Installation on Debian / Ubuntu ........................................... 15
5.1.3 Installation on a Univention Corporate Server (UCS) ...................... 17
  5.1.3.1 Installation through Univention App-Center .......................... 18
  5.1.3.2 Upgrading an existing opsi-Installation from UCS 3 to UCS 4 (over the App-Center) .............. 18
  5.1.3.3 Manual opsi-installation on UCS (without App-Center) .............. 18
  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 ............. 21
  5.1.3.6 Synchronising data from LDAP to opsi ................................ 21
5.1.4 Installation on openSUSE or Suse Linux Enterprise Server (SLES) ........ 21
5.1.5 Installation on CentOS or RedHat Enterprise Linux (RHEL) .................. 23
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 ............................ 27
  5.2.3 Create users and configure the groups opsiadmin and pcpatch ............. 27
  5.2.4 DHCP Configuration ....................................................... 28
    5.2.4.1 Using a DHCP Server at the opsi-server ............................ 29
    5.2.4.2 Using an External DHCP Server .................................... 29
    5.2.4.3 Checking the Backend Configuration for DHCP Entries ................. 29
  5.2.5 Configuration of the name resolution ..................................... 30
5.3 Importing the minimal opsi products ........................................ 30
  5.3.1 Automatic import of the minimal opsi products ............................ 31
  5.3.2 Manual import 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 Adding 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 Creating a new opsi client using the opsi-client-bootcd .................. 34
  7.2 Integration of Existing Clients ............................................. 36
    7.2.1 Using service_setup.cmd on Windows NT6 ................................ 36
    7.2.2 Using service_setup_NT5.cmd on Windows NT5 ............................. 37
    7.2.3 Using opsi-deploy-client-agent ....................................... 37
8 Rollout existing products ....................................................... 40
  8.1 Deploying opsi standard products: opsi-configed .......................... 40
  8.2 Hard- and Software Inventory with the localboot products hwaudit and swaudit .... 40
  8.3 Hardware Inventory with the netboot product hwinvent ........................ 41
9 Installation of a new Windows PC with opsi (OS Installation)

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

9.2 NT6 family: Win7 / 2008R2 and up ................................................................. 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 Drivers that suitable for your hardware but not specially assigned .......... 50

9.6.3 Drivers manually assigned to clients .......................................... 50

9.6.4 Drivers automatically assigned to the clients using the inventory fields .... 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 Add and check drivers ................................................... 51

10 Integration of New Software Packages into the opsi Server ................................ 56

10.1 A Brief Tutorial: How to write a opsi-winst Script ............................................ 56

10.1.1 Introduction ......................................................... 56

10.1.2 Methods of Non-Interactive Installation ............................................ 56

10.1.3 Structure of a opsi-script / opsi-winst Script ......................................... 57

10.1.4 Primary Sections ...................................................... 57

10.1.5 Important Kinds of Secondary Sections ............................................. 58

10.1.6 Global Constants ...................................................... 59

10.1.7 Second Example: tightvnc .................................................. 59

10.1.8 Elementary Commands for Primary Sections .......................................... 60
10.1.8.1 String Variable ........................................................ 60
10.1.8.2 Message / showbitmap ........................................... 60
10.1.8.3 if [else] endif ...................................................... 60
10.1.8.4 Functions ............................................................. 61
10.1.8.5 Error, Logging and Comments .................................... 61
10.1.8.6 Requirements ....................................................... 61

10.1.9 Third example: The Generic Template opsi-template .................. 61

10.1.10 Interactive Creation and Testing of a opsi-winst Script ................. 68

10.1.11 Suggestions on How to Solve Problems with opsi-winst Scripts ............ 72
  10.1.11.1 Search for Unattend or Silent Switches .......................... 72
  10.1.11.2 Some Important opsi-winst Commands ............................. 73
  10.1.11.3 Installation When the User is Logged on ......................... 74
  10.1.11.4 Working with MSI-packages ..................................... 74
  10.1.11.5 Customization after a silent/unattended Installation ................ 75
  10.1.11.6 Integration with Automated Answers for the setup Program .......... 75
  10.1.11.7 Analyze and Repackage ........................................ 78
  10.1.11.8 How to uninstall Products ..................................... 78
  10.1.11.9 Known Issues with the 64 Bit Support ........................... 79

10.2 Creating an opsi Package ................................................ 80
  10.2.1 Create, Pack, and Unpack a New Product ............................ 80
    10.2.1.1 Create with opsi-newprod .................................. 81
    10.2.1.2 Build the Package with opsi-makepackage ..................... 88

11 More Information .......................................................... 90
Chapter 1

Copyright

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

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

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

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

Most parts of the opsi software is open source.
Not open source are the parts of the source code which contain new extensions, that are still under cofunding, which have not been paid off yet. See also: opsi cofunding projects
All of the open source code is published under the AGPLv3.

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

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

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

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

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

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

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

For the integration into existing networks you can use consulting services by uib.
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 of an opsi-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 Buster</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Debian 9 Stretch</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Debian 8 Jessie</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Debian 7 Wheezy</td>
<td>✗</td>
<td>⚠</td>
</tr>
<tr>
<td>Ubuntu 20.04 LTS Focal Fossa</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Ubuntu 18.04 LTS Bionic Beaver</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Ubuntu 16.04 LTS Xenial Xerus</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 14.04 LTS Trusty Tahir</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Ubuntu 12.04 LTS Precise Pangolin</td>
<td>✗</td>
<td>⚠</td>
</tr>
<tr>
<td>RHEL 7</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>RHEL 6</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>CentOS 7</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CentOS 6</td>
<td>✗</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>
</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.

4.2.1 Notes on determining hardware requirements

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

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

The following values were recommended for Samba 3:

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

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

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

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

You should also implement a safety margin here.

4.2.1.2 CPU

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

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

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

4.3 Configuration requirements

Your server and your network have to comply to 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 domainname, in which at least come two dots, e.g. opiserver.domain.local:
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:

### 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:

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

The result should look like the following example:

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

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

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

---

**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 require 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:

```bash
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 nothing or an undesired value is set:

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

To apply these settings systemwide the server should be restarted.

For more information please consult the documentation of your Linux distribution.
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
  Server to client (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-server are shown. You will end up with a server system ready for final configuration and commissioning. To evaluate opsi we recommend using the pre-installed virtual machine. Otherwise, you should select the operating system you are most familiar with.

In this case please make sure that the packages of the server are up to date.

Please set the appropriate 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 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 a new name. The virtual machine will still boot.
- 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 next 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 dialog box](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.`
Fill in the configuration information for your network and answer the questions.

In the following, you will be asked for:

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

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

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

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

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

**country**
For the creation of the SSL-certificate: Identification of the nation (two capital letters) 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 department (optional)

email 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 Internet access, the proxy information: e.g. http://myuser:mypass@192.168.1.5:8080

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

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

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

Password of root
Password of root

Password of adminuser
Password of local opsi-admin.

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

5.1.1.4 Second Start

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

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

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

- use `ssh` at the command line to access to 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 multiple desktops that are reachable using the variety of buttons in the upper-left-hand corner of the display.

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

Example snippets from configuration files are formatted like this:

```
depoturl = smb://smbhost/sharename/path
```

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

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

Angle brackets `< >` mark abstract names. When entering commands, please replace the `<abstract name>` with a real name.

For example: The file share, where opsi places the software packages, may abstractly be noted as `<opsi-depot-share>`. If the real file share is `/var/lib/opsi/depot`, then you have to replace the abstract name by this exact string. The location of the package `<opsi-depot-share>/ooffice` becomes `/var/lib/opsi/depot/ooffice`.

### 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 website using the browser in the start window.

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

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

```
1stboot.py
```

A reboot is forced with the command:

```
reboot
```

If the network connection works, then you can install opsi packages or update them, and configure the environment for the first installation test. If you want to use the virtual machine (and not install the opsi-server directly to your host system), then skip to 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 enter 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 enter the current password for the adminuser. Through this installation the actual current version 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

You can start the management interface by double clicking on the icon *opsi Management GUI*. For a description of the management 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 Debian / Ubuntu

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

---

**Important**

Please check the requirements!

---

We recommend to install the following packages:

```bash
apt install wget host pigz
```

Furthermore, samba needs to be installed:

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

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

```bash
apt install mysql-server
```

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

**Ubuntu 18.04 LTS Bionic Beaver:**

```bash
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:**

```bash
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:**

```bash
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*:

```bash
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*:

```bash
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 signing key of the repository:

Ubuntu 18.04 LTS *Bionic Beaver*:

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

Ubuntu 16.04 LTS *Xenial Xerus*:

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

Debian 10 *Buster*:

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

Debian 9 *Stretch*:

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

Debian 8 *Jessie*:

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

All:

Check the successful import by:

```bash
apt-key list
```

This should contain the output:

```
<home:uibmz:opsi@build.opensuse.org>
```

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

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

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

```bash
apt update
apt remove tftpd
# 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 base directory during the tftpd-installation answer with /tftpboot.

During the installation of the opsiconfd package, you will be asked for information for the creation of a local SSL certificate.

During the opsi-server installation, you will be asked 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) specifics:** 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`:

```
systemctl disable winbind
```

or

```
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`.

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

Because you have installed opsi on an existing machine, we assume that the network is properly configured. Next continue 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 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 available for the new version of the operating system.

**Note**

Installations on a system with 32-bit and 64-bit architecture are currently both supported.

In the future the support for 32-bit will be dropped. Therefore we recommend to only use opsi on a system with a 64-bit architecture.

With both installation methods the package `opsi4ucs` will be installed. This will prepare the UCS environment for use with opsi. This is done by using a join script. This preparation includes adding the required users, groups and shares.

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

When updating from opsi 4.0 the join script will execute the migration of the backend.
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 interface in the category *System*. With the App-Center opsi can only be installed on a server with roles *master*, *backup* or *slave*.

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

Additional packages will be installed automatically: *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 the existing server. On installation *opsi-package-updater* will be run to download and install a minimal set of opsi packages. This will not be done during an update. On an major or minor update the 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)

One of the important changes in opsi 4.0.5 is support for the group *opsifileadmins*. This group will in the future replace the group *pcpatch* in UCS. The group *opsifileadmins* has already been introduced with the support of UCS 3.0, but would only be used on installations that had Samba 4 and the Univention Directory Services (Samba4-AD) installed. In all other variants and roles the group *pcpatch* continues to be used, as before.

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

---

**Warning**

To implement this in 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 a server with the 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 first on the config server and then on the depot servers as well. Otherwise the operation on multi depot-environments could lead to issues. If your group is already named *opsifileadmins* there should be no issues. Nevertheless, it is recommended after installing 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:

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

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

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

---

**Caution**

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

---

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

• Next add the opsi4ucs repository:

**UCS 4.4:**

```bash
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/Univention_4.4/ /" > /etc/apt/
  sources.list.d/opsi.list
```

**UCS 4.3:**

```bash
echo "deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/4.1:/stable/Univention_4.3/ /" > /etc/apt/
  sources.list.d/opsi.list
```

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

**UCS 4.4:**

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

**UCS 4.3:**

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

For installation the following commands must be entered next:

**UCS 4.4:**

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

**UCS 4.3:**
If the role of the target system different than master or backup then we have to run the opsi4ucs Join-Script:

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

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

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

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

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

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

Please be advised that samba 4 will not be automatically restarted, since it is an important service on which other software may depend. You have to restart it manually. After restarting samba there may be a slight delay before the new shares are accessible.

Because there is no direct connection between the Univention LDAP and the opsi-backend all Clients have to be created twice. First in the Univention-LDAP using UDM and then in opsi including all system information (in particular the MAC address). Deleting a LDAP client in Univention will not delete the client in opsi and vice versa. This problem is further discussed in Section 5.1.3.6.

Since opsi was installed on an existing server we assume that the network configuration is correct. Continue with the installation by 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 opsi on a server with 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 exists in the current domain. Check the host parameter `clientconfig.depot.user` for this. Let’s assume that the domain is backstage, then the value has to be `backstage\pcpatch`. If it is `memberserver\pcpatch` then it has to be changed.

Setting the password for the user `pcpatch` through `opsi-admin` fails because of the missing AD write access of a member server. To change the password you have to do so additionally on a server with write access - a master, backup or slave.
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 often the configuration is already in use.
- 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 must be pxelinux.0 instead of linux/pxelinux.0.

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

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

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

5.1.3.6 Synchronising data from LDAP to opsi

In an opsi4ucs installation Windows clients have to be created in the UDM first and then they have to be created in opsi-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 an incorrect MAC address.

The 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.

- Samba must be installed and configured.
- mariadb-server must be installed.
- If the machine should also act as DHCP-server then the dhcpr daemon has to be configured and running.
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, the installation can be started:

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

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

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

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

```
127.0.0.2 <fqdn> <hostname>
```

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

Please continue with Section 5.2.

---

**Warning**
The unix commands used in the following chapters are based on Debian systems. You may have to adapt them to the corresponding commands for your linux system.
5.1.5 Installation on CentOS or Red Hat Enterprise Linux (RHEL)

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

---

Important
Please check the configuration requirements!

---

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

```bash
subscription-manager register
subscription-manager attach --auto
```

Necessary preparations:

- Install wget and add the MariaDB repository, as the version in the official repositories is outdated.

```bash
yum install wget
cd /etc/yum.repos.d/
cat <<EOF > mariadb.repo
[mariadb-main]
name = MariaDB Server
baseurl = https://downloads.mariadb.com/MariaDB/mariadb-10.5/yum/rhel/$releasever/$basearch
gpgkey = file:///etc/pki/rpm-gpg/MariaDB-Server-GPG-KEY
gpgcheck = 1
enabled = 1
EOF
cd /etc/pki/rpm-gpg
wget https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
yum makecache
```

- Install Samba and the database:

```bash
yum install mariadb-server samba samba-client
```

- Configure samba and database:

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

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

Add the repository:

**CentOS 7:**

```bash
cd /etc/yum.repos.d/
yum makecache
```
RHEL 7:

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

After adding the repository you may start the opsi installation:

```bash
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 previous steps were completed successfully we can assume that the network is properly configured. Next continue with Section 5.2.

---

**Warning**
The unix commands used in the following chapters are based on Debian systems. You may have to adapt them to match your CentOS- /RHEL system.

---

5.2 Update and Configuration of the opsi-server

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

5.2.1 Backend Configuration

Opsi supports different backends for data storage. These are essentially:

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

Besides these there are some backends for special purposes:

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

By default the mysql backend is used for inventory data. The usage of the file backend for inventory data is possible but noticeably slower and therefore not recommended.
Note
The use of the mysql backend for inventory data is free and does not require activation.
More information about the activation can be found in the opsi manual.

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

Caution
Since MySQL server version 5.7 the previously optional 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, and that
the credentials for a database administrator are known. For specific information on installation and configuration of
the database please refer to the manuals of your distribution.

For the initial configuration of the mysql backend use the command:

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

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

The following screenshots show examples for the MySQL configuration setup:

![Figure 5.5: Dialog opsi-setup --configure-mysql: Input mask](image)
You may accept the defaults for all questions except the Database Admin Password. The Database Admin Password is linux123 on the pre-installed opsi-VM, otherwise it is the password you entered during the mysql-server installation.

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

Here an example:

```plaintext
# = backend dispatch configuration =
#
# This file configures which methods are dispatched to which backends.
# Entries have to follow the form:
# <regular expression to match method name(s)> : <comma separated list of backend name(s)>
# # Backend names have to match a backend configuration
# file basename <backend name>.conf beneath /etc/opsi/backends.
# For every method executed on backend dispatcher
# the first matching regular expression will be decisive.

# Recommended standard configuration (dhcpd not at the opsi server)
# file as main backend, mysql as hw/sw invent
# and license management backend and opsipxeconfd backend:
backend_:* : file, mysql, opsipxeconfd
host_:* : file, opsipxeconfd
productOnClient_:* : file, opsipxeconfd
configState_:* : file, opsipxeconfd
license_:* : mysql
softwareLicense_:* : mysql
audit_:* : mysql
*: : file
```

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

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

```
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 available please enter the following command:

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

Please restart the samba services using the following commands:

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

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

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

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

```
opsi-admin -d task setPcpatchPassword
```

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

### 5.2.3 Create users and configure the groups opsiadmin and pcpatch

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

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

Firstly create the user:

```
useradd -m -s /bin/bash adminuser
```

then set the unix password:

```
passwd adminuser
```

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

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

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

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

**Note**

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

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

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

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

Example:

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

### 5.2.4 DHCP Configuration

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

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

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

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

e.g.

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

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

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

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

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

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

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

5.2.4.2 Using an External DHCP Server

Using the opsi-Server VM: If you use an external DHCP server, then you can uninstall the DHCP server on the opsi-server, which is done by entering:

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

Your own installation: Since opsi 4.0.3 a DHCP server will not be installed automatically in this situation.

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

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

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

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

If the opsi server runs on UCS, then `filename=pxelinux.0`.

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

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

5.2.4.3 Checking the Backend Configuration for DHCP Entries

Regardless of whether or not you use an external DHCP server, the configuration of the opsi-server must be changed. The file `/etc/opsi/backendManager/dispatch.conf` defines which backends are used (i.e. `file, mysql`).

The lines with the `backend_.*` and `host_.*` entries configure whether or not the opsi-server should work with the local DHCP configuration. If you are using the DHCP server on the opsi-server, then the backend dhcpd has to be added here. The corresponding entry with `file` backend must then look like this:
If the local DHCP service on the opsi-server isn’t used (because another server in the local network performs this task, and is also used for the opsi-clients), then the backend `dhcpd` is not required:

<table>
<thead>
<tr>
<th>backend_*</th>
<th>file, opsipxeconfd, dhcpd</th>
</tr>
</thead>
<tbody>
<tr>
<td>host_*</td>
<td>file, opsipxeconfd, dhcpd</td>
</tr>
</tbody>
</table>

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

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

### 5.2.5 Configuration of the name resolution

To install software on the clients before login, generally only the clients have to know how to contact the opsi-server. However, opsi also has a number of push features such as on_demand events, sending messages, starting remote control software, and retrieving session information.

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

Therefore we show two typical extremes:

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

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

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

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

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

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

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

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

- systemctl restart opsiconfd.service

### 5.3 Importing the minimal opsi products

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

It is possible to install the packages in automated or manual fashion. The automated way is recommended.
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 as configured as in `/etc/opsi/opsi-package-updater.conf`, automatically fetches the current packages from the opsi repository and installs them on the server.

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

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

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

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

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

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

---

**Note**

Please note that OS installation products like win7-x64, are not immediately ready for use after installation. The installation has to be supplemented by the installation files from the corresponding installation media (i.e. DVD, see Section 9.1).

---

5.3.2 Manual import of opsi products

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


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

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

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

6.1 Installation of the management interface opsi-configed

The management interface is installed as a local application on the administration PCs. In your web browser, go to the address https://<opsidepotserver>:4447/. There you will find links to installers for different operating systems. Alternatively, you can find corresponding installers under https://download.uib.de/opsi4.1/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 one PC is equipped with the management interface, further PCs can have easily have the interface Section 8.1 installed with the localboot product opsi-configed, as long as the opsi agent is already installed on the PC.

6.2 Start of the management interface opsi-configed

Start opsi-configed via the shortcut in your Start menu.
Log in as a user who is a member of the group opsiadmin.
The operation of the management interface is pretty much self explanatory. You will find detailed instructions in the opsi manual.

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 the Reload data button.
Chapter 7

Adding clients to opsi

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

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

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

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 later management. This is particularly helpful if you want to install an operating system on your computer using opsi. For the integration of clients with an already installed operating system, please read the chapter integration of existing Clients.

7.1.1 Creating a new opsi client via the graphical management interface

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

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

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

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

The list of configured opsi clients can be viewed at any time in the opsi-configed mode 'Client configuration' under the clients tab.
7.1.2 Creating a new opsi client via the command line

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

The syntax is the following:

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

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

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

```
opsi-admin -d method host_createOpsiClient testclient.domain.local "null" "Testclient" "" 00:0c:29:12:34:56 192.0.2.1
```

7.1.3 Creating a new opsi client using the opsi-client-boot-cd

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 then should see the following screen:

![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. Otherwise you have to complete the missing data by hand. You must at least give the hostname.
Then choose *OK*.

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

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

### 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 realised in several ways. After you have installed the opsi-client-agent as described below, the client will also appear in the client list of opsi-configed, unless you have already added the client there.

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

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

#### 7.2.1 Using service_setup.cmd on Windows NT6

1. Logon to the Windows client with administrative privileges.

2. Mount the share `\<opisserver>\opsi_depot` on a drive letter.

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

4. The script copies the needed files to a temporary local directory and starts from there the opsi-script (winst32.exe) elevated in order to do the installation. This may cause an UAC Message at this point.

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

![Figure 7.5: bootimage/boot-cd: netboot product list](image-url)
Caution
After installation the client reboots without notice.

7.2.2 Using service_setup_NT5.cmd on Windows NT5

1. Logon to the Windows client with administrative privileges.
2. Mount the share \opsiserver\opsi_depot on a drive letter.
3. On the drive 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 server via the opsi webservice in order to create the client on the serverside and to retrieve the pckey. This is tried first with the user and password provided in config.ini. If the connection fails, a login window will appear, with the Service-URL (opsi-config-server), and user and password. The user required here needs to be a member of the group opsiadmin.

Warning
After installation the client reboots without notice.

7.2.3 Using opsi-deploy-client-agent

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

Requirements for the clients are:

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

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

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

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


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

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

With the opsi-deploy-client-agent script you can also install to a list of clients. To do this, either any number of clients can be passed as the last parameter or the clients can be read from a file using the -f option. When using a file, there must be a client on every line.
The script can work with IP addresses, hostnames or FQDNs. It will try to automatically detect what type of address it is processing.

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

```plaintext
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

    [--debug-file DEBUGFILE] [--username USERNAME]
    [--password PASSWORD]
    [--use-fqdn | --use-hostname | --use-ip-address]
    [--ignore-failed-ping]
    [--reboot | --shutdown | --start-opsiclientd | --no-start-opsiclientd]
    [--hosts-from-file HOSTFILE]
    [--skip-existing-clients]
    [--threads MAXTHREADS] [--depot DEPOT]
    [--group GROUP] [--smbclient | --mount]
    [--keep-client-on-failure | --remove-client-on-failure]
    [host [host ...]]

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).
  --password PASSWORD, -p PASSWORD password for authentication
  --use-fqdn, -c Use FQDN to connect to client.
  --use-hostname Use hostname to connect to client.
  --use-ip-address Use IP address to connect to client.
  --ignore-failed-ping, -x try installation even if ping fails
  --reboot, -r reboot computer after installation
  --shutdown, -s shutdown computer after installation
  --start-opsiclientd, -o Start opsiclientd service after installation (default).
  --no-start-opsiclientd Do not start opsiclientd service after installation.
  --hosts-from-file HOSTFILE, -f HOSTFILE File containing addresses of hosts (one per line). If there is a space followed by text after the address this will be used as client description for new clients.
  --skip-existing-clients, -S skip known opsi clients
  --threads MAXTHREADS, -t MAXTHREADS number of concurrent deployment threads
  --depot DEPOT Assign new clients to the given depot.
  --group GROUP Assign fresh clients to an already existing group.
  --smbclient Mount the client’s C$-share via smbclient.
  --mount Mount the client’s C$-share via normal mount on the server for copying the files. This imitates the behaviour of the ‘old’ script.
  --keep-client-on-failure
```
If the client was created in opsi through this script it will not be removed in case of failure. (DEFAULT)

--remove-client-on-failure
   If the client was created in opsi through this script it will be removed in case of failure.
Chapter 8

Rollout existing products

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

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

8.1 Deploying opsi standard products: opsi-configed

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

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

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

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

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

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

8.2 Hard- and Software Inventory with the localboot products hwaudit and swaudit

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

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

Switch to the tab Product configuration, look for the line with the product hwaudit. Click in the column Requested Action, and select the action setup. Repeat this for the product swaudit.

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

Reboot the client. The opsi-client-agent should start and install the products hwaudit and swaudit. With hwaudit and swaudit, hardware and software information is collected and transmitted to the opsi-server. After the installation you can find opsi-configed in the start menu. The collected data is displayed under the Hardware information or Software inventory tabs.
8.3 Hardware Inventory with the netboot product hwinvent

If the product hwinvent is installed on your opsi server and you have added a client Section 7.1 which is configured to boot over the network, you can do something else useful: Hardware inventory when there is no operating system installed.

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

Then reboot the client. It should now pull a Linux image over the network (via PXE), to scan the hardware of the PC and then reboot it. If the computer was not otherwise already set up, afterwards the message appears that no operating system is installed on the disk.

The results of the hardware scan have been transmitted to the opsi-server. The results can be viewed under the Hardware information tab.

---

Note

In case the screen remains black after booting the bootimage or if the network card does not work (correctly), the start parameters of the bootimage may have to be adjusted for this specific hardware.

This can be achieved using opsi-configed in the Host parameters tab by editing the entry opsi-linux-bootimage.append. More information can be found in the opsi manual, in the chapter netboot products.
Chapter 9

Installation of a new Windows PC with opsi (OS Installation)

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

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

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

For the following chapter you should create a corresponding client in opsi Section 7.1. This can be done easily 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.

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

9.2 NT6 family: Win7 / 2008R2 and up

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

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

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

- **Windows 10 / 8.1 ADK**

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

- link: [WAIK Windows 7](#)

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

### 9.2.1 Creating a PE

The simplest method requires a computer that has opsi-client-agent installed, as well as the Windows ADK (Win8.1, Win10). The manual method is described below in Section 9.2.1.2.

#### 9.2.1.1 Automated PE creation using opsi

- Using opsi-configed set the localboot-product `opsi-winpe` to once for the client you intend to use, if desired adjust the property to `x86` instead of `x64` at the lower right side, and save (right click > save).
- If the opsi-product `opsi-winpe` is missing, install it onto your opsi-server with the command `opsi-package-updater -v install opsi-winpe`.
- Launch an installation event for the client (right click > on-demand, or reboot).
- After a successful completion of this action, move or copy the contents of the now existing folder on your client `C:\winpe\<ARCH>\media\` into the pre-existing folder within the OS folder you want to install: `\\opsiserver\opsi_depot_rw\<operating system>\winpe\`
- Finally run the following command on the console of your opsi server. Finished.

`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 replaced with either `x86`, `amd64` or `ia64`.

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

- Copy the WinPE

`copype.cmd <ARCH> C:\winpe`

- Mount the Image
opsi Getting Started opsi-Version 4.1

44 / 90

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

• replace startnet.cmd

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

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

• Unmount the Image

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

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

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 replaced with either x86, amd64 or ia64.

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

• Copy the WinPE

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

• Mount Image:

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

• replace startnet.cmd

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

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

• Unmount the Image

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

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

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

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

opsi-set-rights /var/lib/opsi/depot/<productid>/winpe
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 specially recommended for use in a PE. These instructions only work with Windows 7. (Windows Vista does not inherit the needed DISM- Deployment Image Servicing and Management.) These instructions assume that you have already completed the previous chapter and have created a PE.

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 7-zip or the command-line-tool Expand.exe. For simplicity, we recommend creating a directory called "dell-driver" on the local disk, and then extracting the CAB-File into this directory.

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

  ```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. Normally a PE-image is a one-image-file, so you can generally use the index 1, but it is better to check first.

- The next command mounts the image for modification:

  ```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 32-bit, the x64 must be replaced with x86. The Driver-CAB from Dell contains the 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. Furthermore, with the option /ForceUnsigned it is possible to integrate unsigned drivers to the image.

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

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

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

  ```bash
opsi-set-rights /var/lib/opsi/depot/<productid>/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`. Any modifications to this file should be made in this directory and not in the opsi directory.

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

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

9.2.4 Driver Integration

The driver integration proceeds as described here: Section 9.6.

9.2.5 Providing the Installation Files

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

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

9.2.6 Installation Log files

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

9.3 Windows Product Key

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

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

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

- select a client
- switch to the netboot products tab
- select the product (e.g. win7-x64)
- change the product property productkey in the lower right corner
• enter the key in the value field
• exit the input field and save the changes

Another possibility is to use the command line. By specifying an opsiserver: `<opsiserver.domain.local>`, common values are queried or set for all clients at the depot:

To view the server defaults use (if necessary, modify the product id and replace `<opsiserver.domain.local>` with the FQDN of your opsi server.):

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

To change the default values, you can proceed as follows: Read the data and write the result to a file. This file can now be changed and the result can be written back.

Reading and writing to a file (if necessary, modify the product id and replace `<opsiserver.domain.local>` with the FQDN of your opsi server.):

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

Edit the resulting file `/tmp/property_config.json` and change the entries for `values` of the desired properties. Read the modified file with the command:

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

Check the result using the following command (if necessary, modify the product id and replace `<opsiserver.domain.local>` with the FQDN of your opsi server.):

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

### 9.4 Start the Windows Installation

To start a Windows installation, select the relevant client in opsi-configed, set in the **Netboot products** tab the action to setup for the desired operating system (e.g. win7-x64). Click on the red checkmark (which turns green again).

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

**Note**

If the screen remains black after loading the boot image or the network card does not work correctly, the start parameters of the boot image may have to be adjusted for this specific hardware.

You can do this in opsi-configed in the **Host parameters** tab at the entry `opsi-linux-bootimage.append`. You can find details on this in the opsi manual in the **Netboot Products** chapter.

**Caution**

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

This chapter applies to the Windows netboot products.

9.5.1 Directory Tree Overview

```
<productid>-
  |-i386/
  |-installfiles/  NT5 only: Installation files
  |-winpe/  NT6 only
  |-opsi/  scripts and templates by opsi.org
    | |-$oem$/  NT5 only: $oem$ according to Microsoft
    | |-postinst.d/  scripts after OS-install by opsi.org
    | |-unattend.(txt/xml).template  Template by opsi.org
    |-custom/
      | |-$oem$/  NT5 only: $oem$ according to Microsoft by 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
    |-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 metadata of the product as prepared from the package maintainer. These files are here for information purposes only. Changes to this file have no effect on the system.

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

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

9.5.3 Directory installfiles / winpe

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

- `winpe`
  Contains a bootable winpe image.
9.5.4 Directories opsi and custom

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

The opsi directory contains files that can be overwritten without notice by updates. So no changes to these files should be made. For adjustments, you can make changes in the directory custom, which is preserved during updates.

The subdirectory postinst.d contains scripts which are started via the `postinst.cmd` after the actual installation of the operating system, e.g. to install the opsi-client-agent. The scripts are processed in alphabetical order. To clarify the order of execution, the file names begin with a two-digit number (10_dhcp.cmd). If you want to make extensions here, you can store scripts in the custom/postinst.d directory with starting numbers between decades (13_myscript.cmd). The starting numbers 10, 20, 30,... are reserved for maintenance by opsi.org/uib. The script 99_cleanup.cmd is the final script and ends with a reboot.

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

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

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

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

There are multiple levels of driver integration:

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

How these different levels can be used is described below:

9.6.1 General Driver Packages

When the hardware configuration across the computers is very heterogeneous, then it can make sense to work with general driver packages.

General drivers can be placed under ./drivers/drivers.

You can find such general driver packages on http://driverpacks.net/.

Download the appropriate driver package to a temporary directory, and then unpack the driver package using:

```
./extract_driver_pack.py <path to the temporary directory with the compressed driverpacks>
```
This will unpack and store the drivers in the directory .drivers/drivers/.
The disadvantage of these packages is that there are also drivers that match the description of your hardware but do not necessarily work with your hardware.
Drivers which are found in .drivers/drivers/, will be matched to the corresponding hardware using the PCI IDs (or USB- or HD_Audio-IDs) in the description file, and then integrated into the Windows setup if needed.

9.6.2 Drivers that suitable for your hardware but not specially assigned

In case you have to support few different hardware configurations, you can use the drivers provided by the manufacturers.
Additional or tested drivers belong in their own directories (name and depth of the directory structure do not matter) below the directory .drivers/drivers/preferred.
Drivers located in the directory .drivers/drivers/preferred are prioritised over the drivers in .drivers/drivers/ by using the PCI IDs (or USB- or HD_Audio-IDs) in the description file, and then integrated into the Windows setup if needed.
Problems can occur when the same PCI ID can be found in the description file of different drivers in preferred. In this case a direct assignment of the drivers to the client is necessary.

9.6.3 Drivers manually assigned to clients

Additional drivers that are to be installed regardless of their assignment or detection via the PCI- or USB-IDs must be in their own directories (name and depth of the directory structure are irrelevant) below the directory .drivers/drivers/additional. Via the product property additional_drivers you can assign one or more paths of driver directories within .drivers/drivers/additional to a client. Directories specified in the additional_drivers product property are searched recursively and all included drivers will be integrated. Symbolic links are also followed. You can use this to create a directory for certain computer types (e.g. dell-optiplex-815).
If a driver for a matching PCI device (or HD audio, USB) is found in the driver directories specified via additional_drivers, then no other driver from drivers/preferred or drivers/ is integrated for this device (additional_drivers can be thought of as super-preferred). This means that additional_drivers has the function of adding drivers that would not be found via normal driver detection.

9.6.4 Drivers automatically assigned to the clients using the inventory fields

The mechanism of direct assignment of drivers to devices described in the previous section can be automated since opsi 4.0.2. The directory .drivers/drivers/additional/byAudit is searched for a directory name that corresponds to the vendor found during hardware inventory. A search is now made in this vendor directory for a directory name that corresponds to the model found during hardware inventory. If such a directory is found, this directory is treated as if it were manually assigned via the product property additional_drivers. The directory name byAudit is case sensitive. The directory names for Vendor and Model are not case sensitive (Dell and dELL are treated the same way).
Since opsi 4.0.5, the drivers for a opsi-client can be made available via opsi-configed in the Hardware Inventory tab (see: opsi manual "Automatic driver upload").
The opsi-linux-bootimage looks for drivers in the order
- <vendor>/<model> (<sku>)
  - if in the previous no match is found <system vendor>/<system model> is checked.
  - if in the previous no match is found <motherboard vendor>/<motherboard model> is checked.
Some manufacturers use model names, which are very unfavourable for this method, because you can not use some special characters such as / in file- or directory names. An example of this would be a model name like: "5000/6000/7000". A directory with this name is not permitted due to the special characters. Since opsi 4.0.3 the following special characters: < > ? * : \ / * have therefore been replaced internally by a _. With this change you can create the directory for the example as: "5000_6000_7000" and the directory is automatically assigned, although the information in the hardware inventory does not correspond to the directory structure.
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)
        |-pciids/  (Links to PCI-ID drivers)
        |-pci.ids  (PCI database)
        |-usbids/  (Links to USB-ID drivers)
        |-usb.ids  (USB database)
        |-vendors/  (Links to manufacturer drivers)
        |-additional/  (place for general driver packages)
          |-byAudit/  Model-specific drivers that
            |-<vendor> are assigned by
              |-<model> Hardware Inventory
        |-additional/  (manually assigned drivers)
        |-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 to include all drivers that are found using the property `additional_drivers` or using the inventory data in `./drivers/drivers/additional/byAudit`. As part of the integration of drivers, it is checked for which hardware of a device (based on the PCI-, USB-, HD-Audio IDs) a driver has been made available in this way. Only for devices that are not matched by a driver, the following methods are used in order to find a matching driver.

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

**Integration** of drivers means the following:

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

9.6.7 Add and check drivers

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

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

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

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

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

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

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

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

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

```
./show_drivers.py pcdummy

PCI-Devices
[(Standard Systemgeräte), Standard PCI to PCI bridge]
   No driver - device directory /var/lib/opsi/depot/<productid>/drivers/pciids/1022/9602 not found
   [ATI Technologies Inc., Rage Fury Pro (Microsoft Corporation)]
      Using build-in windows driver
[(Standard-IDE-ATA/ATAPI-Controller), Standard-Zweikanal-PIC-IDE-Controller]
   /var/lib/opsi/depot/<productid>/drivers/drivers/D/H/B/12B
[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_sb7xx
[(Standard-USB-Hostcontroller), Standard OpenHCD USB-Hostcontroller]
   No driver - device directory /var/lib/opsi/depot/<productid>/drivers/pciids/1002/4397 not found
[ATI Technologies Inc, ATI SMBus]
   /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/ati_smbus

USB-Devices
[(Standard-USB-Hostcontroller), USB-Verbundgerät]
   /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]
   /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPI13.inf
   /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPI11.inf
   /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp52852/Vista64/HDXHPI12.inf
   /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/autorun.inf
   /var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sp50134/libHDMI/IntcHdmi.inf
   /var/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
   /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27DA] Intel : Intel(R) N10/ICH7 Family SMBus Controller - 27DA
   /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27C9] Intel : Intel(R) N10/ICH7 Family USB Universal Host Controller - 27C9
   /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
[8086:27DF] Intel : Intel(R) ICH7 Family Ultra ATA Storage Controllers - 27DF
   /var/lib/opsi/depot/<productid>/drivers/drivers/preferred/R293337/WIN7
```
Example for a client with byAudit:

```
./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]
[/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/Display/Radeon \ X300-X550-X1050 Series (Microsoft Corporation - WDDM)/atiilhag.inf]
[/var/lib/opsi/depot/<productid>/drivers/drivers/additional/byAudit/nvidia/awrdacpi/pctry5detlef/MEDIA/Realtek AC\'97 Audio/oem21.inf]
```

```
PCI-Devices
```

```
```

```
No driver - device directory '/var/lib/opsi/depot/<productid>/drivers/pciids/10DE/0053' not found
```

```
Using build-in windows driver
```

```
[10EC:0662] Realtek High Definition Audio
Manually selected [hp_e5800] [/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sps28428/Vista64]
```

```
US-Devices
[0461:0010] (Standardsystemgeräte) : USB-Eingabegeräte
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
[0461:4D20] (Standardsystemgeräte) : USB-Eingabegeräte
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/058F' not found
[0461:0010] (Standard-USB-Hostcontroller) : USB-Verbindergerät
No driver - vendor directory '/var/lib/opsi/depot/<productid>/drivers/usbids/0461' not found
```

```
HD-Audio-Devices
[10EC:0662] Realtek High Definition Audio
Manually selected [hp_e5800] [/var/lib/opsi/depot/<productid>/drivers/drivers/additional/hp_e5800/sps28428/Vista64]
```
opsi Getting Started opsi-Version 4.1

TIPS

- Directory names such as NDIS1 contain Vista drivers; NDIS2 contain Win7 drivers
- NDIS versions:


NDIS 6.0: Windows Vista
NDIS 6.20: Windows 7, Server 2008 R2
NDIS 6.30: Windows 8, Windows Server 2012
NDIS 6.50: Windows 10, version 1507
NDIS 6.60: Windows Server 2016 and Windows 10, version 1607
NDIS 6.70: Windows 10, version 1703
NDIS 6.80: Windows 10, version 1709
NDIS 6.81: Windows 10, version 1803
- Some chipset drivers contain description files, which specify hardware without actually providing drivers. An example would be the `cougar.inf` or `ibexahci.inf` from Intel. If such a pseudo driver directory is assigned via `additional_drivers` (or `byAudit`), this means that the hardware listed here is excluded from further searches for drivers in the `preferred` directory.

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

- Check the output of `./show_drivers.py` carefully!
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:

```
[Actions]
WinBatch_tightvnc_silent_install

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

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_tightvnc_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.

- **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

```
setup
  Head
  uninstall
  install

uninstall
  Head
  uninstall
```

Figure 10.1: double code for deinstallation

```
setup
  Head
  uninstall
  install
  uninstall

uninstall
  Head
```

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 an 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 source code 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"

; the path where the product will be found after the installation
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
[Winbatch_install]

; You can 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" /l* $LogDir$\$ProductId$.install_log.txt /qb-! ALLUSERS=1
; REBOOT=ReallySuppress
; "%ScriptPath%\setup.exe" /s /v" /qb-! ALLUSERS=1 REBOOT=ReallySuppress"
;
; === InstallShield
;=======================================================================================
; Create setup.iss answer file by running: setup.exe /r /f1"c:\setup.iss"
; You may use an answer file by the parameter /f1"c:\setup.iss"
; "%ScriptPath%\setup.exe" /s /ms /f2"$LogDir$\$ProductId$.install_log.txt"
;
; === Inno Setup
;=======================================================================================
; http://unattended.sourceforge.net/InnoSetup_Switches_ExitCodes.html
; You may create setup answer file by: setup.exe /SAVEINF="filename"
; You may use an answer file by the parameter /LOADINF="filename"
; "%ScriptPath%\setup.exe" /sp- /silent /norestart /nocancel /SUPPRESSMSGBOXES

[Files_install]

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

[Registry_install]

; Example of setting some values of an registry key:
;
; openkey [HKEY_LOCAL_MACHINE\Software\$ProductId$]
; set "name1" = "some string value"
; set "name2" = REG_DWORD:0001
; set "name3" = REG_BINARY: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 an 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 licensepool 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
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 sourcecode 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("", $ProductId$)
; If there is an assignment of a license pool to a windows software id, it is possible to use
; DefVar $WindowsSoftwareId$
; $WindowsSoftwareId$ = "...
; set $result$ = FreeLicense("", ", $WindowsSoftwareId$)

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:
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.

Figure 10.3: opsi-winst Started in Interactive Mode
Figure 10.4: opsi-winst log view window

- 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.
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 handle 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):

```
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):

```
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`

```plaintext
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:

```plaintext
[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 opsinwait($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:

```plaintext
includeLog %opsiLogDir% + "\au3.log" "500"
```

Example:

```plaintext
[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, $id)
  FileClose($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 & " \n 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.t_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:
If none of these methods are available or sufficient, the uninstall can be done using a 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 a 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 a opsi-winst files section:

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

or to delete a directory including sub directories:

```plaintext
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:

```plaintext
[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):

```plaintext
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:

```plaintext
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:

```plaintext
DeleteVar <VarName>
```

To delete a registry key with all sub keys and registry variables, you can use the opsi-winst command DeleteKey:

```plaintext
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).

![Select product type](image)

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>none</td>
</tr>
<tr>
<td>always</td>
<td>installed</td>
<td>always</td>
</tr>
<tr>
<td>once</td>
<td>not_installed</td>
<td>none</td>
</tr>
<tr>
<td>custom</td>
<td>unchanged</td>
<td>unchangend</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**.
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 value contains a backslash \ it has to be doubled.
An example showing how a path would be defined: C:\\temp

---

Figure 10.14: Description of the product properties

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

---

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*.

![Property name (identifier) myboolprop Property description yes or no](image)

**Figure 10.16: Description of a boolean property**

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.

![Maintainer name Maintainer e-mail address](image)

**Figure 10.17: Input of the maintainer data**

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 \(<\text{package name}>\) 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 \(<\text{package name}>\).

.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]
 [--custom-name custom name | --custom-only custom name]
 [--temp-directory directory] [--md5 | --no-md5]
 [--zsync | --no-zsync] [--package-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.

```bash
positional arguments:
  source directory

optional arguments:
  --help         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}, -- {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 use in production. If you are using your opsi server in production, we recommend that you familiarize yourself with the *opsi-backup* tool in order to create a backup of your data.

If you cannot find what you are looking for or need help, please visit the opsi community.

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