

opsi Getting Started opsi-Version 4.0.1



uib gmbh
Bonifaziusplatz 1b
55118 Mainz
Tel.: +49 6131 275610
www.uib.de
info@uib.de

Contents

1	Copyright	1
2	Introduction	2
2.1	Steps for installation and getting starting	2
2.2	Hardware requirements	2
2.3	Configuration requirement	3
3	Installation	4
3.1	opsi-server base installation	4
3.1.1	Starting up a Vmware Machine	4
3.1.1.1	First Start	4
3.1.1.2	Language selection	5
3.1.1.3	„1stboot“	5
3.1.1.4	Second start	7
3.1.1.5	Terminal window	8
3.1.1.6	Check and if necessary correct the network connection	9
3.1.2	Installation of a Debian / Ubuntu System	9
3.1.3	Installation on a univention corporate server (UCS)	12
3.1.4	Installation on openSUSE	13
3.1.5	Installation auf Suse Linux Enterprise Server (SLES)	15
3.1.6	Installation auf einem RedHat Enterprise Linux (RHEL)	16
3.1.7	Installation on CentOS Server	17
3.2	Update and configuration of opsi-server	18
3.2.1	Proxy entry in apt-configuration	18
3.2.2	Update of the opsi-server	19
3.2.3	Backend Configuration	19
3.2.4	Set Samba Configuration and Change passwords	21
3.2.5	Checking the java configuration	22
3.2.6	Create users and administrate the groups opsiadmin / pcpach	22
3.3	DHCP configuration	23
3.3.1	Using DHCP server at the opsi-server	23

3.3.2	Using external DHCP server	23
3.3.3	Checking the backend configuration for DHCP entries	24
3.4	Configure how to get the clients IP-Address	24
3.5	Install and check of the activation file	25
3.6	Install the minimal opsi products	26
3.7	Start of the opsi-configed	26
4	First steps	27
4.1	Software Deployment	27
4.1.1	Integration of existing clients	27
4.1.1.1	Usage of service_setup.cmd	27
4.1.1.2	Usage of the opsi-deploy-client-agent	28
4.1.2	First Tests	28
4.1.2.1	Hard- and Softwareinventory with the products hwaudit and swaudit	28
4.1.2.2	Hardware inventory with the netboot product hwinvent	29
4.2	Installation of a new Windows Machine using opsi (OS Installation)	29
4.2.1	Creating a new client via opsi management interface	29
4.2.2	Hardware inventory with the netboot product hwinvent	30
4.2.3	Create a new client using the opsi-client-bootcd	30
4.2.4	OS-Installation: Complete the base package for Windows	32
4.2.5	NT 5 family: XP, 2003	32
4.2.5.1	Copy the i368 directory	32
4.2.6	NT 6 family: Vista / 2008 / Win7	32
4.2.6.1	Creating a PE	33
4.2.6.2	Extending a PE	33
4.2.6.3	unattend.xml	34
4.2.6.4	Driver integration	35
4.2.6.5	Providing the installation files	35
4.2.6.6	Log files of the installation	35
4.2.7	Windows product key	35
4.2.8	Start the Windows installation	36
4.2.9	Structure of the unattended installation products	36
4.2.9.1	Directory tree overview	37
4.2.9.2	The files	37
4.2.9.3	Directory i386 / installfiles / winpe	37
4.2.9.4	Directory opsi / custom	38
4.2.9.5	Directory drivers	38
4.2.10	Simplified driver integration for the automatic OS installation	38

5	Integration of new software packets into the opsi software deployment.	41
5.1	A small tutorial: How to write a opsi-winst script	41
5.1.1	Introduction	41
5.1.2	Methods of not interactive installation	41
5.1.3	Structure of a winst script	42
5.1.4	Primary sections	42
5.1.5	Important kinds of secondary sections	43
5.1.6	Global Constants	44
5.1.7	Second example: tightvnc	44
5.1.8	Elementary commands for primary sections	45
5.1.8.1	String Variable	45
5.1.8.2	Message / showbitmap	45
5.1.8.3	if [else] endif	45
5.1.8.4	Functions	46
5.1.8.5	Error, logging and comments	46
5.1.8.6	Requirements	46
5.1.9	Third example: The generic template <i>opsi-template</i>	46
5.1.10	Interactive creation and testing of a opsi-winst script	53
5.1.11	Hints to detail problems	56
5.1.11.1	search unattend or silent switches	56
5.1.11.2	Some more important opsi-winst commands	56
5.1.11.3	Installation with a logged on user	57
5.1.11.4	Work with MSI-packages	57
5.1.11.5	Customizing after a silent/unattended installation	58
5.1.11.6	Integration with automated answers for the setup program	58
5.1.11.7	Analyze and repackage	59
5.1.11.8	How to deinstall products	59
5.1.11.9	Known issues at the 64 Bit support	61
5.2	Creating an opsi package	61
5.2.1	Create, pack and unpack a new product	62
5.2.1.1	Create with opsi-newprod	62
5.2.1.2	Build package with opsi-makeproductfile	69
6	More informations	71

Chapter 1

Copyright

The Copyright of this manual holds the uib gmbh in Mainz, Germany.

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



A German description you will find here:

<http://creativecommons.org/licenses/by-sa/3.0/de/>

The German legally binding license:

<http://creativecommons.org/licenses/by-sa/3.0/de/legalcode>

The English description:

<http://creativecommons.org/licenses/by-sa/3.0/>

The English license: <http://creativecommons.org/licenses/by-sa/3.0/legalcode>

The opsi software is in most parts open source.

Not open source are only this new parts which are still under cofunding.
see:

http://uib.de/en/opsi_cofunding/index.html

All the rest of the source code is published under the GPLv3:



The legally binding GPLv3 license:

<http://www.gnu.org/licenses/gpl.html>

The name *opsi* is a registered trade mark of the uib gmbh.

The opsi-logo is owned by the uib gmbh and may be used only with explicit permission.

Chapter 2

Introduction

This instruction explains in detail installation and starting of an opsi-server. It starts from the provided installation package and leads to a the test installation of a client.

The shown network configuration is exemplary and relates to a net without concurrent DHCP server (for example an isolated test net with an opsi-server and some clients for the first trials).

We strongly recommend to do the first trials with opsi in a test network, separated from other DHCP servers. Temporary connection to the main net can be done for download of actual product packages.

For an integration of opsi in your existing productive environment, uib provides consulting services for you.

2.1 Steps for installation and getting starting

The four steps to install and start an opsi-server are:

1. base installation of the server
2. configuration of the server:
configuration of the network, password setting, server update
3. download and installation of the required opsi products for the clients
4. completion of the system software base packages for Windows by using the original Windows CDs

At that point a client can be installed automatically.

To meet your requirements there are different versions of a base installation available. The special installation procedure for the different versions is described in siehe### chapter 2 of this introduction.

2.2 Hardware requirements

For a opsi-server the following hardware is recommended:

- Intel-x86-compatible PC
- network interface card supported by the standard linux kernel
- a hard disk with 16 GB capacity or more
- a bootable CD ROM drive

For testing as for production environment the capacity requirements of the server are moderate.

Working with a VMware machine needs a reasonable host computer. For testing a client can be run as another VMware machine on the same host computer.

2.3 Configuration requirement

Your server and your network have to comply the following requirements to install and work with opsi:

- **valid DNS domain name**

Your DNS domain name should contain at least a domain name and a top level domain. So the full qualified domain name (FQDN) should contain one or more dots. The top level domain must contain at least two chars.

Valid domain names are e.g.: *domain.local* , *uib.de*, *subdomain.domain.de* Not valid are for example: *mydomain.d* because this is only one char at the top-level domain Not valid are for example: *mydomain* because this is only a top-level domain

see also:

http://en.wikipedia.org/wiki/Domain_name

- **valid DNS hostname**

The hostnames (also the client hostnames) have to follow the naming rules. They may contain the ASCII letters a-z, digits 0-9 and the hyphen -. No underscores are allowed.

see also:

<http://en.wikipedia.org/wiki/Hostname>

- **correct name resolution for the server**

Execute the following command and check the result:

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

The result should look like the following example:

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

The result has the scheme:

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

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

Chapter 3

Installation

3.1 opsi-server base installation

This chapter describes different versions of installing an opsi-server. You can choose your installation type and skip the other stages.

If all required steps are successful, the server system is ready for configuration and starting. Finally you should update your system according to the chapter *Update of the opsi-server*.

We recommend to use the VMware-Machine for evaluation purpose.

In order to get your system work, you should execute the commands in the

marked fields

(e.g. via cut&paste from this document)

If there are any problems, please ask for help at <https://forum.opsi.org>

3.1.1 Starting up a VMware Machine

An *opsi-server* can be installed as a virtual machine, for the performance requirements are low. For VMware a ready to use configured virtual machine is provided. The data files are available in the Internet. In order to run this machine, the free of charge VMware player is sufficient.

You may also use VMware ESX. In this case you should use the VMware converter to import the virtual machine. It may happen that you have to change the SCSI controller manually after the Import to ESX.

3.1.1.1 First Start

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

- Download the file *opsi4.0-servervm.zip* from the internet.
- Unzip the file and a directory *opsidemo* will be generated.
- Start the VMware player. Search with the data file browser the directory *opsidemo* and choose the file *opsidemo.vmx*. You might get a message that the CDROM and floppy disc device have another address. You can ignore this message and the virtual machine boots.

The VMware player is free of charge and available for all common operating systems at vmware.com. Usually it can be installed without problems, if the equipment of the host computer (especially memory) meets the needs of parallel running software systems.

The virtual machine provided by uib is based on Linux. The properties of our host system are described in the configuration file *opsidemo.vmx*. If you run the *opsi-server* image under Windows or if your Linux system devices have another address, you have to adapt the file.

3.1.1.2 Language selection

The first step is to choose the preferred language:

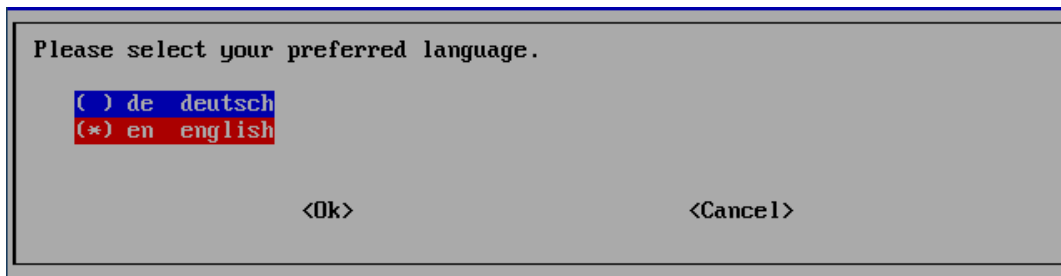


Figure 3.1: Language selection

3.1.1.3 „1stboot“

To work with the opsi-server it should be connected to the internet. For network configuration the script *1stboot.py* will start automatically at the first boot.

You may call *1stboot.py* again from the command line if something went wrong.



Warning

You can not use *1stboot.py* to rename your *opsi-server* afterwards!

The log file of *1stboot.py* is */var/lib/1stboot/1stboot.log*.

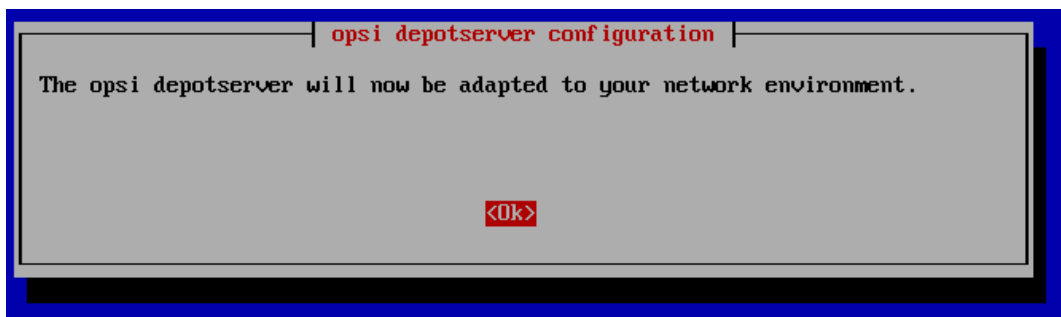


Figure 3.2: 1stboot.py Startup mask

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

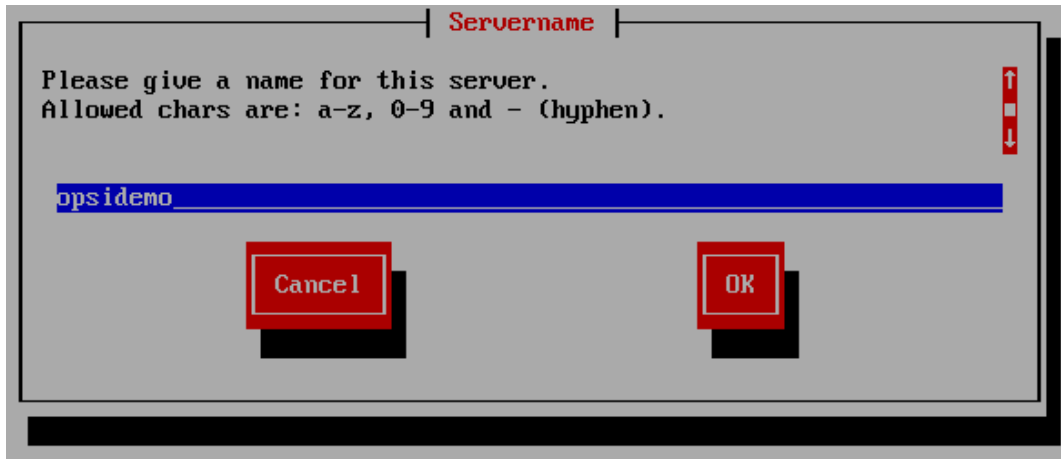


Figure 3.3: 1stboot Input mask

In the following you will be asked for:

server name

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

domain

DNS-Domain (not Windows-Domain) – the name has to include a point e.g. opsi.local

ip address

Address of this server e.g. 192.168.1.50

netmask

Net mask of this server e.g. 255.255.255.0

windows domain

Name of the Windows Domain (not the DNS domain)

country

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

state

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

city

For the creation of the SSL-certificate: Identification of the city e.g. Mainz

organization

For the creation of the SSL-certificate: Identification of the company e.g. uib gmbh

organisational unit

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

email address

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

gateway

IP-adress of the internet gateway e.g. 192.168.1.1

proxy

If required for the internet access the proxy information: e.g. <http://myuser:mypass@192.168.1.5:8080>

DNS server

ip address of the name server e.g. 192.168.1.1

mail relay

ip address of the mail server e.g. 192.168.1.1

tftp server

ip of the tftp server (usually the server)

Password of root

Password of root

After finishing the program *1stboot.py* the machine will be rebooted.

A technical note about the program *1stboot.py*:

The program works with templates to modify the configuration files. The templates are located in:
`/var/lib/1stboot/templates/`.

They can be edited for subsequent use.

3.1.1.4 Second start

After the reboot login as *root* with your root password.

The graphical user interface of the opsi-server is started (implemented as a sustainable window manager). As a welcome a „Firefox“ browser window comes up with further instructions and information, so as a reference to the getting started document (the document you are currently reading).

If you get a message that there is no network connection, you should again reboot the server. This might solve the problem.

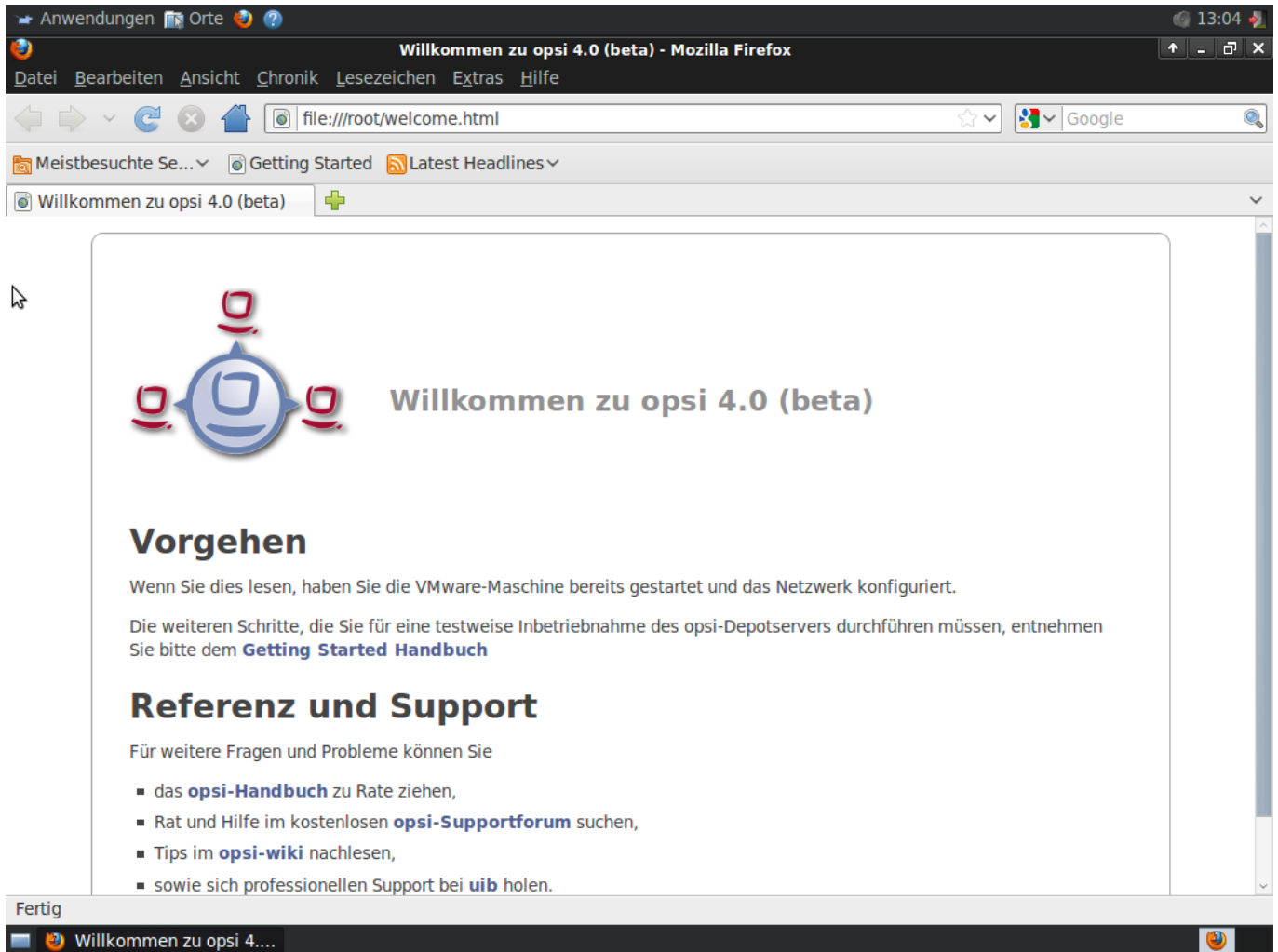


Figure 3.4: View of fresh started opsi-server

If the network configuration information was correct you are able to grab per remote on the opsi-server:

- Via ssh (in linux systems always existent, under Windows with putty, s. <http://www.chiark.greenend.org.uk/~sgtatham/putty/>) you can hit on the command line of the server.

As user name you use *root* and authenticate with the root password.

3.1.1.5 Terminal window

In the following some orders has to put in the command line. It could be a fast way to get the wished result.

A window for the input of orders i.e. a terminal window you can get in different ways:

- Remote access per ssh on the *opsi-server* (see siehe ##the last chapter)
- Open a terminal window in the graphic surface with a click on the terminal icon in the icon bar.
- Open a terminal window in the graphic surface with a right mouse click in the surface and the choice of „Terminal“. Helpfully: the graphic surface has many working surfaces reachable with the choice buttons in the left upper corner of the display.

It's very advantageous to put instruction orders e.g. out of this handbook per cut and paste in a terminal window (as far as the application environment support this).

Example snippets from configuration files are formatted like this:

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

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

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

Angle brackets < > mark abstract names. In a concrete context any marked <*abstract name*> must be replaced by some real name.

Example: The file share, where opsi places the software packets, may abstractly be noted as <*opsi-depot-share*>. If the real file share is `/opt/pcbin/install`, then you have to replace the abstract name by exactly this string. The location of the packet <*opsi-depot-share*>/ooffice becomes `/opt/pcbin/install/ooffice`.

3.1.1.6 Check and if necessary correct the network connection

If the network configuration is correct and the computer is connected with the internet you can access on any address in the internet with the browser in the start window.

If not everything works you have to open a terminal window (maybe the remote access isn't possible yet but the direct server surface) and prove the network connection usual checks.

You can access the following command in the terminal window

```
1stboot.py
```

and put in the network configuration again.

A reboot is forced with the command

```
reboot
```

If the network connection works, you can put in opsi packages or actualize them. And build the environment for the first installation test. Therefore skip the next chapters and go on with Section 3.2.

3.1.2 Installation of a Debian / Ubuntu System

Opsi 4.0 is designed and tested for:

- Debian : Lenny, Squeeze
- Ubuntu: Lucid, Maverick, Natty, Oneiric

Please read the chapter Section 2.3 (if you haven't done yet).

In this chapter we assume you are familiar with the debian-package system (topic information of this topic you will find in appropriate books, on manual pages or under <http://www.debian.org/doc/>).

Please note that an opsi-server needs storage place in `/opt/pcbin` and `/var/lib/opsi`. In both directories a free space of minimum 8 GB is recommended.

We recommend the following installations:

```
aptitude install wget lsof host python-mechanize p7zip-full cabextract openbsd-inetd
```

opsi need a installed samba:

```
aptitude install samba samba-common smbclient smbfs samba-doc
```

Now you should install the mysql-server in order to use it as Backend e.g. for Inventory and license management. (The use of the mysql Backend for inventory data is free):

```
aptitude install mysql-server
```

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

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

The result should look like the following example:

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

The result has the scheme:

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

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

To start with the installation of opsi, add in the file `/etc/apt/sources.list`:

Ubuntu Lucid:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_10.04 ./
```

Ubuntu Maverick:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_10.10 ./
```

Ubuntu Natty:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_11.04 ./
```

Ubuntu Oneiric:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_11.10 ./
```

Debian Lenny:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/Debian_5.0 ./
```

Debian Squeeze:

```
deb http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/Debian_6.0 ./
```

Now execute commands to import the signature key of the repository:

Ubuntu Lucid:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_10.04/Release.key | apt-key add -
```

Ubuntu Maverick:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_10.10/Release.key | apt-key add -
```

Ubuntu Natty:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_11.04/Release.key | apt-key add -
```

Ubuntu Oneiric:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/xUbuntu_11.10/Release.key | apt-key add -
```

Debian Lenny:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/Debian_5.0/Release.key | apt-key add -
```

Debian Squeeze:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/Debian_6.0/Release.key | apt-key add -
```

All:

Check for key import success:

```
apt-key list
```

should contain the output:

```
pub 1024D/4DC87421 2010-07-23 [verfällt: 2012-09-30] + uid home:uibmz OBS Project  
<home:uibmz@build.opensuse.org>
```

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

```
aptitude update  
aptitude safe-upgrade  
aptitude remove tftpd  
update-inetd --remove tftpd  
aptitude install opsi-atftpd  
aptitude install opsi-depotserver  
aptitude install opsi-configed
```

During the tftpd-installation you may be asked for the tftp directory. Answer with `/tftpboot`. The question after the multicast support you can answer with `no`.

During the installation of the opsisconfd you will be asked for information for a SSL certificate preparation.

During the opsi-server installation you have to allow the patching of the files `dhcpcd.conf` and `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 consider chapter *Change of passwords*.

During the opsi-server installation they may warn about a missing `/etc/opsi/modules` file which you should ignore.

If you like to run the opsi management interface `opsi-configed` directly at the server, you need to install a Java Runtime Environment. from Sun/Oracle. In order to install these packages you have to go on like this:

Debian: At the file `/etc/apt/sources.list` add the branches `non-free` and `contrib` to your repositories.

The result may look like this:

```
deb http://ftp.de.debian.org/debian/ lenny main non-free contrib  
deb-src http://ftp.de.debian.org/debian/ lenny main non-free contrib  
  
deb http://security.debian.org/ lenny/updates main non-free contrib  
deb-src http://security.debian.org/ lenny/updates main non-free contrib
```

Note

If your running *squeeze* the repositories have to be *squeeze* instead of *lenny*.

To install the Java JRE execute:

```
aptitude update  
aptitude install sun-java6-jre sun-java6-plugin
```

Ubuntu: Since Ubuntu has removed the Sun/Oracle JRE from its repositories, we recommend here to use the OpenJDK. This will work fine if you start the opsi-configed as application or via webstart. Running the opsi-configed inside the browser as applet may lead to problems with the OpenJDK 6.

Ubuntu Lucid, Maverick, Natty: To install the Java JRE execute:

```
aptitude update
aptitude install openjdk-6-jre icedtea-plugin
```

Ubuntu Oneric: Since Oneric we recommend to use the version 7 of the OpenJDK. To install the Java JRE execute:

```
aptitude update
aptitude install openjdk-7-jre icedtea-plugin
```

Cause you install opsi on an existing machine we assume of a correct network configuration. So you can go on with Section 3.2.3

3.1.3 Installation on a univention corporate server (UCS)

Please read the chapter Section 2.3 (if you haven't done yet).

opsi 4.0 is tested and released for UCS 2.4

It is possible to install opsi4ucs on an UCS Master, Backup or Slave.
Installation on Memberservers is not supported.

The package opsi4ucs-ldap-schema has to be installed on an UCS Master.

The following documentation assumes that opsi is to be installed on an UCS Master.

Necessary preparations:

- The command

```
hostname -f
```

must return a fully qualified domain name containing two dots, e.g. *opsidemo.domain.local*

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

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

The result should look like the following example:

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

The result has the scheme:

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

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

- Samba must be configured
- mysql-server must be installed
- If the machine should also act as DHCP-server, the daemon dhcpd has to be configured and should be up and running

Use the following command to activate the UCS unmaintained repositories on your target server:

```
ucr set repository/online/component/backports4opsi=yes repository/online/component/backports4opsi/server=apt.univention\
.de repository/online/component/backports4opsi/parts=unmaintained
```

You don't need to run this command on the Master if you only want to install the opsi schema there.

Append the opsi repository to your `/etc/apt/sources.list` with:

```
echo "deb http://download.opensuse.org/repositories/home:uibmz:/opsi:/opsi40/ucs2.4 ./" >> /etc/apt/sources.list
```

Now import the opsi repository signature keys to your system:

```
wget -O - http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/ucs2.4/Release.key | apt-key add -
```

To install opsi on a UCS Master, run the following commands:

```
aptitude update
aptitude install opsi4ucs-ldap-schema
aptitude install opsi4ucs
```

During the opsi-server installation they may warnings about a missing `/etc/opsi/modules` file which you should ignore.

To install opsi on a UCS Backup or Slave, run the following commands:

At first run the following commands at the Master:

```
aptitude update
aptitude install opsi4ucs-ldap-schema
```

Now run the following commands at your target system:

```
aptitude update
aptitude install opsi4ucs
```

During the opsi-server installation they may warnings about a missing `/etc/opsi/modules` file which you should ignore.

If the server role is something other than UCS Master, it is required to run the join script after the installation.

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

Now you can find a link to the opsi management interface by browsing to the URL `https://<servername>`.

Only members of the group `opsiadmin` are permitted to use the opsi management interface. Please use Univention-Admin to edit the group memberships of an user. The user Administrator will be added to the opsiadmin group automatically during the installation procedure of opsi.

To increase the performance of the LDAP server to index the attribute `opsiHostId`. To do this just run the following command at the UCS master:

```
ucr set ldap/index/eq=$(ucr get ldap/index/eq),opsiHostId
/etc/init.d/slaped stop
slapindex
/etc/init.d/slaped start
```

Since you are installing opsi on an existing machine we assume that the network configuration is correct. You can continue with Section [3.2.3](#)



Warning

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

3.1.4 Installation on openSUSE

Please read the chapter Section [2.3](#) (if you haven't done yet).

opsi 4.0 is tested and released for openSUSE 11.3.

First general notes:

These packages are tested with openSUSE 11.3. We have no information if opsi 4.0 will work with other versions.

uib gmbh strongly recommend for evaluation purpose the usage of the opsi-VM.

Necessary preparations:

- The command

```
hostname -f
```

must return a fully qualified domain name containing two dots, e.g. *opsidemo.domain.local*

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

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

The result should look like the following example:

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

The result has the scheme:

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

If the result looks different (contains e.g. *127.0.0.1* or *localhost*) or the full qualified hostname does not contain one or more dots you must correct your name resolution (DNS or `/etc/hosts` file).

- Samba must be configured
- mysql-server must be installed
- If the machine should also act as DHCP-server, the daemon `dhcpcd` has to be configured and should be up and running

You can use `zypper` to add the opsi-SUSE-Repository:

```
zypper ar -refresh http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/openSUSE_11.3/home:uibmz:opsi:\
opsi40.repo
zypper -p 100 mr home_uibmz_opsi_opsi40
```

After the repository is added you may start the opsi installation:

```
zypper refresh
<Accept the key>
zypper -v install opsi-depotserver {opsi-configed}
/etc/init.d/opsiconfd restart
/etc/init.d/opsipxeconfd restart
```

During the opsi-server installation they may warnings about a missing `/etc/opsi/modules` file which you should ignore.

In case you used a tool like `yast` or `autoyast` to help you with your network configuration the tool may have created an entry in your `/etc/hosts` file that has the following pattern:

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

If you want opsi to manage the configuration of the DHCP server, you need to correct this entry to point to the servers public IP adress.

Since you are installing opsi on an existing machine we assume that the network configuration is correct. You can continue with Section [3.2.3](#)



Warning

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

3.1.5 Installation auf Suse Linux Enterprise Server (SLES)

Please note:

- opsi 4.0 is build for SLES 11 SP1
- opsi on SLES is new.
- We have no experience with productive installations of opsi on SLES.
- We have no information if opsi 4.0 will work with other SLES versions.
- uib gmbh strongly recommend for evaluation purpose the usage of the opsi-VM.

Please read the chapter Section 2.3 (if you haven't done yet).

opsi 4.0 is tested and released for SLES 11 SP1.

First general notes:

These packages are tested with openSUSE 11.3. We have no information if opsi 4.0 will work with other versions.

uib gmbh strongly recommend for evaluation purpose the usage of the opsi-VM.

Necessary preparations:

- The command

```
hostname -f
```

must return a fully qualified domain name containing two dots, e.g. *opsidemo.domain.local*

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

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

The result should look like the following example:

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

The result has the scheme:

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

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

- Samba must be configured
- mysql-server must be installed
- If the machine should also act as DHCP-server, the daemon `dhcpd` has to be configured and should be up and running

You can use `zypper` to add the SLES-Repository:

```
zypper ar --refresh http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/SLE_11_SP1/home:uibmz:opsi:\  
opsi40.repo  
zypper -p 100 mr home_uibmz_opsi_opsi40
```

After the repository is added you may start the opsi installation:

```
zypper refresh  
<Accept the key>  
zypper -v install opsi-depotserver {opsi-configed}  
/etc/init.d/opsiconfd restart  
/etc/init.d/opsipxeconfd restart
```

During the opsi-server installation they may warnings about a missing `/etc/opsi/modules` file which you should ignore.

In case you used a tool like yast or autoyast to help you with your network configuration the tool may have created an entry in your `/etc/hosts` file that has the following pattern:

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

If you want opsi to manage the configuration of the DHCP server, you need to correct this entry to point to the servers public IP adress.

Since you are installing opsi on an existing machine we assume that the network configuration is correct. You can continue with Section [3.2.3](#)



Warning

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

3.1.6 Installation auf einem RedHat Enterprise Linux (RHEL)

Please note:

- opsi 4.0 is build for RHEL 5.
- opsi on RHEL is new.
- We have no experience with productive installations of opsi on RHEL.
- We have no information if opsi 4.0 will work with other RHEL versions.
- uib gmbh strongly recommend for evaluation purpose the usage of the opsi-VM.

Please read the chapter Section [2.3](#) (if you haven't done yet).

opsi 4.0 is tested and released for RHEL 5.

Necessary preparations:

- The command

```
hostname -f
```

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

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

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

The result should look like the following example:

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

The result has the scheme:

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

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

- Samba must be configured
- mysql-server must be installed

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

Register at the Red Hat Network:

```
rhn_register
```

Add the opsi RHEL Reopository: Create the file `/etc/yum.repos.d/opsi40.repo` with the following content:

```
[opsi4]
name=opsi4.0 for RHEL/ CentOS $releasever - $basearch
baseurl=http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/RedHat_RHEL-5/
enabled=1
gpgcheck=1
gpgkey=http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/RedHat_RHEL-5/repdata/repomd.xml.key
```

After the repository is added you may start the opsi installation:

```
yum makecache
yum remove tftp-server
yum install opsi-depotserver {opsi-configed}
  Importing GPG key 0x4DC87421 "home:uibmz OBS Project <home:uibmz@build.opensuse.org>" from http://download.opensuse.\
  org/repositories/home:/uibmz:/opsi:/opsi40/RedHat_RHEL-5/repdata/repomd.xml.key
  Is this ok [y/N]: y
/etc/init.d/opsiconfd restart
/etc/init.d/opsipxeconfd restart
yum install p7zip p7zip-plugins cabextract mysql-server
```

During the opsi-server installation they may warnings about a missing `/etc/opsi/modules` file which you should ignore.

Since you are installing opsi on an existing machine we assume that the network configuration is correct. You can continue with Section [3.2.3](#)



Warning

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

3.1.7 Installation on CentOS Server

Please note:

- opsi 4.0 is build for CentOS 5.
- opsi on CentOS is new.
- We have no experience with productive installations of opsi on CentOS.
- We have no information if opsi 4.0 will work with other CentOS versions.
- uib gmbh strongly recommend for evaluation purpose the usage of the opsi-VM.

Please read the chapter Section [2.3](#) (if you haven't done yet).

opsi 4.0 is tested and released for CentOS 5.

Necessary preparations:

- The command

```
hostname -f
```

must return a fully qualified domain name containing two dots, e.g. *opsidemo.domain.local*

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

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

The result should look like the following example:

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

The result has the scheme:

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

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

- Samba must be configured
- mysql-server must be installed
- If the machine should also act as DHCP-server, the daemon `dhcpcd` has to be configured and should be up and running

Add the opsi CentOS Reopository: Create the file `/etc/yum.repos.d/opsi40.repo` with the following content:

```
[opsi4]
name=opsi4.0 for RHEL/ CentOS $releasever - $basearch
baseurl=http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/CentOS_CentOS-5/
enabled=1
gpgcheck=1
gpgkey=http://download.opensuse.org/repositories/home:/uibmz:/opsi:/opsi40/CentOS_CentOS-5/repo/repodata/repomd.xml.key
```

After the repository is added you may start the opsi installation:

```
yum makecache
yum install opsi-depotserver {opsi-configed}
  Importing GPG key 0x4DC87421 "home:uibmz OBS Project <home:uibmz@build.opensuse.org>" from http://download.opensuse.\
  org/repositories/home:/uibmz:/opsi:/opsi40/CentOS_CentOS-5/repo/repodata/repomd.xml.key
  Is this ok [y/N]: y
/etc/init.d/opsiconfd restart
/etc/init.d/opsipxeconfd restart
yum install p7zip p7zip-plugins cabextract mysql-server
```

During the opsi-server installation they may warn about a missing `/etc/opsi/modules` file which you should ignore.

Since you are installing opsi on an existing machine we assume that the network configuration is correct. You can continue with Section [3.2.3](#)



Warning

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

3.2 Update and configuration of opsi-server

3.2.1 Proxy entry in apt-configuration

Adapt if necessary the file `/etc/apt/apt.conf` on your network circumstances (put the right proxy in or comment/delete lines). You can edit your file with e.g. a program like „midnight commander“:

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

3.2.2 Update of the opsi-server

Update the opsi-server with the commands:

```
aptitude update  
aptitude safe-upgrade
```

Tip

If you will be asked during the update to modify the `smb.conf` file, you should accept this question. If you have modified the `smb.conf` before, you should hold the default and make a diff later on both files. If the question is answered with default before you read this advice you can repeat this action from the console on the opsi-server with following command:

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

3.2.3 Backend Configuration

opsi supports for storing it's data different backends.

The most important backends are:

- file (storage in files)
- ldap (storage in the ldap database) (depricated)
- mysql (storage in a MySQL database)

Beside of these main backend, there are some special backends:

- opsipxeconfd (the service for the opsi pxe boot)
- dhcpd (used for configure and restart the local dhcp service)
- jsonrpc (to redirect all calls to a other opsi-server via JSON RPC)

Now we recommend to initialize the mysql backend in order to use it for e.g. inventory data). Therefore use the command:

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

A example session:

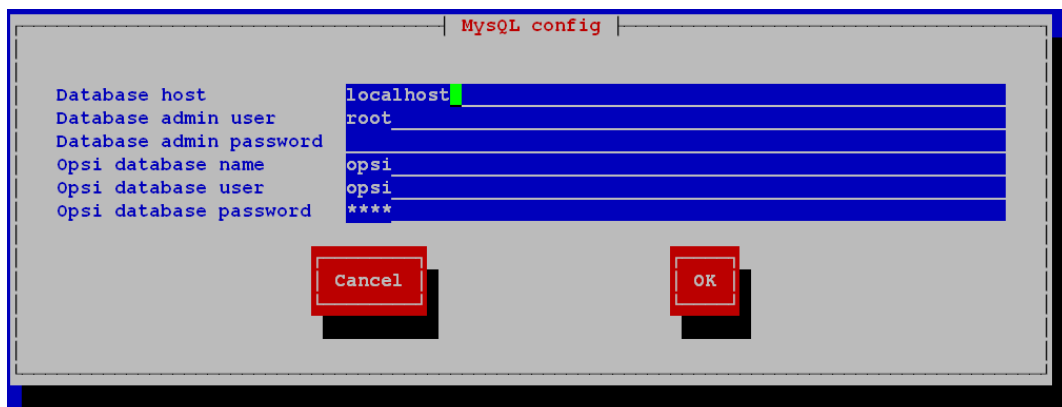


Figure 3.5: Dialog opsi-setup --configure-mysql

```

Connecting to host 'localhost' as user 'root'
Successfully connected to host 'localhost' as user 'root'
Creating database 'opsi'
Database 'opsi' created
Creating user 'opsi' and granting all rights on 'opsi'
User 'opsi' created and privileges set
Testing connection to database 'opsi' as user 'opsi'
Successfully connected to host 'localhost' as user 'opsi'
Updating backend config '/etc/opsi/backends/mysql.conf'
Backend config '/etc/opsi/backends/mysql.conf' updated
Initializing mysql backend

```

Figure 3.6: Output: opsi-setup --configure-mysql

At all questions (beside the *Database Admin Password*) you may accept the defaults by pressing ENTER. The *Database Admin Password* is at the pre installed opsi-VM *linux123*, in any other case it is the password you had given while the mysql-server installation.

Different kinds of data may stored in different backend. So for some actions (method calls) more than one backend have to be involved. Therefore the different method calls assigned to the backends. This is done in the file `/etc/opsi/backendManager/dispatch.conf`.

Here an example:

```

# = = = = =
# =      backend dispatch configuration      =
# = = = = =
#
# This file configures which methods are dispatched to which backends.
# Entries has to follow the form:
# <regular expression to match method name(s)> : <comma separated list of backend name(s)>
#
# Backend names have to match a backend configuraton
# file basename <backend name>.conf beneath /etc/opsi/backends.
# For every method executed on backend dispatcher
# the first matching regular expression will be decisive.
#
# Typical configurations:
#   file, opsipxeconfd and dhcpd backend:
#     backend_.*      : file, opsipxeconfd, dhcpd
#     host_.*        : file, opsipxeconfd, dhcpd
#     productOnClient_.* : file, opsipxeconfd
#     configState_.*  : file, opsipxeconfd
#     .*             : file
#
#   jsonrpc, opsipxeconfd and dhcpd backend:
#     backend_.*      : jsonrpc, opsipxeconfd, dhcpd
#     .*             : jsonrpc
#
#   ldap as main backend, mysql as hw/sw invent
#   and license management backend, opsipxeconfd and dhcpd backend:
#     backend_.*      : ldap, mysql, opsipxeconfd, dhcpd
#     host_.*        : ldap, opsipxeconfd, dhcpd
#     productOnClient_.* : ldap, opsipxeconfd
#     configState_.*  : ldap, opsipxeconfd
#     license.*       : mysql
#     softwareLicense.* : mysql
#     audit.*         : mysql
#     .*             : ldap

```

```
#
#   mysql and opsipxeconfd only
#   backend_*          : mysql, opsipxeconfd
#   host_*             : mysql, opsipxeconfd
#   productOnClient_* : mysql, opsipxeconfd
#   configState_*     : mysql, opsipxeconfd
#   .*                 : mysql
#
#
# Recommended standard configuration of the preinstalled VM
#   file as main backend, mysql as hw/sw invent
#   and license management backend, opsipxeconfd and dhcpd backend:
#   backend_*          : file, mysql, opsipxeconfd, dhcpd
#   host_*             : file, opsipxeconfd, dhcpd
#   productOnClient_* : file, opsipxeconfd
#   configState_*     : file, opsipxeconfd
#   license_*          : mysql
#   softwareLicense_* : mysql
#   audit_*            : mysql
#   .*                 : file
#
# 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
```

In this file you will find at the top some explemations and examples. In the lines which are not comments you will find in the first column the name of the opsi method call (with wildcard *) and in the second column the list of backends which have to be called. If any method is called the first matching line will be used. The last line (.*) matches all opsi method calls.

The default after installing the opsi packages is the file backend.

Now at the initial start-up (even if you don't changed the file) and after any change on this configuration file you should call:

```
opsi-setup --init-current-config
opsi-setup --set-rights
/etc/init.d/opsiconfd restart
/etc/init.d/opsipxeconfd restart
```

They may warn about a missing `/etc/opsi/modules` file which you should ignore.

The access rights for calling the opsi methods are configured in `/etc/opsi/backendManager/acl.conf`.

3.2.4 Set Samba Configuration and Change passwords

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

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

On the system is a pseudo-user `pcpatch` arranged. For installation of software packages the PC use the user `pcpatch` and you can access the configuration data on designed shares.

The user `pcpatch` has to be arranged with a correct password. Call in a terminal window the program `opsi-admin` and the `opsi-admin` will set the `pcpatch`-password for opsi, unix and samba (after sending the order you have to put in the password):

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

3.2.5 Checking the java configuration

Administrate the opsi-servers and the connected clients with the program *opsi-configed*. The program is written in Java and use minimum Java version 6 or version 1.6 (old version count).

Since Ubuntu has removed the Sun/Oracle JRE from its repositories, we recommend here to use the OpenJDK. This will work fine if you start the opsi-configed as applikation or via webstart. Running the opsi-configed inside the browser as applet may lead to problems with the OpenJDK 6.

To control the Java version call up

```
java -version
```

in a terminal window.

Adapt in a terminal window with `update-alternatives` the Java version if it's not indicated with minimum „1.6.0“:

```
update-alternatives --config java
```

There are 3 alternatives which provide 'java'.

Selection	Alternative
+ 1	/usr/lib/j2se/1.4/bin/java
* 2	/usr/lib/j2sdk1.5-sun/bin/java
3	/usr/lib/j2re1.6-sun/bin/java

Press enter to keep the default[*], or type selection number: 3

Using '/usr/lib/j2re1.6-sun/bin/java' to provide 'java'.

```
update-alternatives --config mozilla-javaplugin.so
```

There are 2 choices for the alternative mozilla-javaplugin.so (providing /usr/lib/mozilla/plugins/libjavaplugin.so).

Selection	Path	Priority	Status
* 0	/usr/lib/jvm/java-6-openjdk/jre/lib/amd64/IcedTeaPlugin.so	1061	auto mode
1	/usr/lib/jvm/java-6-openjdk/jre/lib/amd64/IcedTeaPlugin.so	1061	manual mode
2	/usr/lib/jvm/java-6-sun/jre/lib/amd64/libnpp2.so	63	manual mode

Press enter to keep the current choice[*], or type selection number: 2

update-alternatives: using /usr/lib/jvm/java-6-sun/jre/lib/amd64/libnpp2.so to provide /usr/lib/mozilla/plugins/libjavaplugin.so (mozilla-javaplugin.so) in manual mode.

3.2.6 Create users and administrate the groups opsiadmin / pcpatch

The opsi administration is only allowed for user members in the UNIX-group *opsiadmin*.

In the following example we create the user *adminuser* like you should create an account for your self.

Let's create the user:

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

now set the unix password:

```
passwd adminuser
```

and now the samba password:

```
smbpasswd -a adminuser
```



Caution

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

Create and test the group membership

```
adduser adminuser opsiadmin  
getent group opsiadmin
```

the getent command should have a result like:

```
opsiadmin:!:1001:opsiconfd,adminuser
```

Even *root* has to be member of the group *opsiadmin* in order to use the opsi administration commands.

All user which build opsi packages (*opsi-makeproductfile*), install (*opsi-package-manager*) or edit manually configuration files have to be also in the group *pcpatch* :

```
adduser adminuser pcpatch
```

The Test

```
grep pcpatch /etc/group
```

results

```
pcpatch:x:992:adminuser
```

root is allowed to do anything and have not to be explicit registered in the group.

3.3 DHCP configuration

Important:

A correct working name resolving and DHCP is essential for opsi. To simply the setup, the opsi-server VM is delivered with a running DHCP server. At the other hand in most productive environments a DHCP server exists and should used together with opsi. Both alternatives are described below.

3.3.1 Using DHCP server at the opsi-server

The DHCP server at the opsi-server VM is configured with no free leases, so no unknown client will get a IP-Number from this DHCP server.

If you create a client at the opsi-server using the opsi-configed, it will also create a dhcp entry for this client. Therefore you have to supply the IP-number and the MAC-address.

3.3.2 Using external DHCP server

If you use an external DHCP server you may want to disable the DHCP server at the opsi-server:

```
/etc/init.d/dhcp3-server stop  
update-rc.d -f dhcp3-server remove  
update-rc.d dhcp3-server stop 20 2 3 4 5 .
```

Next you have to configure your external DHCP server to tell the clients that our opsi-server is now the boot server. If your external DHCP runs on Linux you need the following entries for the clients in the */etc/dhcp3/dhcpd.conf* file.

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

Replace *<ip of opsi-server>* by the IP-number of the opsi-server.

Using a Windows server the corresponding entries may be *bootserver* or *startserver* and *bootfile* or *startfile* (*Options 66 / 67*).

If you create a client at the opsi-server you have to supply the MAC-address, but no IP-number.

3.3.3 Checking the backend configuration for DHCP entries

Whether you use the dhcp of the opsi-server or not you have to configure this at the opsi-server.

In the file `/etc/opsi/backendManager/dispatch.conf` is defined which backend have to be used by opsi (file, ldap, mysql).

The lines `backend_.` and `host_.` configure how changes at host entries are handled. If you are using the dhcp server at the opsi-server the backend `dhcpd` has to be added here.

The accordant entry is (e.g.):

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

If the local DHCP isn't used, also the backend `dhcpd` is not required:

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

After adapting the backend configuration you should execute:

```
opsi-setup --init-current-config
opsi-setup --set-rights
/etc/init.d/opsiconfd restart
/etc/init.d/opsipxeconfd restart
```

They may warn about a missing `/etc/opsi/modules` file which you should ignore.

3.4 Configure how to get the clients IP-Address

For the default way of software deployment via opsi only the client must know how to contact the opsi-server.

But if you like to use one of the opsi *push* features, like send messages to the client, fire *on_demand* events, get session information or start remote control software - for all these tasks the server needs to know how to get the IP-Address of the client.

How the opsi server should do this depends on your DNS/DHCP configuration and policy. There is a large number of possible configurations. So we talk here about two typical extremes:

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

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

- The entry `resolveHostAddress` at the file `/etc/opsi/backends/hostcontrol.conf`
This option controls whether the name resolution of a opsi-client address is primary done by the opsi database or by the name resolution of the operating system of the *opsi-server*.
If this option is `True`, the *opsi-server* tries at first to get the IP-Address of a *opsi-client* by the name resolution of the operating system (DNS, `/etc/hosts`) and if this fails the opsi database is used.
To use the opsi database at first, you have to set this option to `False`.
- The entry `update_ip` at the file `/etc/opsi/opsiconfd.conf`
If this entry is *yes*, so the opsi-server will update its own IP database whenever the opsi-server gets a client IP-Address (e.g. at every web service contact of a client). The default is *no*.

If you running the above variant 1, you probably should set `resolveHostAddress` to *False* and `update ip` to *yes*.

If you running the above variant 1, the best configuration is to set `resolveHostAddress` to *True* and `update ip` to *no*.

You should decide for yourself which combination fits the needs of your situation.

If you changed anything at these configurations, you should reload the `opsiconfd`:

```
/etc/init.d/opsiconfd reload
```

3.5 Install and check of the activation file

Even opsi is open source, there are some components which are not free at the moment. These components are developed in a co-funding project which means that until the complete development costs are paid by co-funders, they are only allowed to use by the co-funders or for evaluation purposes. If we have earned the development cost we will give these modules for everybody for free. To control the use of these components until they are free there is a activation file `/etc/opsi/modules`, which is protected against changes via electronic signature. If this activation file doesn't exist, only the free parts of opsi will work.

If you need for evaluation a temporary valid activation file please contact info@uib.de.

If you become a co-funder, you will get a unlimited activation file. Copy the file as root to `/etc/opsi`.

Then do the command:

```
opsi-setup --set-rights /etc/opsi
```

You may check your activation state with one of the following methods:

Using the `{opsi-configd}` choose the menu entry `Help/opsi-Module` which shows a window with the activation state.

At the command line you may use the command `opsi-admin` with the method `backend_info`. (Remark: Never give your activation file or the output of this command to third people without deleting the signature).

```
opsi-admin -d method backend_info
```

Example:

```
{
  "opsiVersion" : "4.0.1",
  "modules" :
  {
    "customer" : "uib GmbH",
    "dynamic_depot" : true,
    "vista" : true,
    "treeview" : true,
    "license_management" : true,
    "swondemand" : true,
    "expires" : "2011-04-30",
    "valid" : true,
    "multiplex" : true,
    "signature" : "THIS-IS-NOT-A-VALID-SIGNATURE",
    "vpn" : true,
    "mysql_backend" : true,
    "high_availability" : true
  }
}
```

We point out that you need the `modules-file` only for additional components and not for a general use of opsi.

3.6 Install the minimal opsi products

One important new feature of opsi 4.0 is a simple tool to update opsi products from a configured repository with the command `opsi-product-updater`. This tool compares the version of the locally installed products with the versions available at the repository in order to upgrade them. It is also possible to install additional products. How the `opsi-product-updater` works is configured at the `/etc/opsi/{opsi-product}-updater.conf` file. The default repository is <http://download.uib.de/opsi4.0/products> which may be used to install the new essential opsi products.

The `opsi-product-updater` has the following core features:

- **autoInstall:**
Install all available products from the repository.
- **autoUpdate:**
Install available products from the repository only if there is a older version installed on the server yet.
- **autoSetup:** After Installation of a updated product switch the action request to setup at all clients which have this product installed.

For more details to this function refer to the opsi-manual.

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

```
opsi-product-updater -i -vv
```

If the `opsi-product-updater` fails, it may be necessary to add a required proxy to the configuration file:

```
[repository_uib]  
proxy =
```

Please notice that the OS-Installation products like winxpro and win7 aren't ready for action after installation. The installation has to be supplemented by the installation files of the accordant installation mediums (see Section 4.2.4).

3.7 Start of the opsi-configed

Opsi offer with the opsi-configed a comfortable management interface.

You can start it different ways:

- If you are put in the address in the browser (anywhere in the net) <https://<opsi-server>:4447/configed> a web side with an embedded opsi-configed appear. Precondition is a installed java version ≥ 1.6 .
- Alternative you can click with the right mouse tab on the graphic surface to open the context menu and choose the „opsi config editor“.
- The configuration editor is also component of the opsi-adminutils which also can be copied local on the client.

Log in with the member account of the group opsiadmin (at the opsi VM you may use root since you have created new user accounts).

The handling is easy mostly self explaining. So here only a hint: Any changes have to be saved in order to show any effect. To see changes you have to reload the data.

A detailed description you will find at the opsi manual.

Chapter 4

First steps

The next step after the opsi-server installation is the integration of clients. Here we have two possibilities:

- Integration of existing Windows-Clients in opsi
- Installation of a new Windows Machine using opsi

Both ways are described below and you are free to choose which way you like to test at first.

4.1 Software Deployment

4.1.1 Integration of existing clients

To integrate existing Windows clients in opsi, the opsi-client-agent have to be installed on these systems. There are different ways to do this which are described below. After you have done so, you should see the client at the opsi-configed in the crystal tab *clients*.

4.1.1.1 Usage of service_setup.cmd

This method is the first choice for installations on a single computer or for repair purpose. For mass rollout see the chapter below.

1. login with administrative privileges
2. mount the share \\<opsiserver>\opsi_depot to drive letter
3. start the script opsi-client-agent\service_setup.cmd
4. The script connects to the opsi-webservice to create the server side client information and to get the pckey. The connect takes the user/password combination registered in the config.ini. If the connect fails, a login window pops up to fill in service URL, user and password. The provided user has to be member of the group *opsi-admin*.



Warning

During installation the client reboots without notice!

4.1.1.2 Usage of the opsi-deploy-client-agent

The `opsi-deploy-client-agent` script installs the `opsi-client-agent` directly from the `opsi-server` to the clients. Requirements for the clients are:

- an open C\$ share
- an open admin\$ share
- an administrative account

The script creates the client information on the server and copies the installation files, the configuration information and the pkey to the client and starts the installation on the client.

With the `opsi-deploy-client-agent` script you can batch install a list of clients. The script itself is located in `/opt/pcbin/install/opsi-client-agent`.

Run this script with `root` privileges.

```
bonifax:/home/uib/oertel# cd /opt/pcbin/install/opsi-client-agent
bonifax:/opt/pcbin/install/opsi-client-agent# ./opsi-deploy-client-agent -h
Usage: opsi-deploy-client-agent [options] [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.
Options:
  -h          show this help text
  -V          show version information
  -v          increase verbosity (can be used multiple times)
  -u          username for authentication (default: Administrator)
             example for a domain account: -u "<DOMAIN>\\<username>"
  -p          password for authentication
  -c          use fqdn instead of hostname for smb/cifs connection
  -x          try installation even if ping fails
  -r          reboot computer after installation
  -s          shutdown computer after installation
  -f          file containing list of clients (one hostname per line)
  -S          skip known opsi clients
  -t          number of concurrent deployment threads (default: 1)
```

4.1.2 First Tests

4.1.2.1 Hard- and Softwareinventory with the products hwaudit and swaudit

Choose in the `{opsi-configed}`, mode *Configuration of clients* in the crystal tab *Clients* the client.

If haven't done yet, reload all data by clicking at the reload button at the top, left corner of the `{opsi-configed}` (or use the *File* menu).

Switch to the crystal tab *Product configuration*, look for the line of the wished system software (*hwaudit* and/or *swaudit*). Go to the column *Requested Action* and select the action *setup*. Finally save with a click on the hook button (or with the context menu).

Now reboot the client, the *hwaudit* and/or *swaudit* should be started. The client scans the hardware and/or software inventory and send the results back to the server.

You may see this data at the crystal tabs *Hardware information* and *Software inventory*.

4.1.2.2 Hardware inventory with the netboot product hwinvent

Choose in the {opsi-configed}, mode *Configuration of clients* in the crystal tab *Clients* the client.

If haven't done yet, reload all data by clicking at the reload button at the top, left corner of the {opsi-configed} (or use the *File* menu).

Switch to the crystal tab *Netboot products*, look for the line of the wished system software (*hwinvent*). Go to the column *Requested Action* and select the action *setup*. Finally save with a click on the hook button (or with the context menu).

Now reboot the client (over PXE), the bootimage with *hwinvent* should be started. Therefore the client loads after the reboot a Linux boot image which scans the hardware and send the results back to the server.

You may see this data at the crystal tabs *Hardware information*.

4.2 Installation of a new Windows Machine using opsi (OS Installation)

4.2.1 Creating a new client via opsi management interface

You need a client (minimum 512 MB RAM) which is able to boot per PXE the network. For a first test we approve to download a corresponding vmware-image by download.uib.de (http://download.uib.de/vmware_pxeclient.zip). The advantage of vmware (virtuell hardware) is the support of the standard drivers from windows.

Now you have to create a client in the opsi system. Start the installation with a) the opsi-configed or b) the command line.

Graphic frontend of opsi-configed: Choose in the {opsi-configed}, mode *Configuration of clients* in the crystal tab *Clients* the client.

Choose the menu item *OpsiClient/Create new opsi client* and the description of the client:

- IP-name,
- DNS (Internet) domain,
- client description,
- IP-number (which is only requested by the internal DHCP) and
- MAC-address

The client will be created in the opsi database and (if so configured) at the same time as PXE-client at the DHCP configuration on the opsi-server.

Command line opsi-admin A opsi client may also be created at the command line:

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

e.g.:

```
opsi-admin -d method host_createOpsiClient testclient.domain.local "null" "Testclient" "" 00:0c:29:12:34:56 192.168.0.5
```

To see all created clients, choose in the {opsi-configed}, mode *Configuration of clients* the crystal tab *Clients* and reload the data by pressing *F5* or via context menu.

4.2.2 Hardware inventory with the netboot product hwinvent

Choose in the {opsi-configed}, mode *Configuration of clients* in the crystal tab *Clients* the client.

If haven't done yet, reload all data by clicking at the reload button at the top, left corner of the {opsi-configed} (or use the *File* menu).

Switch to the crystal tab *Netboot products*, look for the line of the wished system software (*hwinvent*). Go to the column *Requested Action* and select the action *setup*. Finally save with a click on the hook button (or with the context menu).

Now reboot the client (over PXE), the bootimage with *hwinvent* should be started. Therefore the client loads after the reboot a Linux boot image which scans the hardware and send the results back to the server.

You may see this data at the crystal tabs *Hardware information*.

4.2.3 Create a new client using the opsi-client-bootcd

At the opsi download site you will find in <http://download.uib.de/opsi4.0/> some ISO images of the opsi-client-bootcd. Just download the newest image and burn it to cdrom. Boot your computer from this CD. You should see the following image:

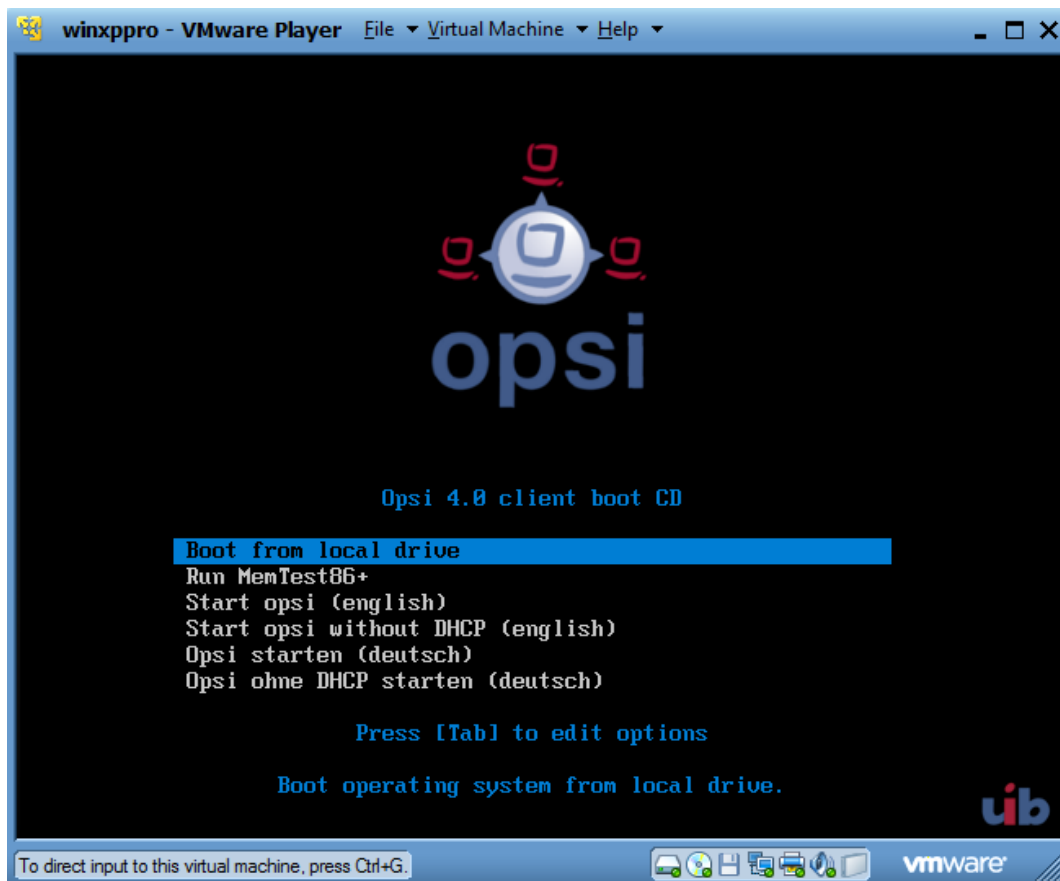


Figure 4.1: Start image opsi-client-boot-cd

Choose *Start opsi (English)*. After a while the following screen will appear. If your DHCP server give IP-numbers to unknown DHCP clients, most things will be completed. You have to complete the missing data. At least you must give the hostname.

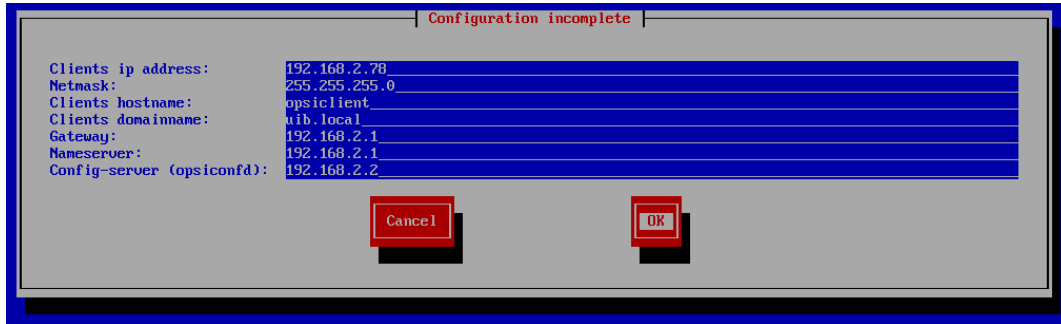


Figure 4.2: bootimage/boot-cd configuration screen

Confirm with *OK*.

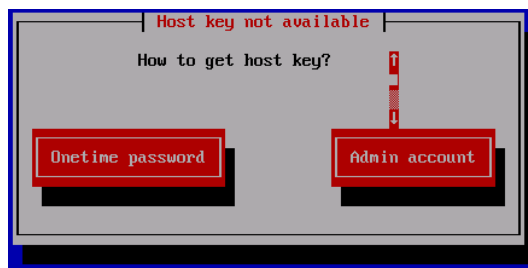


Figure 4.3: bootimage / boot-cd: Choose how to create Client

Choose *Admin account*. This means, the client should register himself at the opsi-server. This procedure must be authorized.

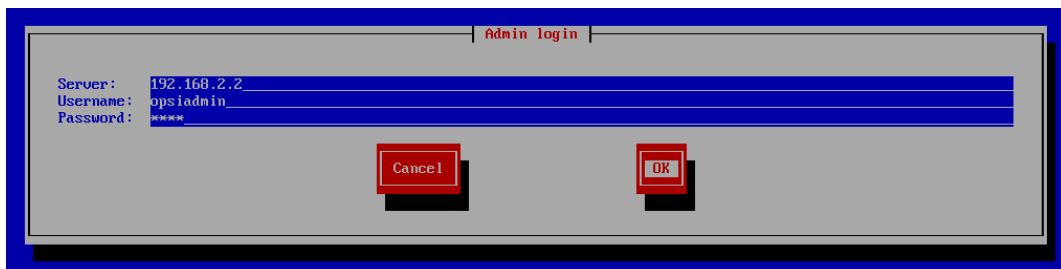


Figure 4.4: bootimage / boot-cd: Authenticate as member of opsi-admin group

Therefore you will get a login mask, where you should authenticate you as a member of the opsi-admin group. If you successful authenticate, the client give its data to the server and the client will be created at the server. In the next step the server gives the list of netboot products to the client.

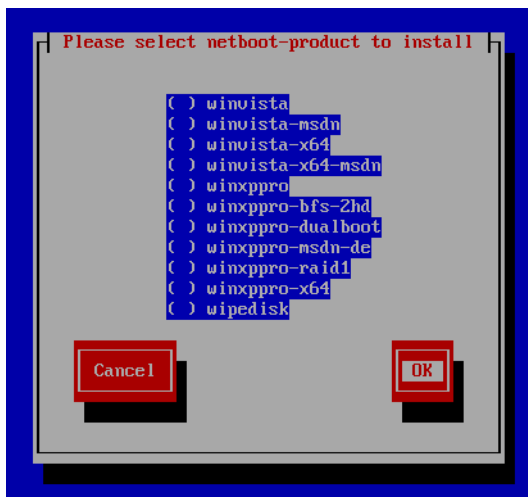


Figure 4.5: bootimage / boot-cd:: netboot product list

Now you may choose the operating system which you like to install (or e.g. hwinvent for testing).

4.2.4 OS-Installation: Complete the base package for Windows

The base package include only files for automatic system software installation – not the system software for his own. If you want to test the automatic Windows XP or Windows 7 system software installation, you have to complete these packages.

4.2.5 NT 5 family: XP, 2003

4.2.5.1 Copy the i386 directory

a) copy the i386-directory of an installation-CD for Microsoft Win2003/WinXP Professional in the directory /opt/pcbin/install/win2003 or /opt/pcbin/install/winxppro directory. When the file copy is done, you have to change the rights of the i386/ directory. Change to the winxppro or win2003 directory an fire the following command:

```
opsi-setup --set-rights i386
```

The files may also copied over the network. Therefore you have to connect the share `opt_pcbin` as user `pcpatch` the opsi-server. The corresponding directory you will find under `install\winxppro` or `install\win2003`.

4.2.6 NT 6 family: Vista / 2008 / Win7

Because these installations only starts from a Win32/Win64 environment we must build a PE-Image which is used to start up the installation.

"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."

<http://technet.microsoft.com/en-us/library/cc766093.aspx>

Therefore you need the Windows Automated Installation Kit (Windows AIK):

<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=696dd665-9f76-4177-a811-39c26d3b3b34>

What you get is a ISO file, which may be burned to CD or mounted by a virtual machine. The content of this CD must be installed in an OS mentioned in system requirements.

4.2.6.1 Creating a PE

Console commands for creating a Windows PE in 32- or 64-bit versions are nearly the same, except for the `<ARCH>` entries below. These have to be set to either `x86`, `amd64` or `ia64`.

- Creating an environment:
Start a terminal as administrator (Start ⇒ Programs ⇒ Accessories ⇒ right click on „Command Prompt“ ⇒ Run as... ⇒ Administrator) and run the following command:

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

- Prepare Image for opsi:
Start a terminal as administrator and run the following command (in one line):

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

- next command (again in one line):

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

(Remark: The file `startnet.cmd` will be created by the opsi linux boot image by executing the script `setup.py`. The `startnet.cmd` contains the call to `wpeinit`.)

- next command (again in one line):

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

- next command (again in one line):

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

- Copy the directory `C:\winpe\ISO` with the target name `winpe` to `/opt/pcbin/install/win7/` respectively `/opt/pcbin/install/win2008`.
Adjust the file access rights (e.g.):

```
opsi-setup --set-rights /opt/pcbin/install/win7/winpe
```

4.2.6.2 Extending a PE

In some cases it is useful to extend a PE. Especially by using Dell-Hardware. Dell provides special network and storage drivers for using in PE. These instruction only works with Windows 7. (Windows Vista don't inherits needed `dism - Deployment Image Servicing and Management`.) These instruction have the precondition, that you have followed the chapter "Creating a PE".

Note

Windows Automated Installation Kit is not needed for following instructions.

First you have to download Dell-PE-drivers from the Dell-Website. You need for Windows 7 the WINPE 3.0 Drivers from Dell. The downloaded CAB-File must be extracted to localdisk. These can be done with 7zip or the command-line tool expand. For reasons of clarity it is recommended to create a directory "dell-driver" on localdisk and extract the CAB-File in this directory.

- To determine the needed index number from the image you can scan the image. Normally a PE-image is a one-image-file, so you can use the index 1, but it is better to check these first. Start a terminal as administrator (Start ⇒ Programs ⇒ Accessories ⇒ right click on „Command Prompt“ ⇒ Run as... ⇒ Administrator) and run the following command:

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

In the output from these command you can see, which images are included in the imagefile.

- The next command mounts the image for modification:

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

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

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

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

Note

If only one driver have to be integrated, leave the option /Recurse away and point the driver-inf-File directly instead of the driver-directory. With the option /ForceUnsigned it is possible to integrate unsigned drivers to the image.

- At least the image must be unmounted and the changes have to be committed:

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

- Copy the directory C:\winpe\ISO with the target name winpe to /opt/pcbin/install/win7/ respectively /opt/pcbin/install/win2008.
Adjust the file access rights (e.g.):

```
opsi-setup --set-rights /opt/pcbin/install/win7/winpe
```

4.2.6.3 unattend.xml

The control file for the unattended Installation is the `unattend.xml` which you will find below /opt/pcbin/install/win7/custom. If you like to make any modifications at this file, do it in this directory and not in the opsi directory.

The `unattend.xml` delivered with the opsi package, contains the activating of the Administrator account with the password `nt123`.

Documentation to the `unattend.xml` you will find (after the installation of the WAIK) in the directory `c:\Program Files\Windows\Waik\docs\chms`.

4.2.6.4 Driver integration

The integration of drivers works at the usual in the opsi manual described way: Place your driver directories in `/opt/pcbin/install/win7/drivers/drivers` and then call the `create_driver_links.py` script.

Please keep in mind that Vista/Win7 only accept signed drivers. Therefore if you want to use driver packs like the driver packs from driverpacks.net, be sure to use only the vista/win7 versions.

4.2.6.5 Providing the installation files

Copy the complete installation DVD to `/opt/pcbin/install/win7/installfiles` Adjust the file access rights:

```
opsi-setup --set-rights /opt/pcbin/install/win7/installfiles
```

4.2.6.6 Log files of the installation

- `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 since specialize phase
- `c:\Windows\Panther\UnattendGC\setupact.err`:
Error log since specialize phase
- `c:\Windows\System32\winevt\Logs*`
- `c:\Windows\ntbtlog.txt` (only with activated startup protocol)

4.2.7 Windows product key

If you using the opsi license management module, you may administrate your Windows product keys by the license management. Information how to do this you will find at the opsi manual.

If you don't want to use the license management module, the product key can be provided by using product properties:

If you still have created a client you can use the opsi management interface to enter the product key:

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

A other possibility is to use the command line. With giving a opsiserver you will read / change the server default. To read the server default use (perhaps you must modify the productId and you must change `<opsiserver.domain.local>` with the fqdn from your opsiserver. Be sure that you write the following commands in one line):

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

To modify the defaults it's the easiest way to read the objects in a file, modify the file and update the objects with the modified file.

First read the actual configuration in a file (perhaps you must modify the productId and you must change `<op-
siserver.domain.local>` with the fqdn from your opsiserver. Be sure that you write the following commands in one line):

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

Now you must modify the file `/tmp/property_config.json` and change the entries in values. Finally you must update the objects with the modified file (command in one line):

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

The success of the modifications can be checked with following command (perhaps you must modify the productId and you must change `<op-
siserver.domain.local>` with the fqdn from your opsiserver. Be sure that you write the following commands in one line):

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

4.2.8 Start the Windows installation

To startup a windows installation:

- choose a client
- change to the tab *netboot products*
- select the product (e.g. winxppro)
- set the *action request* to setup
- save the changes by clicking the red hook (which changes to green)

Now the client should load the opsi linux bootimage via network and start it. Before the windows installations starts you have to confirm once.

4.2.9 Structure of the unattended installation products

This chapter describes the following products

- win2k
- winxppro
- winvista
- win2003
- win2008
- winvista-x64
- win2008-x64
- win7
- win7-x64
- win2008r2

4.2.9.1 Directory tree overview

```

<productid>-
  |-i386/                                NT5 only: Installations files
  |-installfiles/                        NT6/7 only: Installations files
  |-winpe/                                NT6/7 only
  |-opsi/                                  scripts and templates by opsi.org
  |   |-$oem$/                             $oem$ according to MS
  |   |-posinst.d/                          scripts after OS-install by opsi.org
  |   |-unattend.txt.template               Template by opsi.org
  |-custom/                                scripts and templates by customer
  |   |-$oem$/                             $oem$ according to MS by customer
  |   |-posinst.d/                          scripts after OS-install by customer
  |   |-unattend.txt                       unattend.txt by customer
  |-drivers/                               drivers directory
  |   |-drivers/                            drivers directory
  |   |-pciids/                             symbolic links to drivers
  |   |-vendors/                            symbolic links to drivers
  |   |-classes/                            symbolic links to drivers
  |   |-usbids/                              symbolic links to drivers
  |   |-hdaudioids/                         symbolic links to drivers
  |   |-pci.ids                             PCI-IDs DB
  |   |-usb.ids                             USB-IDs DB
  |-setup.py                               installation script
  |-<productid>_<version>.control           meta data (only for info)
  |-<productid>.files                       file list (created automatically)
  |-create_driver_links.py                 driver management script
  |-show_drivers.py                        driver management script
  |-download_driver_pack.py               driver management script
  |-extract_driver_pack.py                 driver management script

```

4.2.9.2 The files

- `setup.py`
This is the installation script which is executed by the boot image.
- `<productid>_<version>.control`
Contains the meta data of the product as prepared from the package maintainer. This files is here only for information purpose. Changes to this file will be with out any effect.
- `<productid>.files`
This file is created automatically and should not be changed.
- `create_driver_links.py`
`show_drivers.py`
`download_driver_pack.py`
`extract_driver_pack.py`
These are scripts for the simplified driver integration which is described in its own chapter ("[Simplified driver integration for the automatic OS installation](#)").

4.2.9.3 Directory i386 / installfiles / winpe

- `i386`
This directory contains the installation file of the `i386` directory of the windows installation CD (NT5 = Windows 2000 to XP). It is possible to have multiple `i386` directories (`i386` , `i386_en` , `i386_xxx`). Which `i386` directory is used for installation, is controlled by the product property `i386_dir`.
- `installfiles`
This directory contains the all files of the windows installation DVD (NT6 = Windows Vista and above).
- `winpe`
This directory contain at Windows Vista and above a bootable winpe image.

4.2.9.4 Directory opsi / custom

Both directories contain scripts and configuration files for the OS installation. While the installation process they working together with priority for the files of the custom directory.

The opsi directory contains files and templates that are maintained by opsi.org and maybe replaced by the next update. So it is no good idea to make custom specific changes at this place. Please use the custom directory for this purpose which is not subject of any changes by opsi.org.

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

4.2.9.5 Directory drivers

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

4.2.10 Simplified driver integration for the automatic OS installation

If some client hardware isn't supported by the standard Windows drivers, it could be useful and sometimes even necessary to integrate new drivers into the unattended installation. Regarding network devices this is very recommendable, because a client without network is neither remote accessible, nor can the automated software distribution connect to any distribution file shares.

opsi supports your work with a automated driver integration and detection. You only have to place the drivers in the correct directory. Calling a script (`create_driver_links.py`) a catalog is created which allows the boot image to find the correct drivers for the detected Hardware via PCI, USB or HD-Audio Identifier. Drivers for mass storage controllers (text mode drivers) may also integrated in the same way.

The server can provide additional drivers to be installed during Windows setup. All of these drivers must come with an `.inf` file, which holds the driver's installation information. Any drivers which are packed as an executable cannot be used for this (but often they can be unpacked to get the plain installation files).

If you have a running machine with the correct drivers installed, you may use the program *double driver* (<http://www.boozet.org/dd.htm>) to extract these drivers and save them to the opsi-server.

If you just have some differing hardware, you can take drivers as provided by the hardware manufacturer and put this to the distribution file share.

But if you have to support a lot of different hardware, it might be convenient to use ready packed packages for a whole bunch of Windows drivers as provided by <http://driverpacks.net/>.

You may download the current driverpacks (!!! about 2,5 GB !!!) from <http://driverpacks.net/DriverPacks/-overview.php> and save the to the server. Calling:

```
/opt/pcbin/install/winxpro/extract_driver_pack.py <path to the compressed driverpacks>
```

These driver packs will be decompressed and stored in the directory `winxpro/drivers/drivers/D`.

Structure of the drivers directory tree:

```
/opt/  
  !-pcbin/  
    !-install/  
      !-winxpro/  
        !-drivers  
          |-classes/           (links to drivers by device classes)  
          |-hdaudioids/       (links to HD-Audio drivers)  
          |-pciids/           (links to drivers via PCI identifier)
```

```
| -pci.ids          (PCI identifier data base)
| -usbids/         (links to drivers via USB identifier)
| -usb.ids         (USB identifier data base)
| -vendors/       (Links to drivers by vendors)
! -drivers         (space for general driver packs)
  | -additional/   (for manual assigned drivers)
  | -buildin/     (inf files from the i386 directory)
  | -preferred/   (preferred drivers for your hardware)
  | -exclude/     (excluded drivers)
  ! -mydriverpacks/ (example driver pack)
```

Additional drivers can be added, each into its own sub directory of `winxppro/drivers/drivers/preferred`.

After adding new drivers (or any other change) adjust rights with (e.g.):

```
opsi-setup --set-rights /opt/pcbin/install/winxppro/drivers
```

Then execute the script `create_driver_links.py` from the `winxppro` directory. The script searches all the directories beneath `drivers` and creates links to assign drivers to its hardware (using the PCI-IDs, USB-IDs, HD-Audio-IDs). Drivers found beneath the directory `preferred` will indeed be preferred against other ones with the same IDs beneath other directories. During installation, the script `setup.py` from the boot image identifies the client hardware and determines the required drivers. The drivers will be copied to the hard disk, and the file `unattend.txt` (which is the control file for the Windows unattended installation) will be patched accordingly.

The `create_driver_links.py` script also extracts once the driver information from the `i386` or `installfiles` directory to the `buildin` directory. If you make any changes to these directories (e.g by integrating a service pack) you should delete the `buil` directory and call `create_dirver_links.py` again.

If there is any hardware inventory data for a client, these data can be used to list the drivers that will be selected by the boot image for this client:

```
winxppro/show_drivers.py <clientname>
```

Additional drivers that should be used by the setup even if they were not selected via PCI-IDs should be placed beneath the directory `winxppro/drivers/drivers/additional`. Using the product property `additional_drivers` it is possible to give a comma separated list of driver paths beneath the additional directory which should be used by the windows setup program. The directories given in the property `additional_drivers` are searched recursively for drivers and all detected driver directories are integrated to the installation. You may use symbolic links in the additional drivers directory.

While driver integration at first all drivers which are assigned via the `additional` property will be integrated and analyzed for which PCI device (or HD-Audio, USB) there will be used. For these devices which are have an assigned driver via additional, no other driver `drivers/prefered/` or `drivers/` will be integrated. According to this behaviour `Additional` has not only the function to add drivers which are not detected. It has also the function to supersede other drivers (so `Additional` you may also call `super-preferred`).

You should examine the output of `show_drivers.py` and check which drivers will be integrated.

It may be that driver directories from hardware vendors contain drivers for different operating systems versions (Vista/Win7) or different configurations (SATA / SATA-Raid). The `create_driver_links.py` script will take the first directory that matches to the PCI-Id (or HD-Audio, USB). If you are suppose that the link goes to the wrong driver directory, so move this driver directory to `drivers/exclude` and run `create_driver_links.py` again. Drivers in `drivers/exclude` are excluded from the driver integration.

Example of the output of `show_drivers.py` :

```
./show_drivers.py pcdummy

PCI-Devices
[(Standardsystemgeräte), PCI Standard-PCI-zu-PCI-Brücke]
  No driver - device directory /opt/pcbin/install/winxppro/drivers/pciids/1022/9602 not found
[ATI Technologies Inc., Rage Fury Pro (Microsoft Corporation)]
  Using build-in windows driver
[(Standard-IDE-ATA/ATAPI-Controller), Standard-Zweikanal-PCI-IDE-Controller]
```

```
/opt/pcbin/install/winxpro/drivers/drivers/D/M/N/123  
[Realtek Semiconductor Corp., Realtek RTL8168C(P)/8111C(P) PCI-E Gigabit Ethernet NIC]  
/opt/pcbin/install/winxpro/drivers/drivers/preferred/realtek_gigabit_net_8111_8168b  
[IEEE 1394 OHCI-konformer Hostcontroller-Hersteller, OHCI-konformer IEEE 1394-Hostcontroller]  
No driver - device directory '/opt/pcbin/install/winxpro/drivers/pciids/197B/2380' not found  
[Advanced Micro Devices, Inc., AMD AHCI Compatible RAID Controller]  
/opt/pcbin/install/winxpro/drivers/drivers/preferred/ati_raid_sb7xx  
[(Standard-USB-Hostcontroller), Standard OpenHCD USB-Hostcontroller]  
No driver - device directory '/opt/pcbin/install/winxpro/drivers/pciids/1002/4397' not found  
[ATI Technologies Inc, ATI SMBus]  
/opt/pcbin/install/winxpro/drivers/drivers/preferred/ati_smbus
```

USB-Devices

```
[(Standard-USB-Hostcontroller), USB-Verbundgerät]  
/opt/pcbin/install/winxpro/drivers/drivers/preferred/brother_844x_pGerb  
[Microsoft, USB-Druckerunterstützung]  
/opt/pcbin/install/winxpro/drivers/drivers/preferred/brother_844x_pGerb
```

Additional drivers

```
[ati_hdaudio_azalia]  
/opt/pcbin/install/winxpro/drivers/drivers/additional/ati_hdaudio_azalia
```

Chapter 5

Integration of new software packets into the opsi software deployment.

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. In that way neither the user can interfere with nor the user is affected by a software installation process.

In order to do this you have to write a script for the script driven installer opsi-winst.

5.1 A small tutorial: How to write a opsi-winst script

5.1.1 Introduction

This tutorial should help you to start with opsi. It is no replacement for professional training (which you may order by uib) or even the study of the complete manuals.

The opsi Manuals you will find at:

<http://download.uib.de/docu> http://download.uib.de/opsi_stable/docu

Most important: Winst reference card and Winst manual

Wiki (Scripte, Tips, Links):

http://www.opsi.org/opsi_wiki/OpsiWikiPage

Support Forum:

<http://forum.opsi.org>

Training and Support:

Get Training by uib gmbh or an opsi partner:

<http://www.opsi.org/support/>

5.1.2 Methods of not interactive installation

Regardless if you are using opsi or a other product, there are three ways to install a software without user interaction:

1. Unattended or Silent Installation

Existing setup programs from the original software manufacturer can be executed from within a opsi-winst script in *silent* or *unattended* mode. It depends on the setup program whether silent installation mode is supported. A special case of this method is the unattended installation of MSI packages.

2. Interactive Setup with recorded Answers

The interactive answers required by the original setup program can be given automatically by using the free tool *AutoIt* or *Autohotkey*. That means providing an *autoIt* script for unattended installation

3. Analyze and Repackaging

The standard setup can be analyzed and *recorded* to do the installation tasks directly by the *opsi-winst* program. Usually that is something like file installation to the local file system and patching the registry

Note

opsi supports all of these variants.

Usually a combination of all different ways in one script does the job best. Like doing the basic installation by the original setup if available and then do some customizing by patching registry or file based configuration.

5.1.3 Structure of a winst script

At first a simple example of a winst script:

```
[Actions]
WinBatch_tightvnc_silent_install

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

A winst script contains **primary** and **secondary** sections. The section header are in square brackets like you know it from ini-files. The primary section is here [Actions] and the secondary section is here [winbatch_...].

The core work like starting programs or copying files is not done in the primary sections but in the secondary sections. These secondary sections are topic specific and have an specific syntax according to their specific topic.

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

At this example the primary section [Actions] calls a secondary section [WinBatch_tightvnc_silent_install]. These secondary section has the type *WinBatch*. The content of secondary sections from type *winbatch* are executed by the windows API. In this case the program *tightvnc-1.3.9-setup.exe* will be started with the parameter */silent*.

5.1.4 Primary sections

Initial

The Initial section is used to set runtime parameters.

This section is optional.

Actions

The section [Actions] is the core main program.

Parts of the code which are called more then one time can be written in sub sections.

Sub-sections

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

The primary sections are the main program which control the program flow. Therefor you have:

- Variables: strings and string lists
- if else endif statements
- for loops through string lists

- Functions

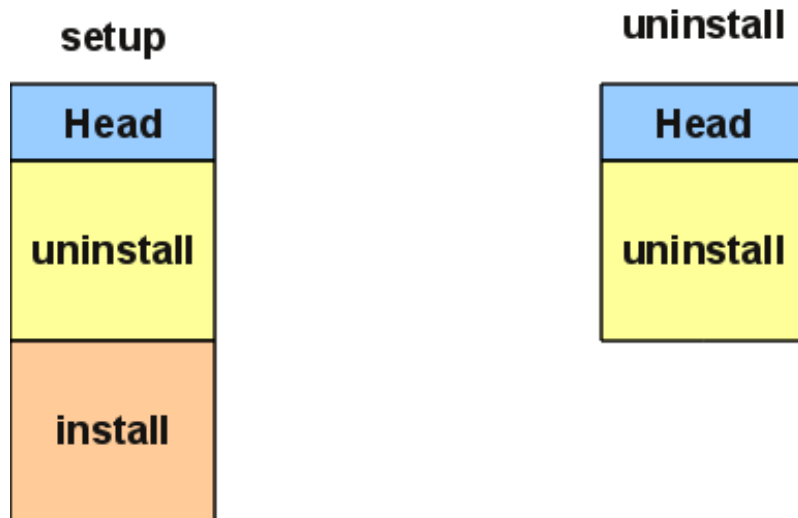


Figure 5.1: double code for deinstallation

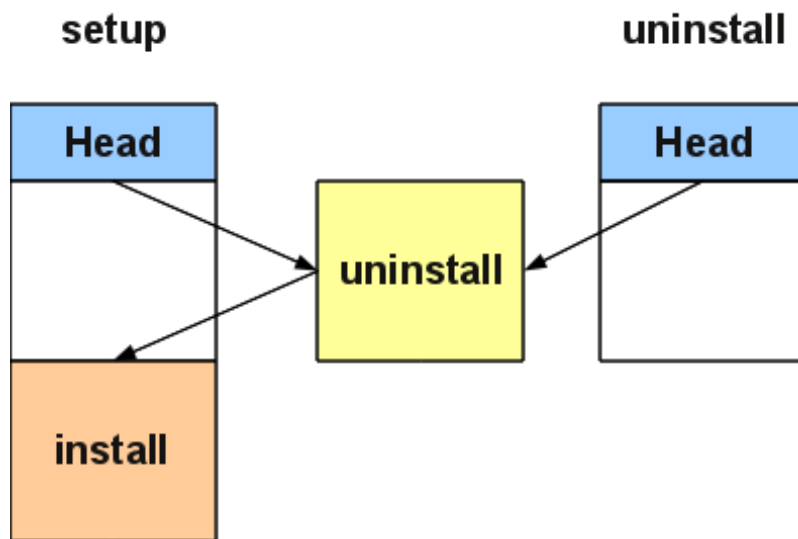


Figure 5.2: avoid double code by using sub sections

5.1.5 Important kinds of secondary sections

Files

File operations like

- copy (regarding the internal version information, recursive, ...)
- delete files or directories
- create directories

WinBatch

Is used for calling programs via Windows API. E.g. for calling setup programs in the silent mode.

DosBatch/DosInAnIcon

The content of these sections is interpreted by the `cmd.exe` like normal batch files.

A variant of *DosBatch* is *DosInAnIcon* which is called in a minimized window.

ExecWith

The content of this section is interpreted by the program which is given as parameter to this section (e.g. `AutoIt`).

Registry

The *Registry* sections are used for registry manipulations.

Linkfolder

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

5.1.6 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>

5.1.7 Second example: tightvnc

For explanation purpose a simple script for a tightvnc installation. This script should be only contain the winbatch call for the silent installation. But if you call this silent installation more the one time it appears a confirm window (which is a bug in the installer). This confirm 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")
```

5.1.8 Elementary commands for primary sections

5.1.8.1 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 are different in primary and secondary sections. In primary section string variables are handled as independent objects. String variables can only be declared and set to values in primary sections. Therefore you have to use a operator (+) to concatenate variables and strings in a string expression.

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

In secondary sections string variables are used as a placeholder for their values.

Example: "Installing \$ProductId\$..."

You should keep this in mind if you copy and paste string expressions between primary and secondary sections.

The advantage of this kind of handling string variables is that is possible to use these variables in secondary sections which are interpreted by other programs (DosBatch / Execwith).

5.1.8.2 Message / showbitmap

Displaying text while runtime:

Message <string>

Example:

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

Display a picture while installation:

ShowBitmap [<file name>] [<sub titel>]

Example:

```
ShowBitmap "%ScriptPath%\python.png" "Python"
```

5.1.8.3 if [else] endif

Syntax:

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

5.1.8.4 Functions

HasMinimumSpace

Check for free space on the hard disk.

FileExists

Check for the existence of a file or directory

5.1.8.5 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

abort the script interpretation and returns the installation state *failed* to the server.

5.1.8.6 Requirements

requiredWinstVersion

Minimum required version of opsi-winst

5.1.9 Third example: The generic template *opsi-template*

You encouraged to use this template when ever you create a own opsi product. Do not use a copy from this manual but 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 this package (-i) or extract (-x) at your server and then grab the scripts.

setup.ins: installation script

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

[Actions]

```
requiredWinstVersion >= "4.10.5"
```

```
DefVar $MsiId$
```

```
DefVar $UninstallProgram$
```

```
DefVar $LogDir$
```

```
DefVar $ProductId$
```

```
DefVar $MinimumSpace$
```

```
DefVar $InstallDir$
```

```
DefVar $ExitCode$
```

```
DefVar $LicenseRequired$
```

```
DefVar $LicenseKey$
```

```
DefVar $LicensePool$
```

```
Set $LogDir$ = "%SystemDrive%\tmp"
```

```

; -----
; - Please edit the following values -
; -----
Set $ProductId$      = "opsi-template"
Set $MinimumSpace$  = "1 MB"
; the path were we find the product after the installation
Set $InstallDir$    = "%ProgramFilesDir%\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
    ; Stop process and set installation status to failed
else
    comment "Show product picture"
    ShowBitmap "%ScriptPath%" + $ProductId$ + ".png" $ProductId$

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

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

    if $LicenseRequired$ = "true"
        comment "Licensing required, reserve license and get license key"
        Sub_get_licensekey
    endif

    comment "Start setup program"
    Winbatch_install
    Sub_check_exitcode

    comment "Copy files"
    Files_install

    comment "Patch Registry"
    Registry_install

    comment "Create shortcuts"
    LinkFolder_install

    ; comment "Test for installation success"
    ; Test if software marked as installed in registry
    ; if (GetRegistryStringValue("[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\Uninstall\{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}] DisplayName") = "")
    ;     logError "Fatal: After Installation [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\
Windows\CurrentVersion\Uninstall\{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}] not found"
    ;     isFatalError
    ; else
    ;     comment "Successful Installation"
    ; endif

```

```
endif

[Winbatch_install]
; Choose one of the following examples as basis for your installation
; You can use $LicenseKey$ var to pass a license key to the installer
;
; === Nullsoft Scriptable Install System =====
; "%ScriptPath%\Setup.exe" /S
;
; === MSI package =====
; You may use the parameter PIDKEY=$Licensekey$
; msixexec /i "%ScriptPath%\some.msi" /l* "$LogDir$\$ProductId$.install_log.txt" /qb! ALLUSERS=2
; REBOOT=ReallySuppress
;
; === InstallShield + MSI=====
; Attention: The path to the log file should not contain any whitespaces
; "%ScriptPath%\setup.exe" /s /v" /l* $LogDir$\$ProductId$.install_log.txt /qb! ALLUSERS=2 REBOOT
; =ReallySuppress"
; "%ScriptPath%\setup.exe" /s /v" /qb! ALLUSERS=2 REBOOT=ReallySuppress"
;
; === InstallShield =====
; Create setup.iss answer file by running: setup.exe /r /f1"c:\setup.iss"
; "%ScriptPath%\setup.exe" /s /sms /f1"%ScriptPath%\setup.iss" /f2"$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

[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: $NewExe$
; parameters:
; working_dir: $InstallDir$
; icon_file:
; icon_index:
; end_link
;
; Example of creating a shortcut to the installed exe on AllUsers desktop:
;
; set_basefolder common_desktopdirectory
; set_subfolder ""
;
; set_link
; name: $ProductId$
; target: $NewExe$
; parameters: /some_param
; working_dir: $InstallDir$
; icon_file: $NewExe$
; icon_index: 2
; end_link

[Sub_get_licensekey]
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
endif
```

```
[Sub_check_exitcode]
comment "Test for installation success via exit code"
set $ExitCode$ = getLastExitCode
; informations to exit codes see
; http://msdn.microsoft.com/en-us/library/aa372835(VS.85).aspx
; http://msdn.microsoft.com/en-us/library/aa368542.aspx
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_UNKOWN_PRODUCT 1605 This action is only valid for products"
        comment "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"
                comment "complete the install. This message is indicative of a
success."
            else
                logError "Fatal: Setup program gives an unknown exitcode unequal
zero: "
                logError $ExitCode$
                isFatalError
            endif
        endif
    endif
endif
endif
```

delsub.ins: 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/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 (GetRegistryStringValue(" [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

comment "Cleanup registry"
Registry_uninstall

comment "Delete program shortcuts"
LinkFolder_uninstall

[Winbatch_uninstall]
; Choose one of the following examples as basis for program uninstall
;
; === Nullsoft Scriptable Install System =====
; "$UninstallProgram$" /S
;
; === Inno Setup =====
; "$UninstallProgram$" /silent /norestart

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

[Files_uninstall]
; Example for recursively deleting the installation directory
; (do not forget the trailing backslash):
;
; delete -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]
;(... siehe oben .....)

```

uninstall.ins: deinstallation script

```
; 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/credits/
```

```
[Actions]
```

```
requiredWinstVersion >= "4.10.5"
```

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

```
Set $LogDir$ = "%SystemDrive%\tmp"
```

```
; -----  
; - Please edit the following values -  
; -----
```

```
Set $ProductId$ = "opsi-template"  
Set $InstallDir$ = "%ProgramFilesDir%\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%\delsub.ins")
```

```
    comment "Start uninstall sub section"
```

```
    Sub "%ScriptPath%\delsub.ins"
```

```
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("", "", $WindowsSoftwareId$)
```

5.1.10 Interactive creation and testing of a opsi-winst script

You may interactive adapt and test a script.

Therefore create a directory (e.g. `c:\test`) and copy the scripts from the opsi-template (`setup.ins`, `delsub.ins` und `uninstall.ins`) to this directory.

Start the opsi-winst (`winst32.exe`) via double click. (On Windows 7 Clients you must click on the right mouse button and use "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\opt_pcb\bin' in the directory `install\opsi-winst\files`.

After starting the opsi-winst you will see the following window:

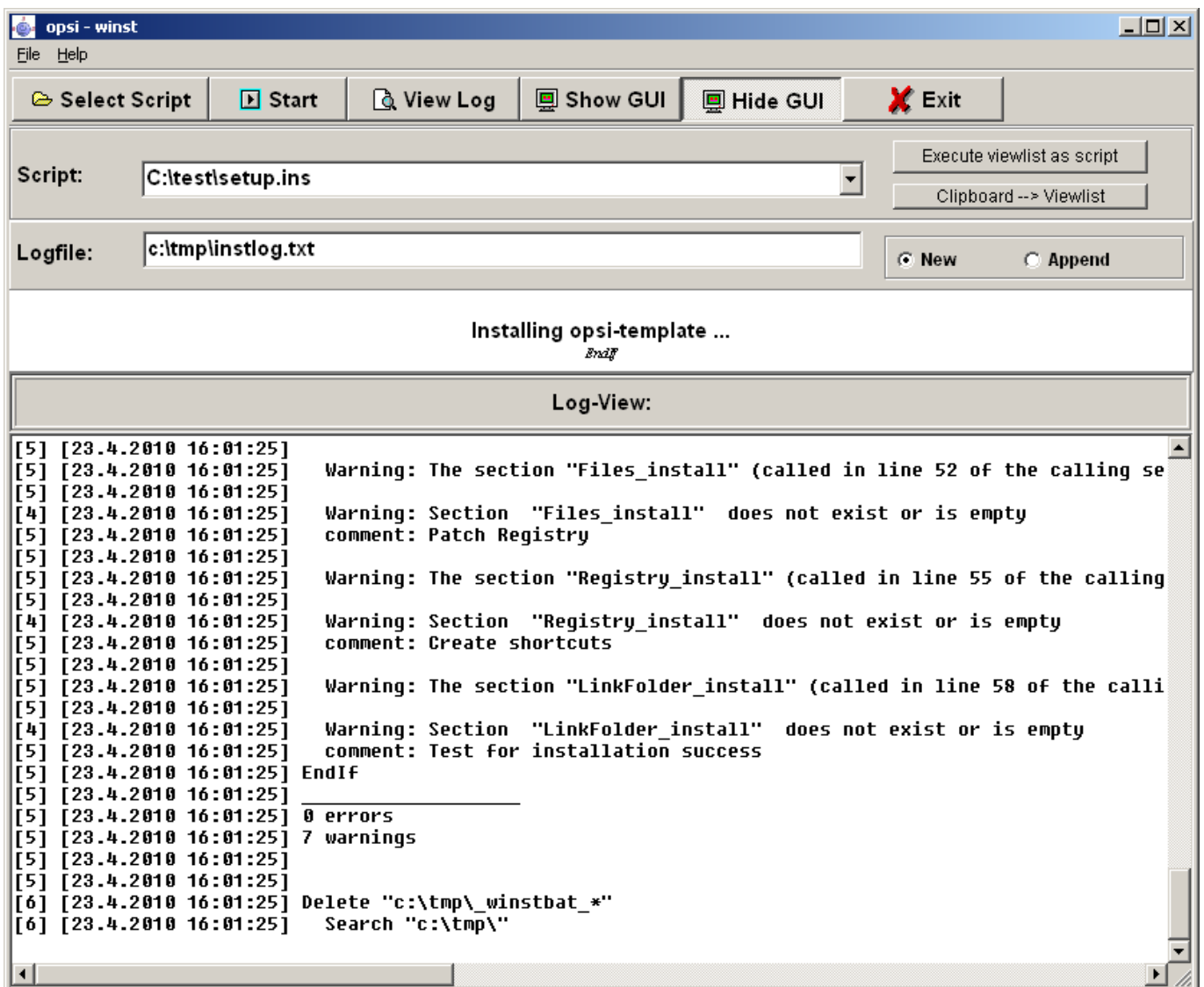


Figure 5.3: opsi-Winst started in interactive mode

- *Select Script* is used to choose the script that you want to execute.

- *Start* will start the execution of the selected script.
- *View Log* is used to read the log file from the last script run.

Select the *setup.ins* script and run it.

```

wInst Logfile
[1] [23.4.2010 16:06:22]
[1] [23.4.2010 16:06:22] ===== Version 4.10.5.0 WIN32 script "C:\test\setup.ins"
[1] [23.4.2010 16:06:22]      start: 2010-04-23 16:06:22
[1] [23.4.2010 16:06:22]      on client named "PCBON4"
[1] [23.4.2010 16:06:22]      user account "oertel"
[1] [23.4.2010 16:06:22] [executing: "C:\Programm\opsi.org\preloginloader\opsi-winst\winst32.exe"]
[1] [23.4.2010 16:06:22] system infos:
[1] [23.4.2010 16:06:22] 00:50:56:C0:00:08 - PC hardware address
[1] [23.4.2010 16:06:22] pcbon4 - IP name
[1] [23.4.2010 16:06:22] 192.168.2.234 - IP address
[1] [23.4.2010 16:06:22] DEU - System default locale
[1] [23.4.2010 16:06:22]
[6] [23.4.2010 16:06:23] wInst has version 4.10.5.0, required is : >= 4.10.5
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $LogDir$ = "C:\tmp"
[6] [23.4.2010 16:06:23] The value of the variable is now: "C:\tmp"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $ProductId$ = "opsi-template"
[6] [23.4.2010 16:06:23] The value of the variable is now: "opsi-template"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $MinimumSpace$ = "1 MB"
[6] [23.4.2010 16:06:23] The value of the variable is now: "1 MB"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $InstallDir$ = "C:\Programm\path to the product"
[6] [23.4.2010 16:06:23] The value of the variable is now: "C:\Programm\path to the product"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $LicenseRequired$ = "false"
[6] [23.4.2010 16:06:23] The value of the variable is now: "false"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Set $LicensePool$ = "p_" + $ProductId$
[6] [23.4.2010 16:06:23] The value of the variable is now: "p_opsi-template"
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] If
[6] [23.4.2010 16:06:23] Free on Disk C: : 456.754.891.264 bytes This is more than the required amount of 1.000.000 bytes
[5] [23.4.2010 16:06:23] HasMinimumSpace ("C:", $MinimumSpace$) <<< result true
[5] [23.4.2010 16:06:23] not(HasMinimumSpace ("C:", $MinimumSpace$)) <<< result false
[5] [23.4.2010 16:06:23] Then
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23] Else
[5] [23.4.2010 16:06:23] comment: Show product picture
[5] [23.4.2010 16:06:23]
[5] [23.4.2010 16:06:23]

```

Figure 5.4: opsi-winst log view window

- Look at the log file and see how opsi-winst interpret the script.
- Copy the setup.exe which you want to install to the directory where the scripts are (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 high lightning for opsi-winst which is part of the essential opsi-products.

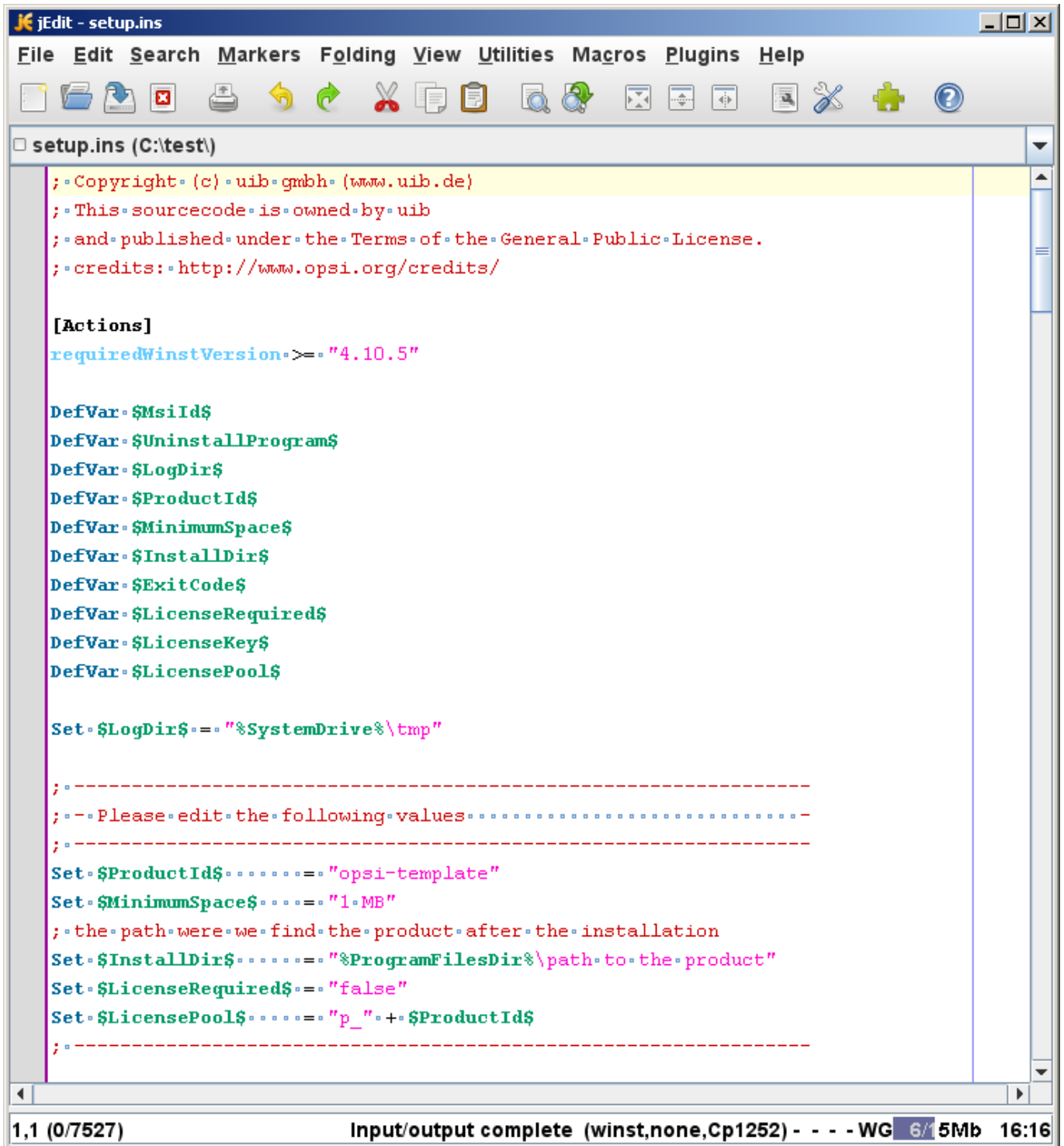


Figure 5.5: jEdit with a opsi script

- You may now change the script using the editor. Save the changes (keep the editor open).
- Now switch to the opsi-winst and start the script again. (You don't have to reselect the script. Just press the *start* button).
- Just have a look at the log again and see how the program flow changed according to your script changes.

- In this way, looping through the steps:
 - Change the script and save
 - run the script
 - review the logYou are able to develop step by step the script until it fit to your needs.

Some hints to detail problems you will find at the next chapter. In the chapter after next is described how to create a opsi-product from your scripts and files and how to install it on the opsi-server.

5.1.11 Hints to detail problems

5.1.11.1 search unattend or silent switches

For an „unattended“ or „silent“ setup the original setup will be switched to an unattended non interactive mode by applicable command line arguments.

The problem is to find the correct switch

Look at the internet Before you start to integrate a new package, you'd better first have a look at internet whether somebody already did that job:

Ready to run opsi-winst scripts from the community you will find at:
<https://forum.opsi.org/wiki/>

A collection of links to web sites with switch collections is
<http://www.opsi.org/en/software-integration-web-links>

Search the software producers site A lot of software manufacturers are aware of the needs of unattended software distribution, so there often are some hints and instructions in the product documentation or on the software producers website.

Search the setup tool manufacturers site Often setup programs are not written by the software manufacturers themselves. In most cases they deploy dedicated software products for creating setup programs. So the command line arguments for the resulting packet often are typical for the setup creation tool it is based on.

You will find at opsi.org <http://www.opsi.org/en/software-integration-web-links>

+ at the section *Installer specific switches* more web links to sites which give hints how to detect the manufacturer of the setup program.

5.1.11.2 Some more important opsi-winst commands

A short overview to the opsi-winst commands are the reference card: <http://download.uib.de/opsi4.0/doc/opsi-winst-reference-card.pdf>

All syntax details are described in the opsi-winst manual: <http://download.uib.de/opsi4.0/doc/winstdoc-en.pdf>

Here are some hints to important elements:

Stringlisten String lists are powerful, specially 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 have product properties which can modify the installations client specific. How you create these properties is described below in "[Creating an opsi package](#)".

To evaluate these properties opsi-winst got the function `GetProductProperty`

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

5.1.11.3 Installation with a logged on user

As a starting point we assume that you have done an unattended installation by using a opsi-winst script. The installation works OK when started as a logged on user (with administrative privileges). But with some software products you will see that when started from within the software deployment (opsi-client-agent) it fails. A possible reason for that difference might be that the installation process requires an user environment or profile.

In case of a MSI package the option `ALLUSERS=2` might help. Example:

```
[Actions]  
DefVar $MsiLogFile$  
Set $MsiLogFile$ = "c:\tmp\myproduct.log"  
winbatch_install_myproduct  
  
[winbatch_install_myproduct]  
msiexec /qb /l* $MsiLogFile$ /i "%ScriptPath%\files\myproduct.msi" ALLUSERS=2
```

A other 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 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>"` calling the winbatch section to wait for end of the second process.

Another more complex way to solve the problem is to create a temporary administrative user account and use this for the program installation. For a detailed description 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*.

5.1.11.4 Work with MSI-packages

With Windows 2000 Microsoft launched its own installation concept based on the Microsoft Installer Service „MSI“. In the meantime many setup programs are MSI compliant.

To be MSI compliant means to provide a packet with install 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 to ask for the installation's start. If this has been approved the control program checks whether MSI is installed and passes *product.msi* over. If there is no suitable MSI, the control program first starts the installation of the MSI.

When you interrupt the installation at that point, you often find the unpacked MSI-package in a temporary directory.

This package can be used for unattended installation for instance with the statement:

```
msiexec /i "%ScriptPath%\Product.msi" /qb-! ALLUSERS=2 REBOOT=ReallySuppress
```

5.1.11.5 Customizing after a silent/unattended installation

After a successful silent installation some more customizing might be useful. The opsi-winst is a powerful tool to do that job. At first you will have to find out what patches have to be applied. For example that could mean to analyze what registry settings are affected by the GUI customizing tools.

You can use the tools portrayed in Section 5.1.11.7 further down. Some more tools can be found here:

<http://www.sysinternals.com/>
<http://www.german-nlite.de/files/guides/regshot/regshot.html>

5.1.11.6 Integration with automated answers for the setup program

Another fast way of integration is to provide an automated answer file for the setup process. To be more precise, the answer file is used by a control tool, which waits for the setup to come up with interactive windows and 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 you will find at the website: <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.

Although there is 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 waiting for input. This input could be done by an interactive user and then the script can take over again and handles the next well known windows.

There is another critical path 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 possibility in opsi to run the installations on a other desktop than the current desktop or if the current desktop is locked you will find some autoit functions which do not work properly under these conditions.

Therefore you should avoid the use of the following autoit commands in *opsi-winst* scripts:

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

Because these commands are widely used, we need substitutes:

winwait()

may be replaced by

winexists()

in the following way (with the additional functionality of 30 seconds timeout):

```
$seconds = 0
$exists = 0
While ($seconds <= 30) and ($exists = 0)
    $exists = WinExists("Setup")
    $seconds = $seconds + 1
    sleep(1000)
WEnd
```

Send()

should be replaced by

controlsend() bzw. controlclick()

Therefore you should use the *Au3info.exe* to get the *ControlId* you need in this commands. Please use the numerical *ControlId*, the other variants seems to work not properly:

```
ControlClick("Setup", "", 1)
```

Here a part of our jedit installation script as example.

In this script we produce also a log file from the autoit activities, which may be integrated in the *opsi-winst* log file by the following commands:

```
setloglevel=7
set $resultlist$ = loadTextFile("c:\tmp\au3.log")
setloglevel=6
```

The example:

```
[ExecWith_autoit_confirm]
; exact title match
Opt("WinTitleMatchMode", 3)
$mylog = FileOpen("C:\tmp\au3.log", 2)
FileWriteLine($mylog,"auto-it started - waiting for the window")
; do not use winwait(), it will fail on other desktops than current
$seconds = 0
$exists = 0
While ($seconds <= 30) and ($exists = 0)
    $exists = WinExists("Setup")
    FileWriteLine($mylog,"winexists result (1=exists): " & $exists )
    $seconds = $seconds + 1
    sleep(1000)
WEnd
FileWriteLine($mylog,"window detected - sending answer")
; say no
$result = ControlClick("Setup", "", 1)
FileWriteLine($mylog,"answer sended: result (1=success) : " & $result)
FileClose($mylog)
Sleep(500)
;and good bye
Exit
```

see also:

http://www.autoitscript.com/wiki/FAQ#Why_doesn.27t_my_script_work_on_a_locked_workstation.3F

<http://www.autoitscript.com/autoit3/docs/>

<http://www.autoitscript.com/autoit3/docs/intro/controls.htm>

<http://www.autoitscript.com/autoit3/docs/functions.htm>

5.1.11.7 Analyze and repackage

When a software developer builds a setup for deployment, he usually knows about the required components of the software that have to be installed. But if somebody just has got the setup as a black box, he first needs to analyze what the setup does. This can be done by monitoring the setup activities with appropriate tools (e.g. monitoring any file and registry access) or by comparing the system states before and after installation.

To analyze the before / after states, there are a lot of tools. For Example:

- *WinINSTALL LE* which is be available as freeware from:
<http://www.ondemandsoftware.com>
- *InstallWatch Pro*
<http://installwatch-pro.software.informer.com/>

5.1.11.8 How to deinstall products

To deinstall a software product from a computer, you need an *uninstall* script to perform the deletion. The fundamental difficulty in software deletion is to distinguish what exactly has to be removed. Not all of the files that came with a

software package can be deleted afterward. Sometimes a packet comes with standard modules which 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 deinstall option which can be embedded in the opsi deinstall script. Otherwise opsi-winst provides several commands for software deletion:

Using an uninstall routine If the product manufacturer provides an option for software deletion, it has to be checked whether it can be run unattended (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, it always should be tested whether all of the files have been deleted and the computer is still in a stable state.

Products which are installed by MSI often come also with an uninstall option, which usually is the `msiexec.exe` parameter `/x`. And the parameter `/qb-!` is for unattended mode (without user interaction). So this is the statement for unattended deinstall:

```
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 and can be found in the registry directory `HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall`

A request using the GUID looks like this:

```
msiexec.exe /x {003C5074-EB37-4A75-AC4B-F5394E08B4DD} /qb-!
```

If none of these methods is available or sufficient, the deinstallation can be done by a opsi-winst script as described in the following:

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 deinstallation has to be done by a opsi-winst script. opsi-winst comes with some powerful uninstall functions. In this chapter we will have an overview, for detailed information refer to the opsi-winst handbook.

The base of deletion is deleting one or more files from the file system. This command can be executed from a opsi-winst files section:

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

or to delete a directory including sub directories:

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

The parameter *f* means *force* – to delete the files anyway, 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 by using the option `/AllNTUserProfiles` (see opsi-winst manual for details).

Directories containing files with the attribute *hidden* or *system* can be deleted by using a *DosInAnIcon*-section:

```
[DosInAnIcon_deleteDir]
rmdir /S /Q "<Verzeichnis>"
```

To stop a running process before deletion use the 'killtask' command with the process' name (look at the task manager for process name):

```
KillTask "thunderbird.exe"
```

If the product – or part of it – runs as a service, you will have to stop the service before deleting the files. One way to do so, is to set the service to state “inactive“ in the registry and restart the computer. Or to stop the service by using the command *net stop*, which doesn’t need a reboot:

```
net stop <servicename>
```

Also deleting DLL files requires special attention, since DLLs could also be used by other products. There is no general concept for handling this.

To delete registry entries with the opsi-winst you can use the command `DeleteVar`. This command deletes entries from the currently open key:

```
DeleteVar <VarName>
```

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

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

5.1.11.9 Known issues at the 64 Bit support

The opsi installer opsi-winst is a 32 bit program. There is no known problem 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-winst manual (<http://download.uib.de/opsi4.0/doc/winstdoc-en.pdf>) for these issues.

5.2 Creating an opsi package

opsi has a package format which contains the installation files, the opsi-winst installation script and meta data.

The essential advantages of this format are:

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

The packet 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 (`/opt/pcbin/install/<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.

5.2.1 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 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-makeproductfile`
- `opsi-package-manager -i <{opsi-product}-file>`

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

You should create products in the directory `/home/opsiproducts`. This directory is also available as share *opsi_worbench*. 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*).

5.2.1.1 Create with opsi-newprod



Warning

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

Change directory to the product directory and start the creation of the new product with `opsi-newprod`. The next question is for the type of product to create. Choose type *localboot* for products which should be installable by *opsi-client-agent/opsi-winst*. Product type *netboot* is used for products which are activated as a bootimage (like OS installation)

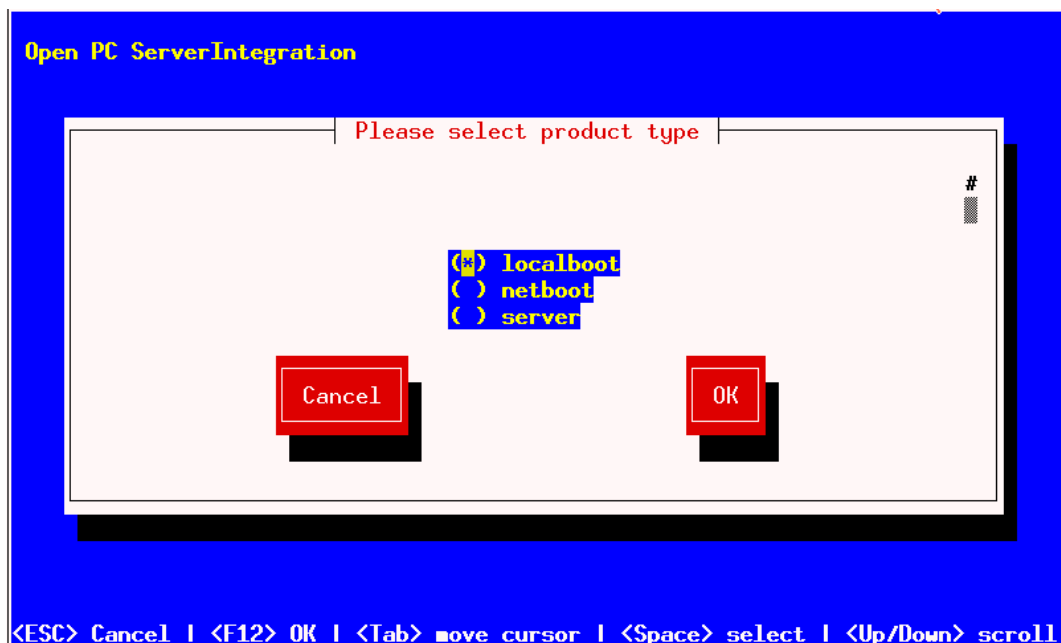


Figure 5.6: Choose the product type: localboot

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

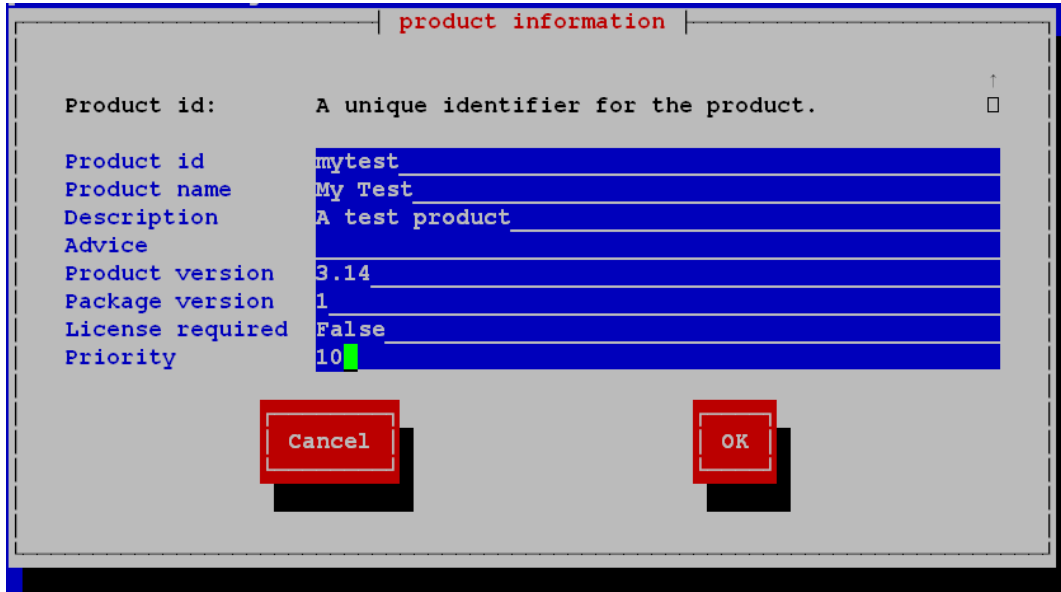


Figure 5.7: Input of the product information

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 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. This helps to differ packages with the same product version but with for instance a modified *opsi-winst* script.

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 to the installation sequence. See the opsi manual for more details.

After the product information is completed, fill in which action scripts should be provided:

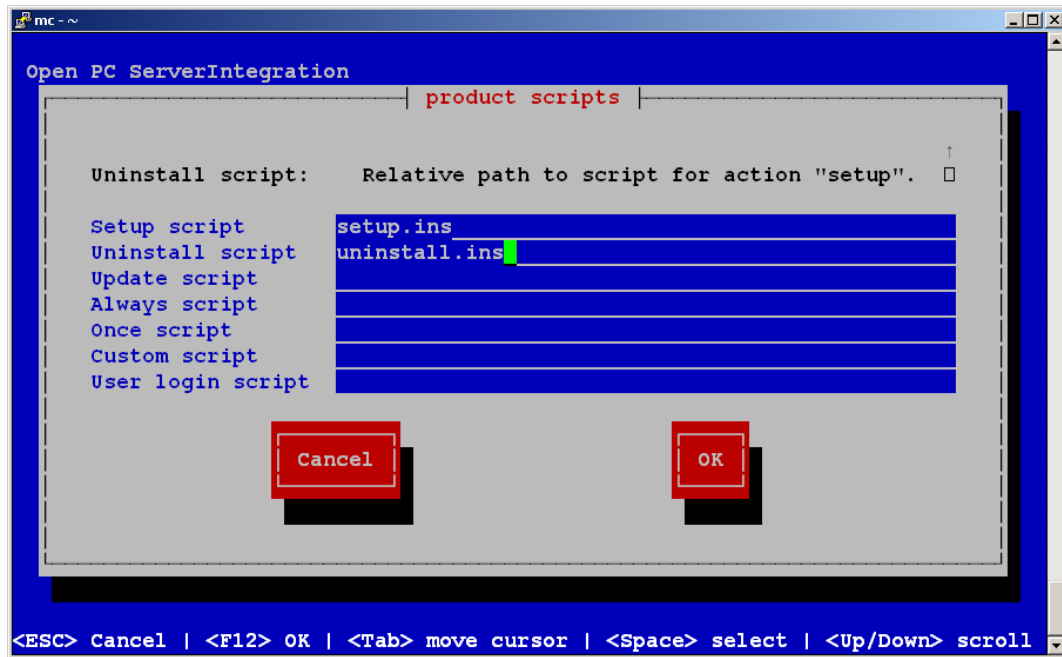


Figure 5.8: Input of the opsi-winst script names for different actions

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`, the update script will be executed after the setup script automatically.

An **Always-Script** will be executed on every activity start of the `opsi-client-agent` (e.g. on every boot).

An **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 doing.

A **Custom-Script** doesn't changes resulting state or resulting state. It is a very special kind of script and you should only use it, if you really know what you doing.

A **userLoginScript** is used to modify the users profile after the user loggedin to the system. It works only with the opsi extension *Roaming Profile Support* which is described at the *Roaming Profile Support* chapter in the opsi-manual.

Typ	resulting state	resulting action
setup	installed	none
uninstall	not_installed	none
update	installed	none
always	installed	always
once	not_installed	none
custom	<i>unchanged</i>	<i>unchanged</i>
User login	<i>unchanged</i>	<i>unchanged</i>

The next step is to define one or more product dependencies. If there are no product dependencies put in *No*.

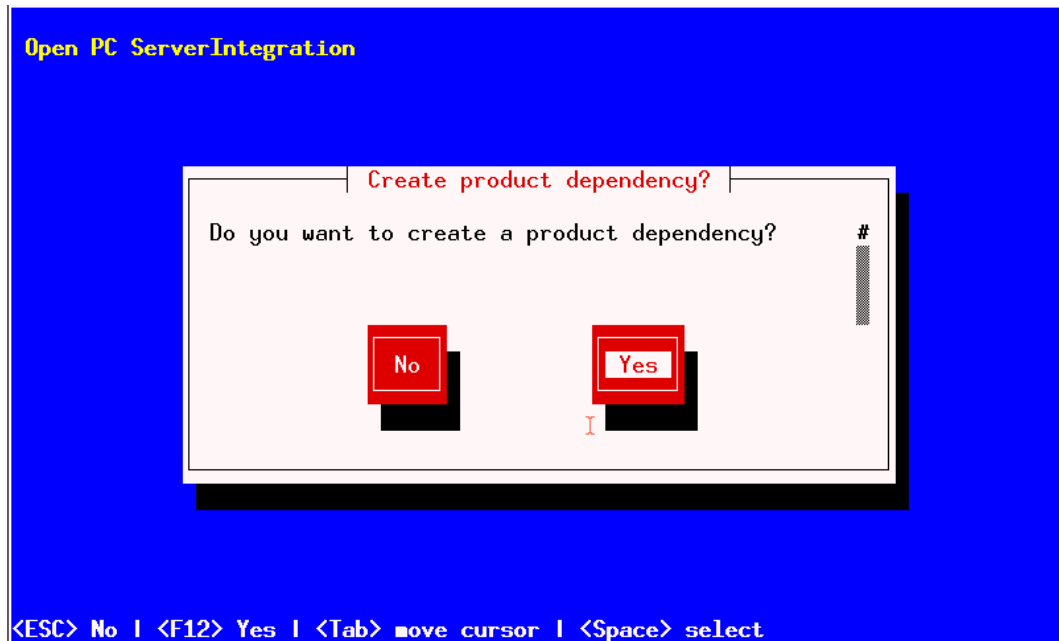


Figure 5.9: Create product dependency: No/Yes

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

Dependency for Action

For which product action shall the dependency be created (setup, deinstall ...).

Required product id

Product id of the required product.

Required product class id

For future use, leave it empty!

Required action

Select the required action (if any) for the required product. Actions can be as *setup*, *deinstall*, *update*. If no *required action* is set, a *required installation status* must be set

Required installation status

Select the required status of the required product (if any). Usually this is *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 deinstall actions or dependencies is broken. After defining a product dependency you will be asked whether to create another product dependency. If you choose *Yes*, the procedure for defining a product dependency is repeated; if you choose *No* 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 how this is done and what you can configure see at the opsi-manual.

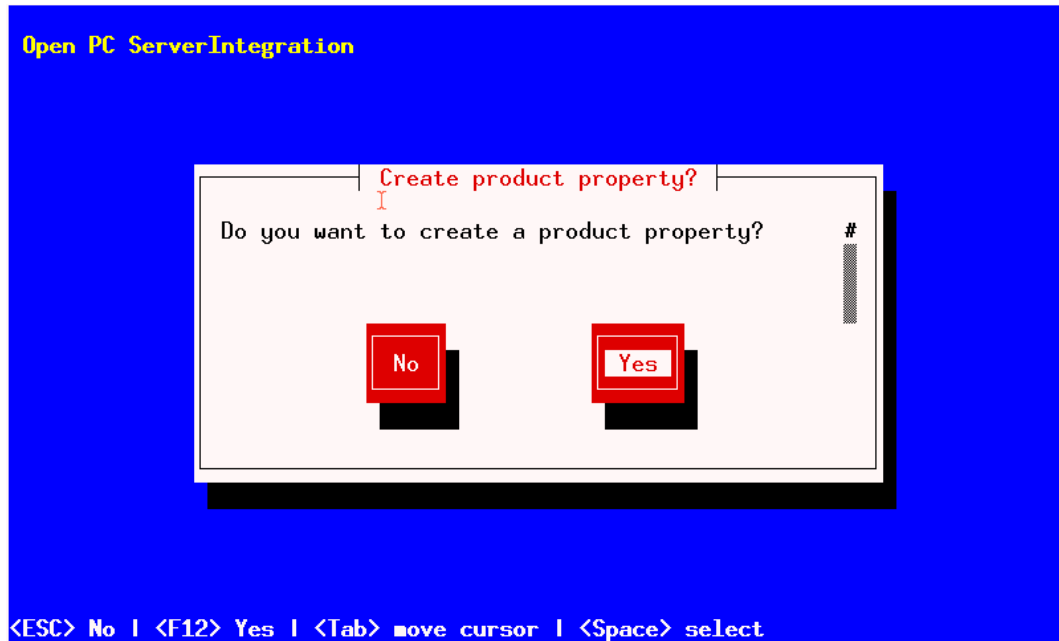


Figure 5.10: A(nother) product property to create?

If you answer *Yes* you will have to describe the product properties.

The product properties are client specific and are a name (key) which can hold different values. These values can be evaluated by the *opsi-winst* script and result in installing different options at installation time.

At first we have to declare if our property is a text value (*unicode*) or a logical value e.g. true/false (*boolean*). If you not sure choose *unicode*.

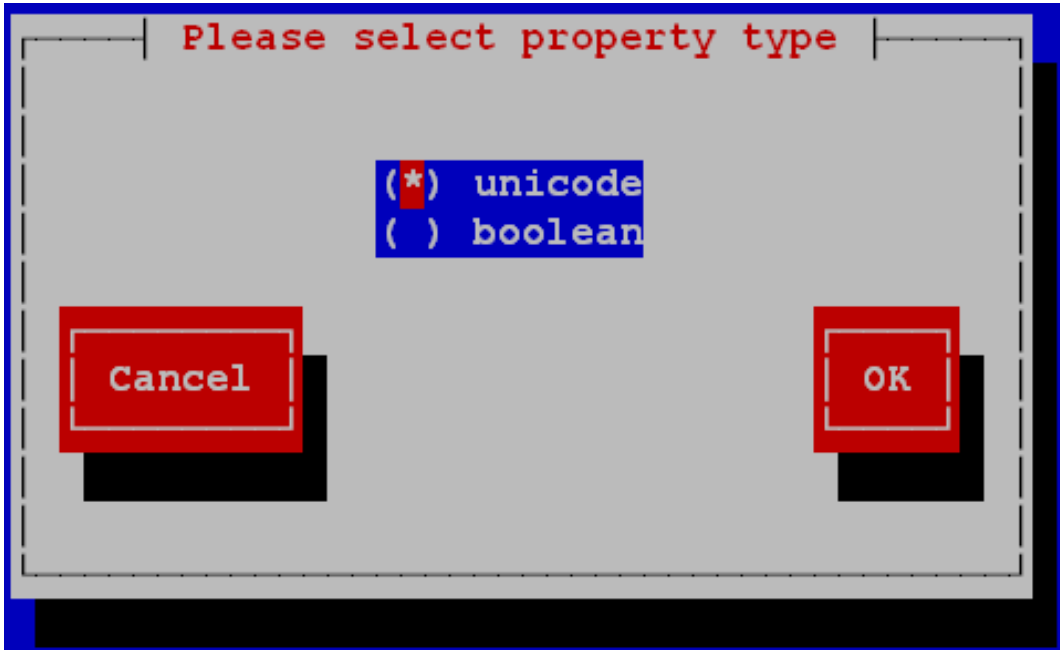


Figure 5.11: Choose the data type of the property

Further on a description for the switch needs to be specified, which will be shown in the *opsi-configeditor* as a help text and also when the package is unpacked. Next you can define the set of values for the switch (separated by comma). If this is left blank, any value is allowed for the switch.

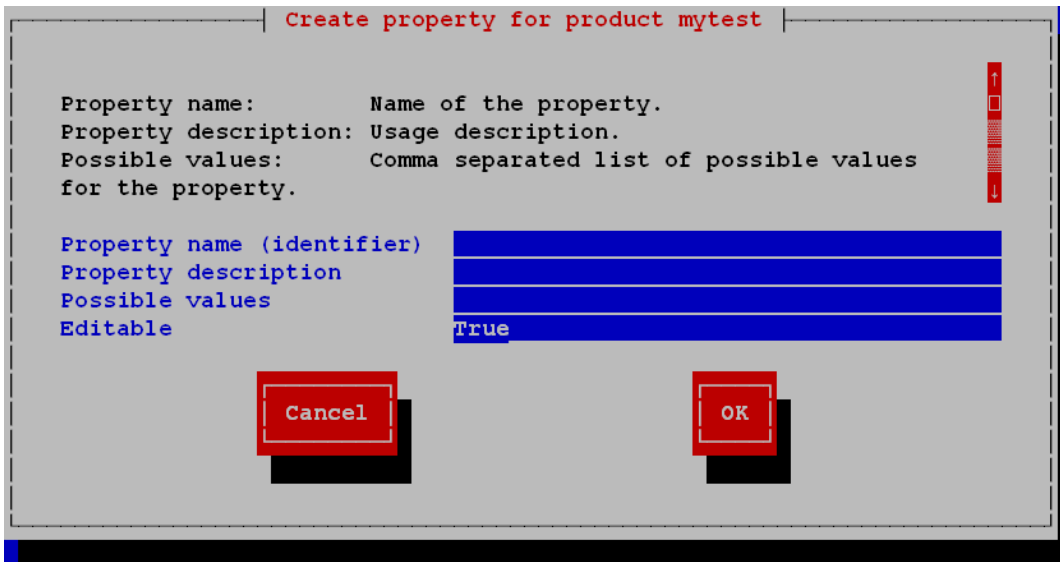


Figure 5.12: Description of the product properties

In the following you can define the default value of the product property (switch).

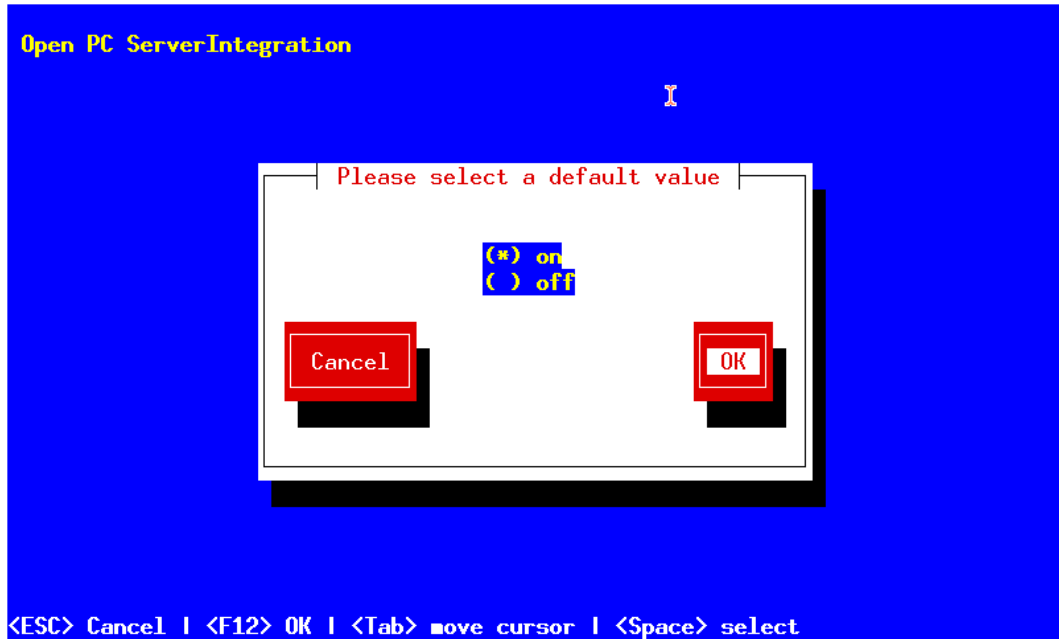


Figure 5.13: Default value of the product property

If choose as data type *boolean*, the description contains only *Property name* and *Property description*.

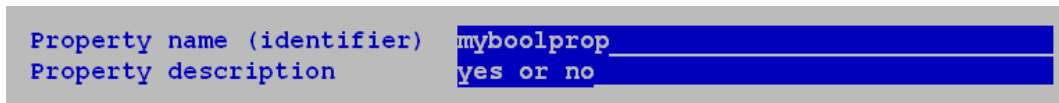


Figure 5.14: Description of a boolean property

After defining a product property, you will be asked whether to create another product property. If you choose *Yes* the procedure of defining a property repeats; if you choose *No*, you will be asked for name and email of the product maintainer. This data will be written to the changelog.



Figure 5.15: Input of the maintainer data

Now 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 have defined and you can load the file into an editor to view or change the entries.

Example of a control file:

```
[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
values: ["off", "on"]
default: ["off"]

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

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

 * Initial package

-- jane doe <j.doe@opsi.org>  Mi, 14 Jul 2010 12:47:53 +0000
```

As the next step you will have to copy the product *opsi-winst* script and the necessary data files into the `CLIENT_DATA` folder.

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

5.2.1.2 Build package with opsi-makeproductfile

Then you may build the package. Change to the root directory of the product and start *opsi-makeproductfile*. The product package will be build.

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

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

```
# opsi-makeproductfile --help

Usage: opsi-makeproductfile [-h] [-v|-q] [-F format] [-l log-level] [-i|-c custom name] [-I required version] [-t temp \
  dir] [source directory]
Provides an opsi package from a package source directory.
If no source directory is supplied, the current directory will be used.
```

```
Options:
-v      verbose
-q      quiet
-l      log-level 0..9
-n      do not compress
-F      archive format [tar|cpio], default: cpio
-h      follow symlinks
-I      incremental package
-i      custom name (add custom files)
-c      custom name (custom only)
-C      compatibility mode (opsi 3)
-t      temp dir
```

Use the option `-C` (compatibility mode to opsi 3) to build a packages on a opsi 4 server which have to be installable at opsi 3 servers as well.

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

```
Package file '/home/opsiproducts/mytest/mytest_3.14-1.opsi' already exists.
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 -i <paketname>`

For more information about the opsi-package-manager see at the opsi-manual.

Chapter 6

More informations

A lot of more and more detailed information you will find in the opsi-manual.

If you need help while evaluation you will get it at <https://forum.opsi.org>

For productive installations we recommend our commercial support: <http://uib.de/en/home/index.html>