**Documentation**
as of: **06/16/2010**

# opsi
# Getting Started

# opsi-Version 3.4

***installation manual opsi-server***
***start-up***
***first steps***

***open pc server integration***

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

# Table of Contents

# 1. Introduction

This instruction explain detailed installation and starting of an opsiserver. It starts from the provided installation package and lead 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 a opsiserver and clients for the first trials).

We approve urgently for the first trials with opsi in a test net separated from other DHCP-server. Temporarily should a connection to the main net be possible for download actual product packages.

For an integration in concomitant nets you can ask for consulting service from your office (uib) if necessary.

## 1.1 Steps for installation and starting

Three steps for an installing and starting of an opsi-server:

(a) base installation of the server

(b) adjustment the server: configuration of the network, password awarding, updating of the server

(c) download and installation of the essential opsi products for the clients

(d) complete the System Software Base Package for Windows from the original-CDs

After wards a client could be installed automatically.

For the base installation exists some versions to choose from your interest. For all these versions are the required data-packages in the internet provided:

The method of base installation variants are described in chapter 2 of this introduction.

## 1.1 Hardware requirement

For a opsiserver the following hardware is recommended:

- Intel-x86-compatible PC
- network interface card assisted by standard-linux kernel
- a hard disk with 16 GB capacity

- a bootable CD-ROM-drive

Neither in test handling or in reality handling are the requirements at the capacity on the machine high.

Working with a VMware-Machine needs a reasonable host computer. It's possible for test status that an other VMware-machine work as client in the same host computer.

# 2. Installation

## 2.1 opsi-server base installation

This chapter describe different version of realization for an opsiserver. If all steps are successful you get an server system, which is prepared for configuration and starting. You can choose your way realizing an opsiserver and ignore the other stages.

At the end you should update your system according 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

```
grey fields
```

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

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

### 2.1.1 Starting of a VMware-Machine

A opsiserver can be installed as a virtual machine because the required computer speed can be low. For VMware a corresponding machine is prepared. The data files are available in the Internet. In order to run this machine, the free of charge VM-ware-player is adequate. 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.

#### 2.1.1.1 First Start

If you already installed a complete VMware software or a VMware player, you only need

few mouse clicks for a opsi-server base installation:

- Download the file 'opsi3.4-servervm.zip' from the internet.
- Unzip the file and a directory 'opsiserver' will be generated.
- Start the VMware-player. Search with the data browser the directory 'opsiserver' and choose the data 'opsiserver.vmx'. Sometime you get a message that CDROM- and floppy-disk device have another address – you can ignore this message. The virtual machine boots.

The VMware-player is free of charge for all system software on vmware.com. Normally you can install it without problems, if the equipment of the host computer (specially memory) meet the needs of parallel running software system.

The virtual machine from uib is based on Linux. Properties for our host-system are described in the configuration file 'opsiserver.vmx'. If you run the opsiserver image under Windows or if your Linux system machine has another address, you have to adapt the file.

#### 2.1.1.2 Language configuration

The first step is to choose the preferred language:


*Figure 1: Language selection*

#### 2.1.1.3 „1stboot“

For the work with the opsiserver it could be helpful to connect them with the internet directly. For network configuration start the script `1stboot.py`.

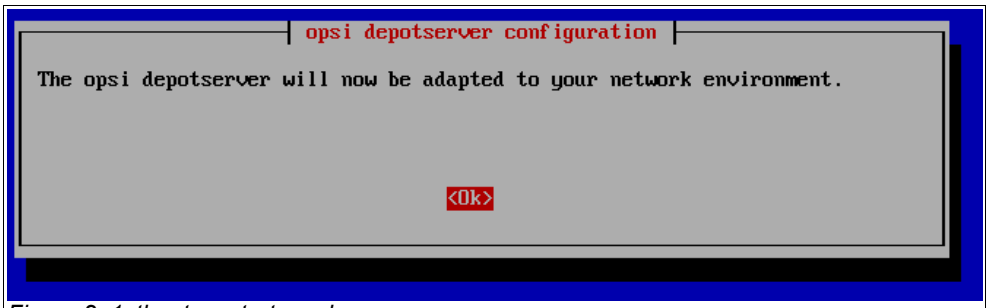


*Figure 2: 1stboot.py start mask*

Put in the configuration informations for your network and answer the questions.

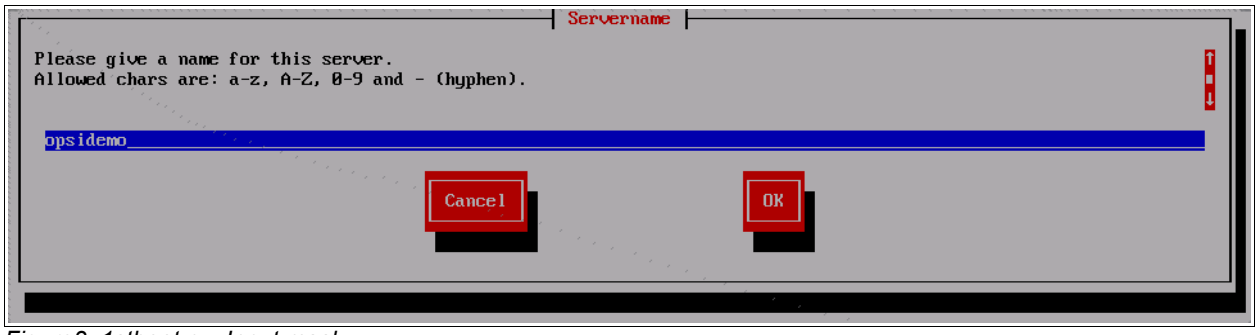


*Figure3: 1stboot.py: Input mask*

In the following you will asked for:

- server name — Name of this server (without domain) e.g. opsidepot
- 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
- 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 useful for the internet access the proxy informations: 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: — As 'TFTP server' you put in IP-number of the server (='IP-address') normally.
- Password of root — Password of root

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

A technical advice to the program 1stboot.py:
The program works with templates to modify the configuration files. If you work reapply with the program and want to edit the configuration files by yourself you find the template in:

**`/var/lib/1stboot/templates/`**

#### 2.1.1.4 Second start

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

You are on the graphic surface of the opsiserver directly (for the surface a sustainable use of resources surface so-called window manager will used). For salutatory an „Iceweasel“-browser-window with furthermore instructions and a reference on the available handbook attend.

If you get a message that their is no network connection, you should reboot the

computer before you are searching for the fault.

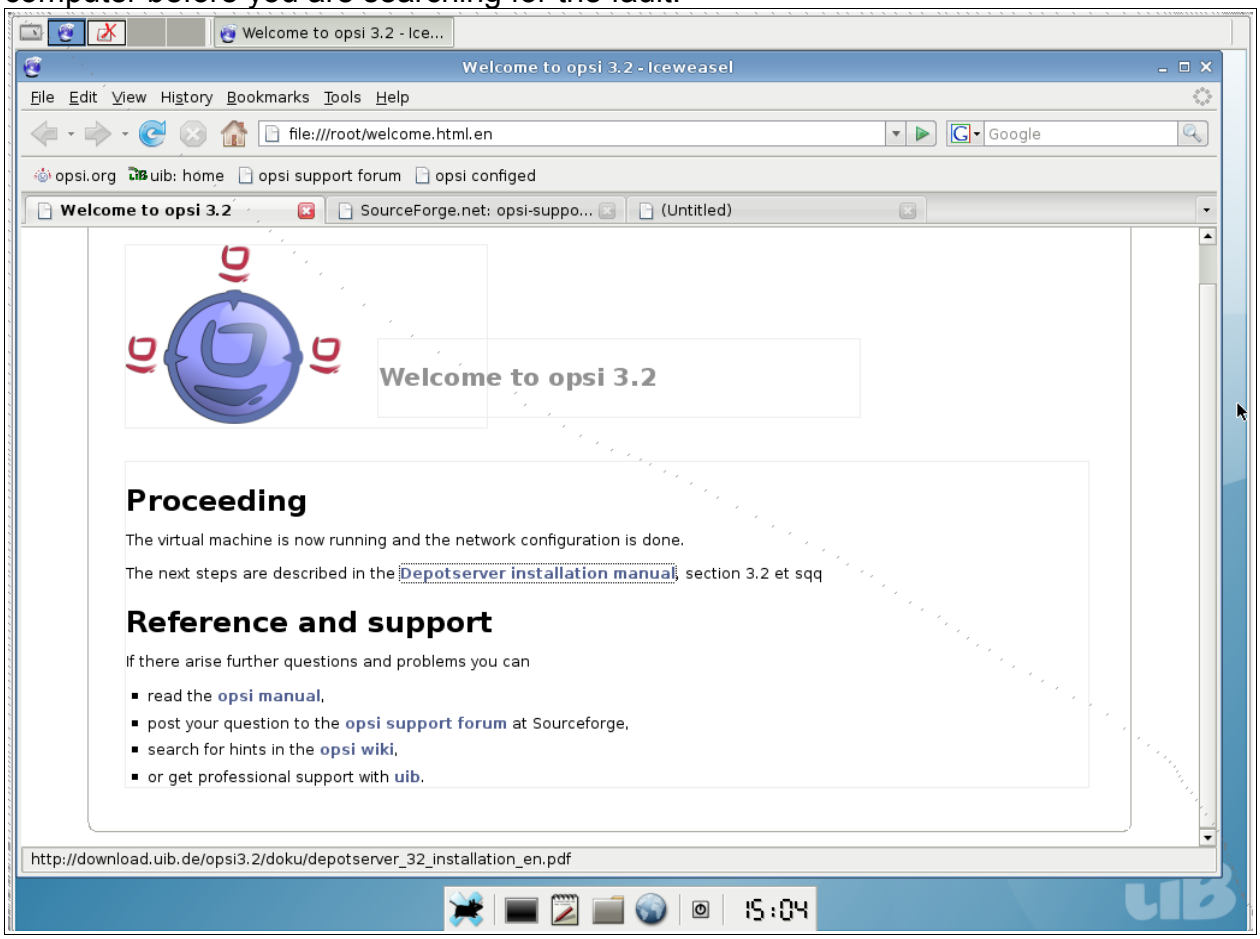



*Figure 4: Graphic start surface of the opsiserver*

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

- Per 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.

- Per vnc (normally under Linux with the  e.g. available vncviewer or krdc, under Windows e.g. with ultravnc, http://www.uvnc.com/ ) you can use per remote a graphic surface. The vnc-address will build out of the IP-address (or the server name by a already working name release) and a trailed „:1“. The password for the vnc access is „linux123“. You should change it soonest – how is describe in the next chapter.

#### 2.1.1.5 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 opsiserver (see the last chapter)
- Open a terminal window in the graphic surface (directly on the opsiserver or via vnc) with a click on the terminal icon in the icon bar.
- Open a terminal window in the graphic surface (directly on the opsiserver or via vnc) 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).

#### 2.1.1.6 Change the vnc password

The default password for the vnc connection is „linux123“ - you have to change it, best change is directly:

Open a new terminal window and write:

```
vncpasswd
```

On the following question you have to put in your new password. Minimum are 8 characters.

**General in productive systems should the vnc access be locked or only allowed tunneld with over SSH for security reasons. If vnc is admitted for the internal network a firewall has to block the internet port.**

#### 2.1.1.7 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 order

```
reboot
```

If the network connection works, you can put in opsi packages or actualize them. And build the environment for the first installation test.

### 2.1.2 Installation of a Debian (lenny) System with aptitude

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

Please note that an opsiserver 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 mc
```

opsi need a installed samba. Install it form Debian:

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

or install samba from the Sernet repositories:
Add in the file '/etc/apt/sources.list':

```
deb http://ftp.sernet.de/pub/samba/tested/debian lenny main
```

and install samba with the commands:

```
aptitude update
aptitude install sernet-cifs-mount sernet-samba sernet-samba-doc sernet-smbclient sernet-smbfs
```

If you want to use MySQL as Backend for Inventory and license management you will need the mysql-server as well:

```
aptitude install mysql-server
```

Working on a Ubuntu server you should also install: openbsd-inetd.

Check the opsiserver entry in **/etc/hosts** or the output of

```
getent hosts `hostname -f`
```

The result should be similar to

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

If the Result isn't like that and contains for example 127.0.0.1 or localhost you should correct your /etc/hosts or your DNS before you continue with the installation.

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

```
deb http://download.uib.de/debian lenny opsi3.4
```

Execute the following orders:

```
aptitude update
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 opsiconfd you will be asked for informations for a SSL certificate preparation.

During the opsiserver installation you have to allow the patching of the files 'dhcpd.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'.

Cause you install opsi on an existing machine we assume of a correct network configuration. So you can go on with chapter 2.2. Update and configuration of opsiserver / 2.2.3 Checking the java configuration

### 2.1.3 Installation on an univention corporate server (UCS 2.x)

Tested and certified for UCS <= 2.2

For opsi evaluation we recommend to use our VMware-Machine or a Debian based System

The Installation of opsi4ucs is possible on servers with the roles Master, Backup and Slave. Installation on Member servers is not possible.

The package opsi4ucs-ldap-schema has to be installed always on the master.

The following documentation is based on a installation on a Master server.

Insert the following informations in the file /etc/apt/sources.list:

```
deb http://apt.univention.de/2.0/unmaintained/ 2.0-0/all/
deb http://apt.univention.de/2.0/unmaintained/ 2.0-0/i386/
deb http://apt.univention.de/2.0/unmaintained/ 2.0-0/amd64/
deb http://apt.univention.de/2.0/unmaintained/ 2.0-0/extern/
deb-src http://apt.univention.de/2.0/unmaintained/ 2.0-0/source/
```

Add for UCS 2.1:

```
deb http://apt.univention.de/2.1/unmaintained/ 2.1-0/all/
deb http://apt.univention.de/2.1/unmaintained/ 2.1-0/i386/
deb http://apt.univention.de/2.1/unmaintained/ 2.1-0/amd64/
deb http://apt.univention.de/2.1/unmaintained/ 2.1-0/extern/
deb-src http://apt.univention.de/2.1/unmaintained/ 2.1-0/source/
```

Add for UCS 2.2:

```
deb http://apt.univention.de/2.2/unmaintained/ 2.2-0/all/
deb http://apt.univention.de/2.2/unmaintained/ 2.2-0/i386/
deb http://apt.univention.de/2.2/unmaintained/ 2.2-0/amd64/
deb http://apt.univention.de/2.2/unmaintained/ 2.2-0/extern/
deb-src http://apt.univention.de/2.2/unmaintained/ 2.2-0/source/
```

Add final:

```
deb http://<username>:<password>@download.uib.de/debian ucs2.0 opsi3.4
```

You have to displace <username> and <password> with your login data.

Complete the following orders:

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

During the tftpd-installation you will possibly be asked for the tftp directory. Answer with '/var/lib/univention-client-boot/'. The question after the multicast support you can answer with 'no'.

During the installation you will be asked some questions – see also 2.1.2 Installation of a Debian (lenny) System with aptitude

If you are installing on a UCS server role other than master please run the opsi4ucs join

script at this point:

```
univention-join [options]
```

The opsi configuration editor can be installed optional as applet on the UCS-server.

Complete the following orders:

```
aptitude install opsi-configed
/etc/init.d/opsiconfd restart
```

The applet can called up with the URL https://<servername>:4447/configed.

For using the opsi configuration editor the user has to be member of the group 'opsiadmin'. The group membership of an user can be configured with the univention-admin.

Please Notice:
The configurations in the at the opsi manual frequently mentioned file
etc/opsi/backendManager.d/30_vars.conf
is superseded by the configurations of the file
/etc/opsi/backendManager.d/40_opsi4ucs.conf
at opsi4ucs installations.

Cause you install opsi on an existing machine we assume of a correct network configuration. So you can go on with chapter 2.2. Update and configuration of opsiserver / 2.2.3 Checking the java configuration

### 2.1.4 Installation on openSUSE 10.3/11.0

First general notes:

●These packages are tested with Open-Suse 10.3/11.0. (Opensuse 11.1 is not supported yet)

●uib gmbh approved for evaluation the usage of opsi-VM or a Debian or Ubuntu Systems because the product dependence could be dissolved comfortable.

●Support for Suse based systems afford uib gmbh in context of a Professional Support contract.

Necessary preparations:

- The command “hostname -f” must return a fully qualified domainname containing two dots, e.g. opsiserver.uib.local

- The command “getent hosts `hostname -f`” must return the ip address of the interface the clients should connect to. If the result is 127.0.0.2 please correct your /etc/hosts.
- Samba must be configured
- 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 opsi-SUSE-Repository:

```
zypper ar http://download.uib.de/suse/opsi3.4 opsi3.4
```

Depending on the used openSUSE version you will have to resolve some package dependencies. The following packages may be not part of the distribution:

- python-crypto
- python-mysql
- python-twisted-web2
- python-newt
- duplicity

The required packages may be available in the following community-repositories:

openSUSE 10.3

http://download.opensuse.org/repositories/devel:/languages:/python/openSUSE_10.3/

http://download.opensuse.org/distribution/10.3/repo/oss/suse

http://download.opensuse.org/distribution/10.3/repo/non-oss/suse

http://download.opensuse.org/repositories/home:/lrupp:/Factory/openSUSE_10.3

http://download.opensuse.org/repositories/home:/Saviq/openSUSE_10.3

http://download.opensuse.org/repositories/home:/jimfunk/openSUSE_10.3/

openSUSE 11.0

http://download.opensuse.org/repositories/devel:/languages:/python/openSUSE_11.0/

http://download.opensuse.org/repositories/home:/dsbhayangkara/openSUSE_11.0_Update/

http://download.opensuse.org/repositories/home:/Saviq/openSUSE_11.0/

After adding the repositories the installation of opsi can be started by executing:

```
zypper install opsi-depotserver
zypper install opsi-configed
rcopsiconfd restart
rcopsipxeconfd restart
```

Cause you install opsi on an existing machine we assume of a correct network configuration. So you can go on with chapter 2.2. Update and configuration of opsiserver / 2.2.3 Checking the java configuration

Please notice: the unix commands that are used in the following chapters are working on Debian systems. Perhaps you must change them in order to work on your linux system.

## 2.2 Update and configuration of opsiserver

### 2.2.1 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
```

### 2.2.2 Update of the opsiserver

Update the opsiserver with the commands:

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

### 2.2.3 Checking the java configuration

Administrate the opsiservers 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).

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'.
```

### 2.2.4 Change passwords

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
```

### 2.2.5 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
```

Create and test the group membership

```
adduser adminuser opsiadmin
grep opsiadmin /etc/group
```

the grep command should have a result like:

```
opsiadmin:x:993:root,adminuser
```

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

All user which pack packages (opsi-makeproductfile), install (opsi-package-manager) or edit manuell configuration files have to be in the group 'pcpatch' additional:

```
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.

### 2.3 DHCP-Configuration

#### 2.3.1 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. See also the DHCP/DNS Chapter at the opsi-manual.

#### 2.3.2 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 it will also create a dhcp entry for this client. Therefore you have to supply the IP-number and the MAC-address.

#### 2.3.3 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-Servers.

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.

### 2.3.4 Checking the backend configuration for DHCP entries

In the file /etc/opsi/backendManager.d/30_vars.conf is defined witch backend manage of opsi be used (**BACKEND_FILE31, BACKEND_FILE, BACKEND_LDAP**). The default backend is **BACKEND_FILE31.** In the entry **clientManagingBackend** may be controlled if opsi also assume the local DHCP configuration. This is sensible if the DHCP-server of the opsiserver will be used (default). The accordant entry is:

```
self.clientManagingBackend = [ BACKEND_DHCPD, BACKEND_FILE31 ]
```

If the local DHCP isn't used also the **BACKEND_DHCPD** not required:

```
self.clientManagingBackend = BACKEND_FILE31
```

After adapting the backend configuration the 'opsi-confd' has to be restarted:

```
/etc/init.d/opsiconfd restart
```

### 2.4 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 payed 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. For the completed co-funding projects you will find a activation file at http://download.uib.de/opsi3.4/modules which you may copy as root to **/etc/opsi**.

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

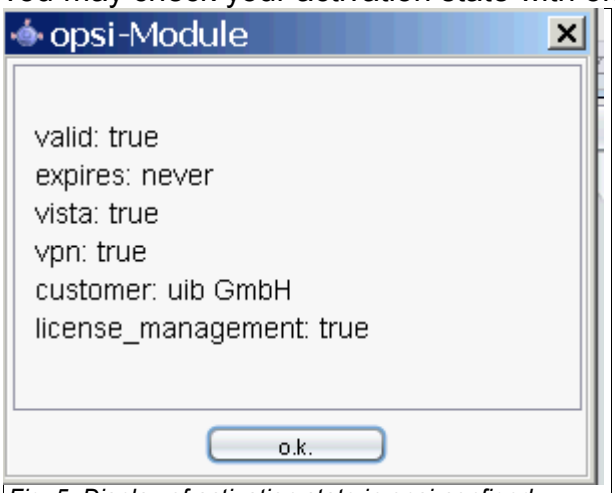


*Fig. 5: Display of activation state in opsi-configed*

Using the opsi-configed 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 **getOpsiInformation_hash.**
(Remark: Never give your activation file or the output of this command to third people without deleting the signature).

```
opsi-admin -d method getOpsiInformation_hash
{
"opsiVersion" : "3.4.0.0",
"modules" :
          {
          "customer" : "uib GmbH",
          "vista" : false,
          "license_management" : true,
          "expires" : "never",
          "valid" : true,
          "signature" : "THIS-IS-NOT-A-VALID-SIGNATURE",
          "vpn" : true
          }
}
```

### 2.5 Install the minimal opsi-products

Get the actual necessary opsi-packages in the opsi-package format.

The packages are located at http://download.uib.de/opsi3.4/produkte/essential/

After the download the packages have to be installed at the server using the command

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

You may do this interactive for every single package. The recommended way is to do this automatically as described here:

```
cd /home/opsiproducts

wget -r -l1 -nd -nc -A '*.opsi'
http://download.uib.de/opsi3.4/produkte/essential
```

If the 'wget' command failed so possible the environment variable 'http_proxy' has to been set on the correct proxy string (e.g. http_proxy=http://192.168.1.5:8080/)

Put in the downloaded packages on your server and install them so the product is available for the clients. The interactive installation of a opsi package start with **`opsi-package-manager -i <paketname>.opsi`**

The following order installed all downloaded packets successive:

```
opsi-package-manager -i *.opsi
```

Please notice that the OS-Installation products like winxppro and win2k aren't ready for action after installation. The installation has to be supplemented by the installation files of the accordant installation mediums (see chapter 3.2.4 OS-Installation: Complete the base package for Windows at page 30 of this manual).

You are invited to download more opsi-products from download.uib.de and install them on your opsi-server similarly.

### 2.6 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 adress in the browser (anywhere in the net) **`https://<opsiserver>: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 teh 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.

The handling is easy mostly self explaining. So here only two hints:

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.

# 3. 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.

## 3.1 Software Deployment

### 3.1.1 Integration of existing clients

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

#### 3.1.1.1 Usage of the opsi-deploy-preloginloader

The opsi-deploy-preloginloader script installs the opsi-preloginloader direct from the opsi depot server on 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 pckey to the client and starts the installation on the client.

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

Attention: During installation the client reboots without notice!

```
bonifax:/home/uib/oertel# cd /opt/pcbin/install/preloginloader
bonifax:/opt/pcbin/install/preloginloader# ./opsi-deploy-preloginloader -h

Usage: opsi-deploy-preloginloader [options] [host]...
Deploy opsi preloginloader to the specified clients.
The c$ and admin$ must be accessible on every client.
Options:
    -h        show this help text
    -V        show version information
    -v        increase verbosity (can be used multiple times)
    -u        username for authentication (default: Administrator)
    -p        password for authentication
    -f        file containing list of clients (one hostname per line)
```

#### 3.1.1.2 Usage of service_setup.cmd

Also in  /opt/pcbin/install/preloginloader located is the script service_setup.cmd. This can be started with administrative privileges from the client side. 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 'opsiadmin'.

Attention: During installation the client reboots without notice!

### 3.1.2 First Tests

#### 3.1.2.1 Hard- and Softwareinventory with the products hwaudit and swaudit

After wards choosing the client and in the crystal tab 'product configuration' in the line of the wished system software (hwaudit and/or swaudit) the action 'setup' choose and save with a click on the hook button.

Now reboot the client, the installation of python (and after a reboot) 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.

#### 3.1.2.2 Hardwareinventory with the netboot product hwinvent

After wards choosing the client and in the crystal tab 'netboot-Products' in the line of the wished system software (e.g. hwinvent) the action 'setup' choose and save with a click on the hook button.

Now reboot the client (over PXE), the hwinvent should be started. The client load after the reboot a linux-boot-image which scans the hardware and send the results back to the server.

### 3.2 Installation of a new Windows Machine using opsi (OS Installation)

#### 3.2.1 Creating a new client via opsi management interface

You need a client (minimum 256 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:

With menu item 'OpsiClient/Create new opsi client' and the description of IP-name, domain, client description, IP-number (which is only requested by the internal DHCP) and MAC-address you finished the client creation. 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 opsiserver.

Command line opsi-admin:

```
opsi-admin -d method createClient <clientname> <domain>
<description>  <notes>  <ipAddress>  <hardwareAddress>
```

e.g.:

```
opsi-admin -d method createClient pxevm uib.local "Testclient" ""
192.168.0.5 00:0c:29:12:34:56
```

After you have done so, you should see the client at the crystal tab 'client selection'.

### 3.2.2 Hardwareinventory with the netboot product hwinvent

After wards choosing the client and in the crystal tab 'netboot-Products' in the line of the wished system software (e.g. hwinvent) the action 'setup' choose and save with a click on the hook button.

Now reboot the client (over PXE), the hwinvent should be started. The client load after the reboot a linux-boot-image which scans the hardware and send the results back to the server.

### 3.2.3 Create a new client using the opsi-client-bootcd

At the opsi download site you will find in http://download.uib.de/opsi3.4/ 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:

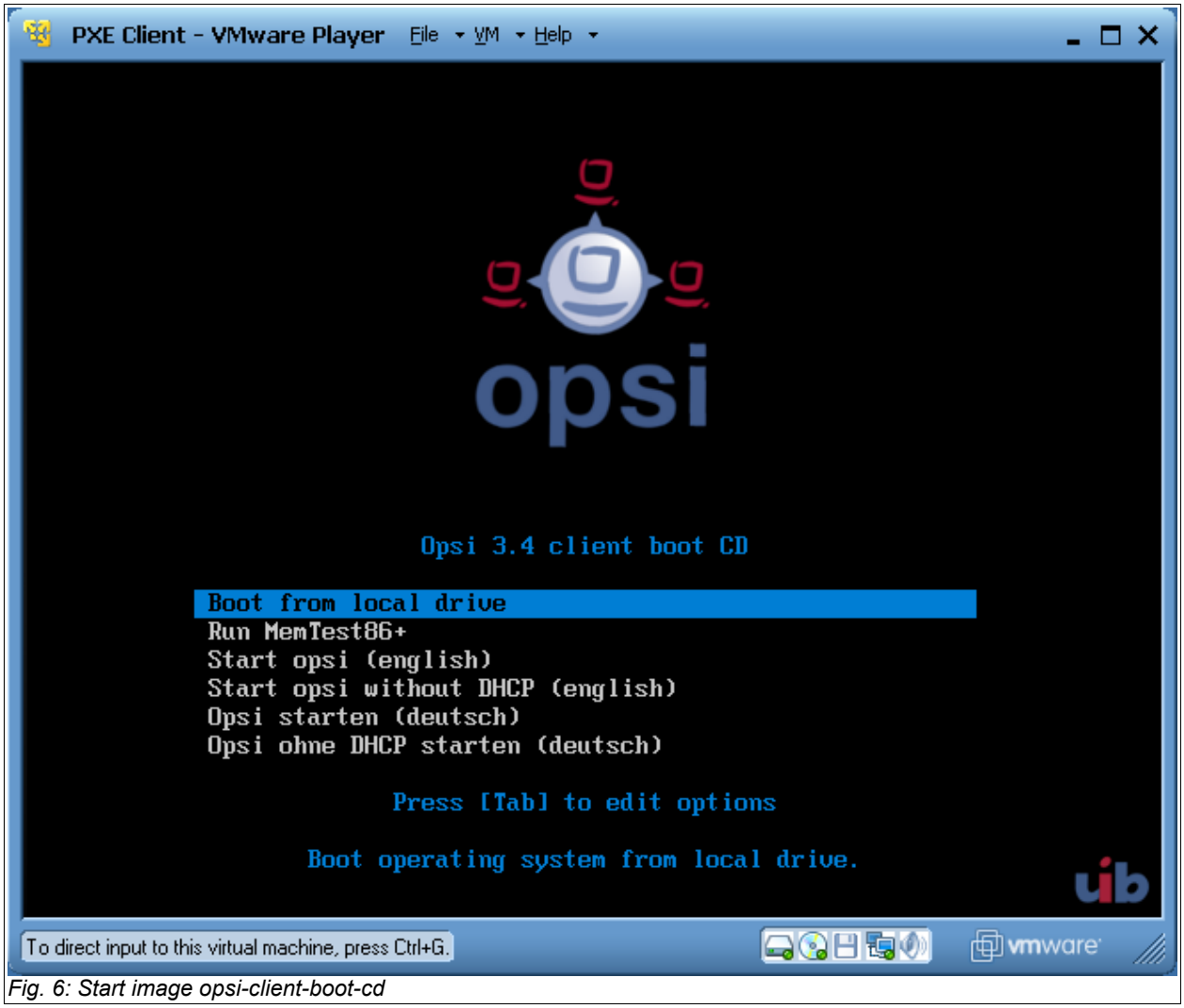


*Fig. 6: Start image opsi-client-boot-cd*

Choose 'Start opsi (english)'. After a while the following screen will appear. If your DHCP-Server give IP-Numbers to unknown DHCP-Clients, so must things will be completed. You have to complete the missing data. At least you must give the host name. Confirm with 'OK'.

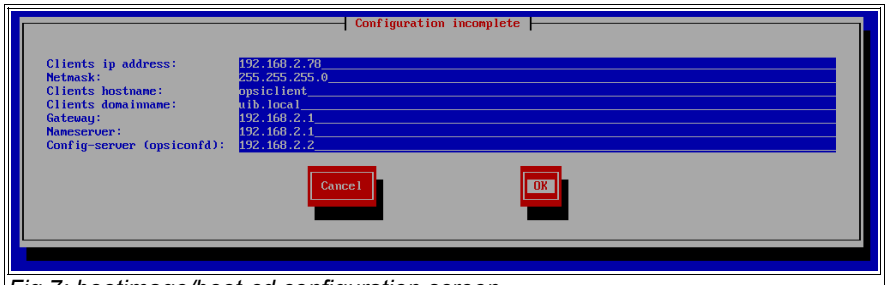


*Fig 7: bootimage/boot-cd configuration screen*

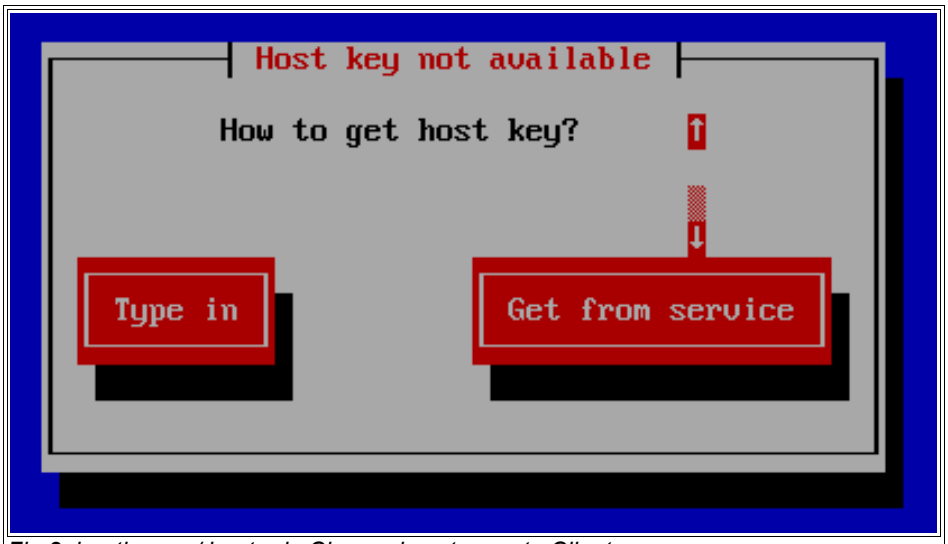


*Fig 8: bootimage / boot-cd: Choose how to create Client*

Choose 'Get from service'. This means, the client should register himself at the opsi server. This procedure must be authorized.

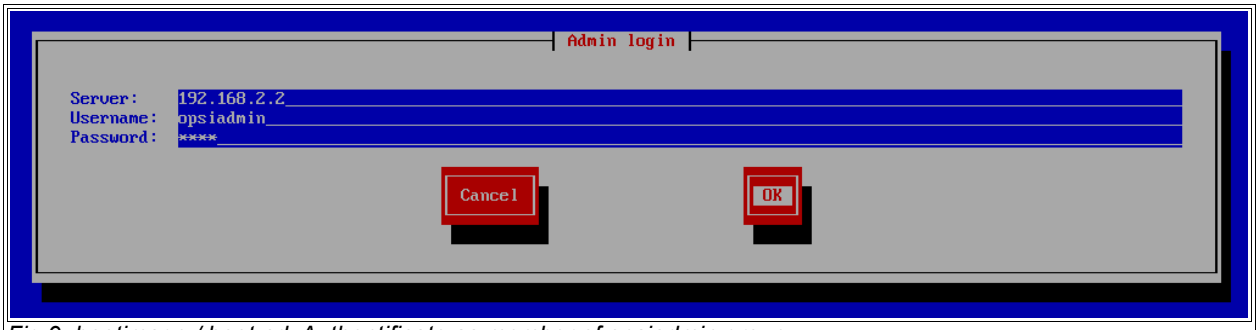


*Fig 9: bootimage / boot-cd: Authentificate as member of opsiadmin group*

Therefore you will get a login mask, where you should authentifcate you as a member of the opsiadmin group. If you successful authenficated, the client give its data to the server and the client will be created at the server. In the next step the server gives thelist of netboot products to the client.

*Fig 10: bootimage / boot-cd:: netboot product list*

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

### 3.2.4 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 2000- or Windows XP-system software installation, you have to complete these packages.

### 3.2.5 Windows 2000, XP, 2003

#### 3.2.5.1 Copy the i368 directory

a) copy the i386-directory of a installation-CD for Microsoft Win2k/WinXP Professional in the directory **`/opt/pcbin/install/win2k`** or **`/opt/pcbin/install/winxppro`** directory. After copy you have to change the rights of the i386/ directory:

```
chown -R opsiconfd:pcpatch i386/
chmod -R ug+rw i386/
```

The file can be copied also over the network. Therefore you have to connect as user “pcpatch“ by the approval “opt_pcbin“ on the opsiserver. The corresponding directory you will find under install\winxppro or install\win2k.

### 3.2.6 Windows Vista / 2008 / Windows 7

Because the Vista installation only starts from a Win32/Win64 environment we must build a PE-Image which is used to startup 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

Version date is 8/6/2009.

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.

#### 3.2.6.1 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**.

1. 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
```

2. Prepare Image for opsi:

   Start a terminal as administrator and run the following commands:

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

```
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**.)

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

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

3. Copy the directory **C:\winpe\ISO** with the target name **winpe** to **/opt/pcbin/install/winvista/** respectively

**/opt/pcbin/install/win2008**

Adjust the file access rights:

```
chown -R opsiconfd:pcpatch /opt/pcbin/install/winvista/winpe
```

3.2.6.1.1 Creating a 64-Bit Vista-PE

1. Installation of the Windows AIK at Vista 64-Bit or Windows 2003 Server 64-Bit
2. Create a WinPE

   Copy the file **boot.wim** from the Winvista-64-Bit-CD to the directory **C:\winpe_amd64\ISO\sources**

   Remove the write protection from **boot.wim**
3. mount the image with:

   **"C:\Program Files\Windows AIK\Tools\amd64\imagex.exe" /mountrw C:\winpe_amd64\ISO\sources\boot.wim 1 c:\winpe_amd64\mount**

   Edit with **Notepad.exe C:\WinPE_amd64\mount\Windows\System32\startnet.cmd**
   Delete the entry wpeinit and add the following line:

   **c:\opsi\startnet.cmd**

   (Remark: The file **startnet.cmd** will be created by the opsi Linux boot image by executing the script **winvista.py.** The startnetcmd contains the call to wpeinit.)

   unmount the image

   **"C:\Program Files\Windows AIK\Tools\amd64\imagex.exe" /unmount /commit C:\winpe_amd64\mount**

   **Please note:** At 64-Bit Vista also the archive **2** must be mounted and edited:

```
"C:\Program Files\Windows AIK\Tools\amd64\imagex.exe"
/mountrw C:\winpe_amd64\ISO\sources\boot.wim 2
c:\winpe_amd64\mount
```

Edit with **Notepad.exe**
**C:\WinPE_amd64\mount\Windows\System32\startnet.cmd**
Delete the entry wpeinit and add the following line:

```
c:\opsi\startnet.cmd
```

**Additional** remove the files **setup.exe** and **sources\setup.exe**

Image unmounted

```
"C:\Program Files\Windows AIK\Tools\amd64\imagex.exe"
/unmount /commit C:\winpe_amd64\mount
```

Copy the directory **C:\WinPE_amd64\ISO** with the target name **winpe** to **/opt/pcbin/install/winvista64/** respectively **/opt/pcbin/install/win2008_64/**

Adjust the file access rights:

```
chown -R opsiconfd:pcpatch /opt/pcbin/install/winvista64/winpe
```

#### 3.2.6.2 unattend.xml

The control file for the unattended Installation is the **unattend.xml** which you will find below **/opt/pcbin/install/winvista/opsi.** If you make any modifications at this file, it should be backuped in a different directory, because the opsi directory is subject of updates in future versions of the opsi winvista package.

The unattend.xml delivered wit the opsi winvista 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
**\Program Files\Windows\Waik\docs\chms**

#### 3.2.6.3 Driver integration

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

Please keep in mind that Vista only accept signed drivers. Therefor it is no good idea to use driver packs which contain none vista drivers like the driver packs from driverpacks.net.

#### 3.2.6.4 Providing the installation files

Copy the complete installation DVD to

**`/opt/pcbin/install/winvista/vistasrc`**

Adjust the file access rights:

```
chown -R opsiconfd:pcpatch /opt/pcbin/install/winvista/vistasrc
```

#### 3.2.6.5 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)

### 3.2.7 Windows product key

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

If you don't want to use the license management module, the productkey 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 'nettboot products'
- select the product (e.g. winxppro)
- change to the product propery '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 out giving a special client you will read / change the server default.

To read the server default use:

```
opsi-admin -d method getProductProperties_hash winxppro
```

To set the server default use (it has to be one line):

```
opsi-admin -d method setProductProperty winxppro "productkey" "ABCDE-FGHIJ-KLMNO-QRTUV-WXYZ1"
```

### 3.2.8 Start the Windows installation

To startup a windows installation:

- choose a client
- change to the tab 'nettboot 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.

### 3.2.9 Structure of the unattended installation products

This chapter describes the following products since opsi version 3.3.1:

- win2k
- winxppro
- winvista
- win2003
- win2008
- winvista-x64
- win2008-x64

#### 3.2.9.1 Directotry 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
```

#### 3.2.9.2 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.

#### 3.2.9.3 Directory i386 / installfiles / winpe

Please note the informations in the installation manual according to these directories.

•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 installation files of the windows installation cd (NT6/7 = Windows Vista and above)

•winpe
his directory contain at Windows Vista and above a bootable winpe image.

#### 3.2.9.4 Directory opsi / custom

Both directories contain scripts and configuration files for th 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. These scripts are needed to install the opsi preloginloader 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.

#### 3.2.9.5 Directory drivers

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

### 3.2.10 Simplified driver integration with symlinks

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 XP-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/winxppro/extract_driver_pack.py <path to the compressed driverpacks>**
these driver packs will be decompressed and stored in the directory **winxppro/drivers/drivers/D** .

You may also use the command: **/opt/pcbin/install/winxppro/download_driver_pack.py** . It will try to download these driverpacks from a driverpack.net mirror and decompress them to the correct directory.

Structure of the drivers directory tree:

```
/opt/
  └─pcbin/
     └─install/
        └─winxppro/
           └─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 drivers that are not detected)
                 ├─buildin/         (inf files from the i386 directory)
                 ├─preferred/       (preferred drivers for your hardware)
                 └─mydriverpacks/   (example driver pack)
```

Additional drivers can be added, each into its own sub directory of

**winxppro/drivers/drivers/preferred** .

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 '**winxxpro.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 informations from the i386 directory to the 'buildin' directory. If you make any changes to the i386 directory (e.g by integrating a service pack) you should delete the building directory an 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' 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' 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

Example of the output of show_drivers.py :

```
./show_drivers.py pcdummy

PCI-Devices
   [(StandardsystemgerÃ¤te), PCI Standard-PCI-zu-PCI-BrÃ¼cke]
      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/winxppro/drivers/drivers/D/M/N/123
   [Realtek Semiconductor Corp., Realtek RTL8168C(P)/8111C(P) PCI-E Gigabit
Ethernet NIC]
```

```
      /opt/pcbin/install/winxppro/drivers/drivers/preferred/realtek_gigabit_ne
t_8111_8168b
   [IEEE 1394 OHCI-konformer Hostcontroller-Hersteller, OHCI-konformer IEEE
1394-Hostcontroller]
      No driver - device directory
'/opt/pcbin/install/winxppro/drivers/pciids/197B/2380' not found
   [Advanced Micro Devices, Inc., AMD AHCI Compatible RAID Controller]
      /opt/pcbin/install/winxppro/drivers/drivers/preferred/ati_raid_sb7xx
   [(Standard-USB-Hostcontroller), Standard OpenHCD USB-Hostcontroller]
      No driver - device directory
'/opt/pcbin/install/winxppro/drivers/pciids/1002/4397' not found
   [ATI Technologies Inc, ATI SMBus]
      /opt/pcbin/install/winxppro/drivers/drivers/preferred/ati_smbus

USB-Devices
   [(Standard-USB-Hostcontroller), USB-VerbundgerÃ¤t]
      /opt/pcbin/install/winxppro/drivers/drivers/preferred/brother_844x_pGerb
   [Microsoft, USB-DruckerunterstÃ¼tzung]
      /opt/pcbin/install/winxppro/drivers/drivers/preferred/brother_844x_pGerb

Additional drivers
   [ati_hdaudio_azalia]
     /opt/pcbin/install/winxppro/drivers/drivers/additional/ati_hdaudio_azalia
```

# 4. 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.

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

### 4.1.1 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/doku

http://download.uib.de/opsi_stable/doku

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

### 4.1.2 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 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 'wInst' program. Usually that is something like file installation to the local file system and patching the registry

**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.

### 4.1.3 Structure of a winst script

A winst script contains primary and secondary sections.

Example:

```
[Aktionen]
winbatch_tightvnc_silent_install

[winbatch_tightvnc_silent_install]
"%SCRIPTPATH%\tightvnc-1.3.9-setup.exe" /silent
```

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

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

### 4.1.4 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:

| | |
|---|---|
| %ProgramFilesDir% | c:\program files |
| %Systemroot% | c:\windows |
| %System% | c:\winnt\system32 |
| %Systemdrive% | c:\ |
| %Scriptpath% | <path to the running script> |

### 4.1.5 Primary sections:

#### 4.1.5.1 Initial

The Initial section is used to set runtime parameters.

This section is optional.

#### 4.1.5.2 Actions

The section [Aktionen] ('actions') is the core main program.

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

#### 4.1.5.3 Sub-sections

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

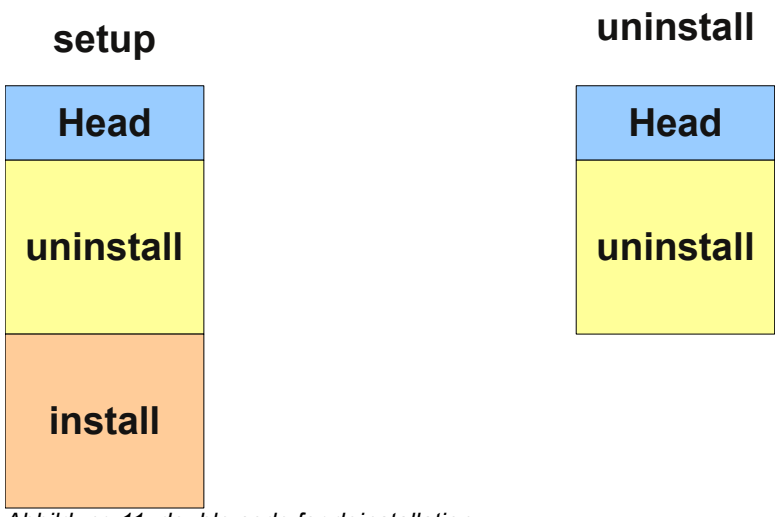



*Abbildung 11: double code for deinstallation*

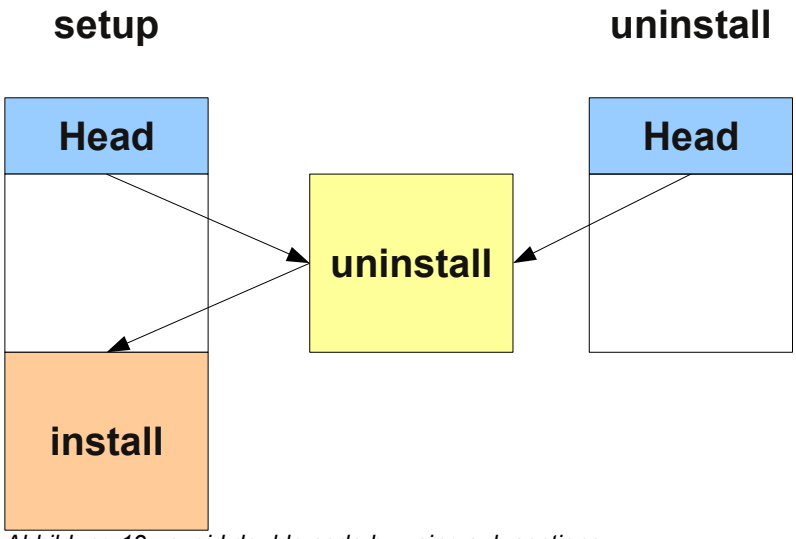


*Abbildung 12: avoid double code by using sub sections*

### 4.1.6 Important kinds of secondary sections

#### 4.1.6.1 Files

File operations like

copy (regarding the internal version information, recursive, ...)

delete files or directories

create directories

.....

#### 4.1.6.2 WinBatch

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

#### 4.1.6.3 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.

#### 4.1.6.4 ExecWith

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

#### 4.1.6.5 Registry

The Registry sections are used for registry manipulations.

#### 4.1.6.6 Linkfolder: startmenu and desktop

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

### 4.1.7 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

```
[Aktionen]
Message=installing tightvnc 1.3.9 ......
ExecWith_autoit_confirm "%SCRIPTPATH%\autoit3.exe" WINST /letThemGo
winbatch_tightvnc_silent_install
killtask "autoit3.exe"

[winbatch_tightvnc_silent_install]
"%SCRIPTPATH%\tightvnc-1.3.9-setup.exe" /silent

[ExecWith_autoit_confirm]
; wait for the confirm dialog
; which only appears if tightvnc was installed before as service

; waiting for the window
WinWait("Confirm")
; activating
WinActivate("Confirm")
; say no
Send("N")
Sleep(500)
;and good bye
Exit
```

### 4.1.8 Elementary commands for primary sections

#### 4.1.8.1 String Variable

Declaration of a variable: DefVar <variable name>

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

Example:

```
DefVar $ProductId$

set $ProductId$ = "softprod"
```

Important notice:

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).

#### 4.1.8.2 Message / showbitmap

Displaying text while runtime:

Message <string>

or

Message = <const string>

Example:

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

Display a picture while installation:

ShowBitMap [/<number>] [<file name>] [<sub titel>]

Example:

```
ShowBitmap /3 "%scriptpath%\python.png" "Python"
```

#### 4.1.8.3 if else endif

Syntax:

if <condition>

    ;statement(s)

[else

    ;statement(s)]

endif

#### 4.1.8.4 Functions

HasMinimumSpace

Check for free space on the hard disk.

FileExists

Check for the existence of a file or directory

#### 4.1.8.5 Error, logging and comments

- LogError

writes error messages to the log file

- isFatalError

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

- comment

writes a comment to the log file

- comment char ';'

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

### 4.1.9 Third example: The generic template 'opsi-template'

You encouraged to use this template (or a updated version) when ever you create a own opsi product.

4.1.9.0.1 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=$Licensekey$
; msiexec /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 an 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]
if opsiLicenseManagementEnabled
 comment "License management is enabled and will be used"

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

 DefVar $ServiceErrorClass$
 set $ServiceErrorClass$ = getLastServiceErrorClass
 comment "Error class: " + $ServiceErrorClass$

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


[Sub_check_exitcode]
comment "Test for installation success via exit code"
set $ExitCode$ = getLastExitCode
; informations to exit codes see
```

```
; 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_UNKNOWN_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
```

4.1.9.0.2 delsub.ins: ausgelagerte deinstallations Sektion

```
; 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
; (don't 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 .....)
```

#### 4.1.9.0.3 uninstall.ins: Deinstallationsscript

```
; 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]
if opsiLicenseManagementEnabled
      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("", "", $WindowsSoftwareId$)
else
      LogError "Error: licensing required, but license management not enabled"
      isFatalError
```

```
endif
```

### 4.1.10 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. If the opsi-client-agent is installed on your computer you will find the opsi-winst at the directory **C:\program files\opsi.org\preloginloader\opsi-winst**. If the opsi-client agent is not installed you will find the opsi winst at the share **\\<opsiserver\opt_pcbin** in the directory **install\opsi-winst\files**

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

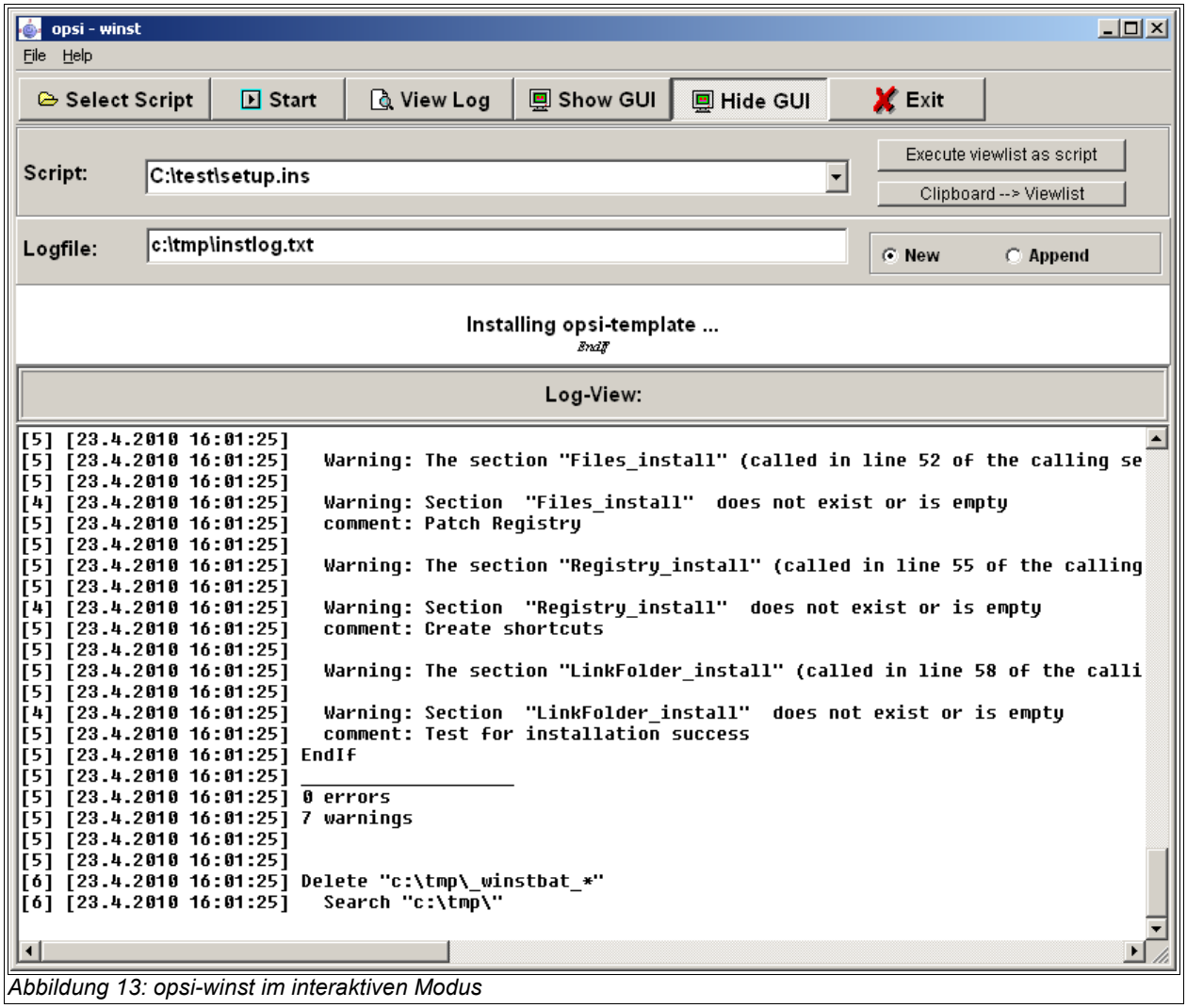


*Abbildung 13: opsi-winst im interaktiven Modus*

- 'Select Script' is used to choose the script that you want to execute
- 'Start' wil 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.

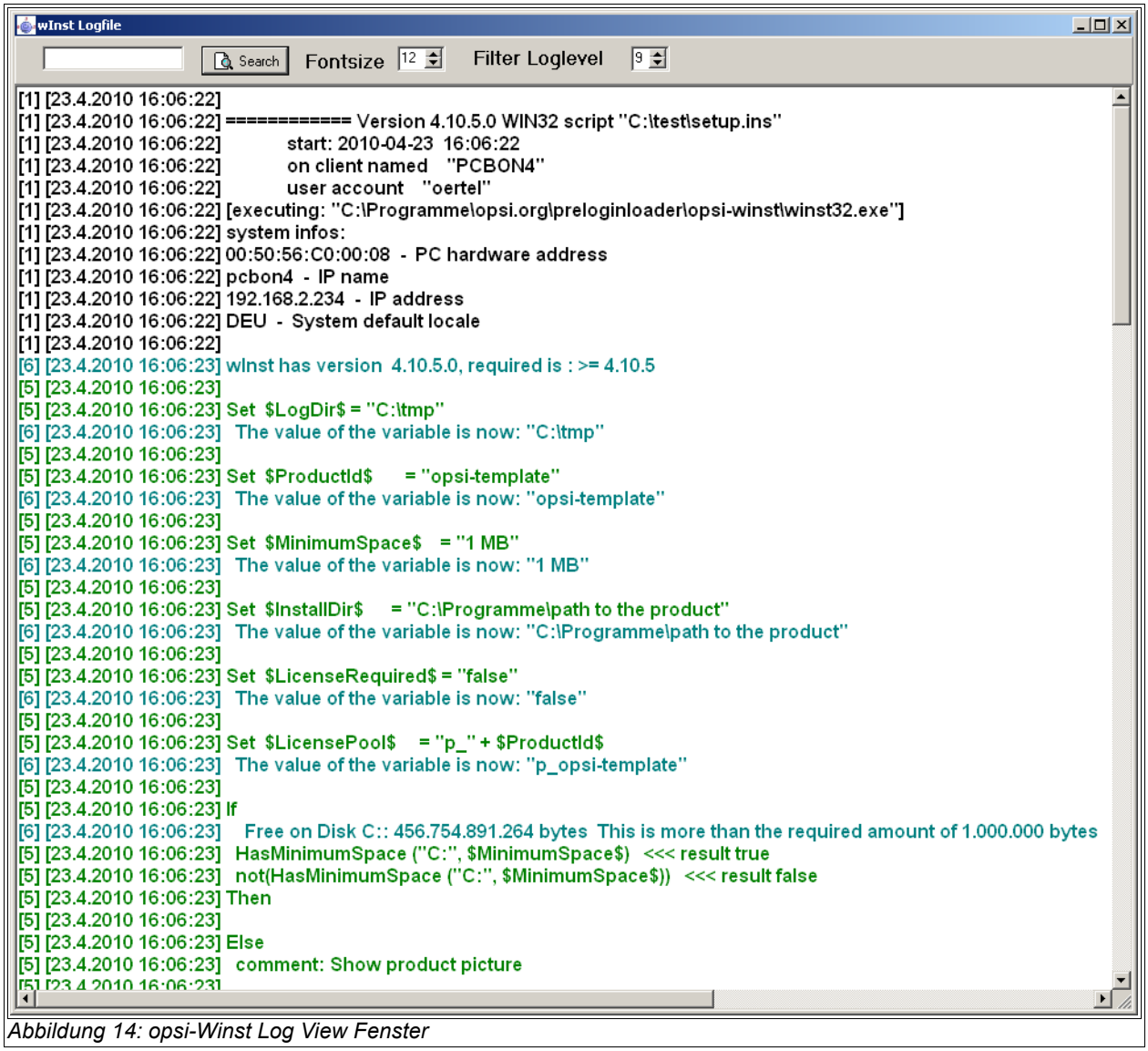


*Abbildung 14: opsi-Winst Log View Fenster*

Look at the log file and see how opsi-winst interpret the scrpt.

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.

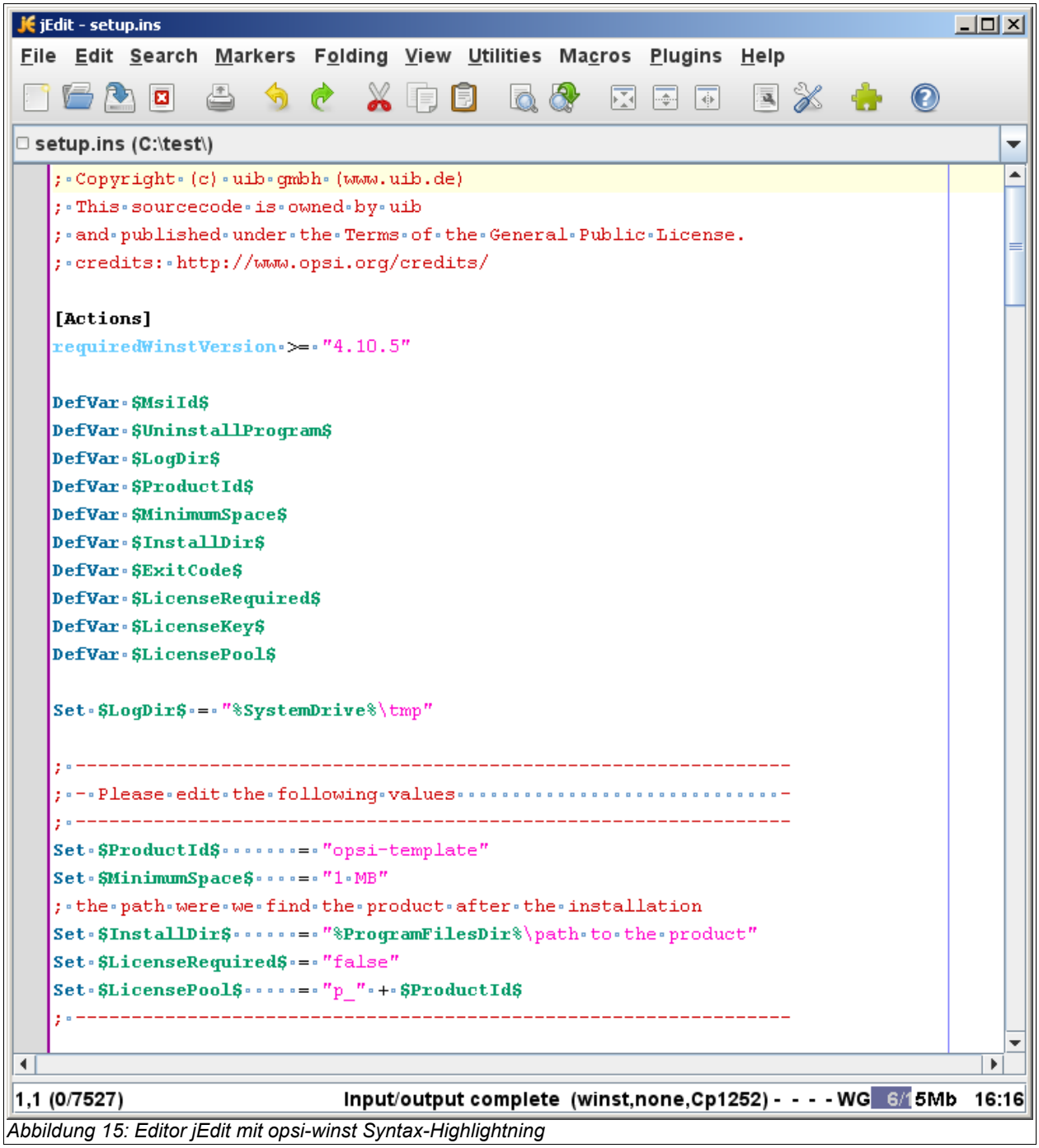


*Abbildung 15: Editor jEdit mit opsi-winst Syntax-Highlightning*

After any changes at the script - save it (you may let the editor window open). Change to the opsi-winst window and start the script again (you don't have to choose the script again). Look at the log file to see how your changes affected the run.

In this way, by iterating through the steps

- change and save the script
- start the script
- check the log file

you will be able to get a script which does, what do you want it to do.

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.

### 4.1.11 hints to detail problems

#### 4.1.11.1 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

##### 4.1.11.1.1 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:

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

A collection of links to web sites with switch collections is

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

##### 4.1.11.1.2 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.

##### 4.1.11.1.3 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/opsi_wiki/SoftwareIntegrationWebLinks

at the section 'Installer specific switches / Allg. Schalter der Setup Programme' more web links to sites which give hints how to detect the manufacturer of the setup program.

#### 4.1.11.2 More important opsi-winst commands

A short overview to the opsi-winst commands are the referencecard:
http://download.uib.de/opsi3.4/doku/opsi-winst-reference-card.pdf

All syntax details are described in the opsi-winst manual:
http://download.uib.de/opsi3.4/doku/winstdoc-en.pdf

Here are some hints to important elements:

##### 4.1.11.2.1 Stringlisten

- String lists are powerful, specially to review the output from other programs. Read the opsi-einst manual for details.

##### 4.1.11.2.2 ExitWindows

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

##### 4.1.11.2.3 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 the

chapter 4.2 Creating an opsi package.

To evaluate these properties opsi-winst got the function GetProductProperty

```
if GetProductProperty ("myproperty","off") = "on"
      Files_copy_extra_files
endif
```

#### 4.1.11.3 Installation with a logged on user

As a starting point we assume that you have done an unattended installation by using a wInst-script. The installation works OK when started as a logged on user (with administrative privileges).
But when started from within the software deployment (preloginloader) 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:

```
[Aktionen]
DefVar $LOG_LOCATION$
Set $LOG_LOCATION$ = "c:\tmp\myproduct.log"
winbatch_install_myproduct

[winbatch_install_myproduct]
msiexec /qb ALLUSERS=2 /l* $LOG_LOCATION$ /i %SCRIPTPATH%\files\myproduct.msi
```

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 wInst-handbook chapter 8.3 'Script for installation in the context of a local administrator' and use the template 'opsi-template-with-admin'.

#### 4.1.11.4 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
```

#### 4.1.11.5 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 chapter 'Analyze and repackage' further down. Some more tools can be found here:

http://www.sysinternals.com/

http://www.german-nlite.de/files/guides/regshot/regshot.html

#### 4.1.11.6 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.

#### 4.1.11.7 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

#### 4.1.11.8 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 afterwards. 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 wInst provides several commands for software deletion:

#### 4.1.11.8.1 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 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 myPacket.msi /qb-!
```

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 wInst-script as described in the following:

#### 4.1.11.8.2 Useful wInst commands for uninstall

If a product has been installed by wInst functions, or if there is no uninstall routine for the product, the complete deinstallation has to be done by a wInst script. WInst comes with some powerful uninstall functions. In this chapter we will have an overview, for detailed information refer to the wInst handbook.

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

```
delete -f filename
```

or to delete a directory including sub directories:

```
delete -sf dirname\
```

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 wInst-handbook for details).

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

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

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 <service name>
```

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 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 wInst command **DeleteKey**:

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

#### 4.1.11.9 Known issues at the 64 Bit support

The opsi installer 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, registry settings going to the 32 bit part of the registry and not to the 64 bit part.

New Versions of opsi-winst have special commands to handle these problems. So read the opsi-winst manual for these issues.

### 4.2 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>).

SERVER_DATA

Holds directories which will be unpacked to /. Root privileges for unpacking might be required.

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.

### 4.2.1 Create, pack and unpack a new product

In order to create a new opsi package you must login to 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).

***Attention: 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 preloginloader/wInst. Product type 'netboot' is used for products which are activated as a bootimage (like hardware inventory) and type 'server' is used for products that are just installed on a server.

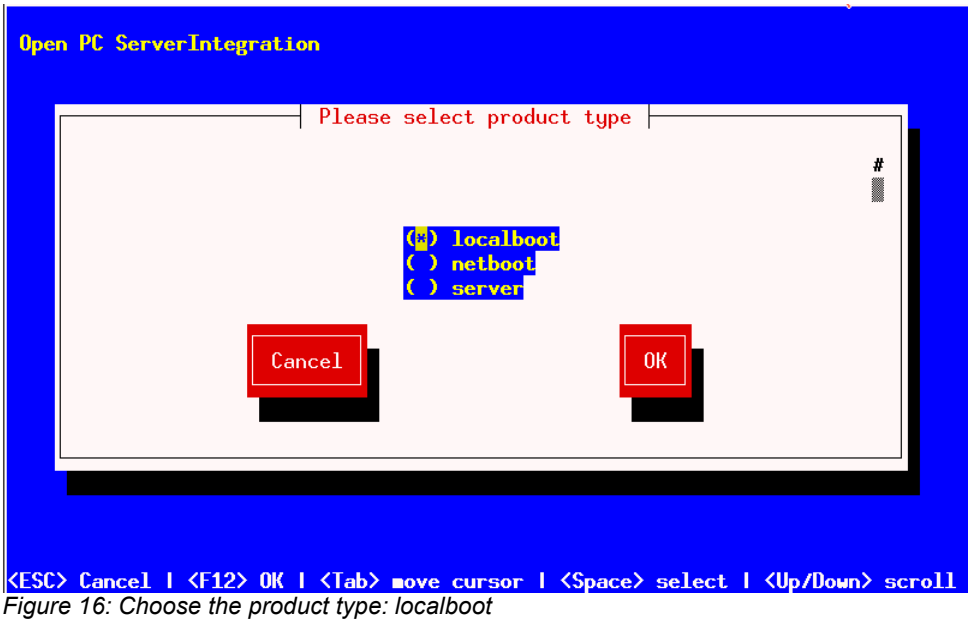
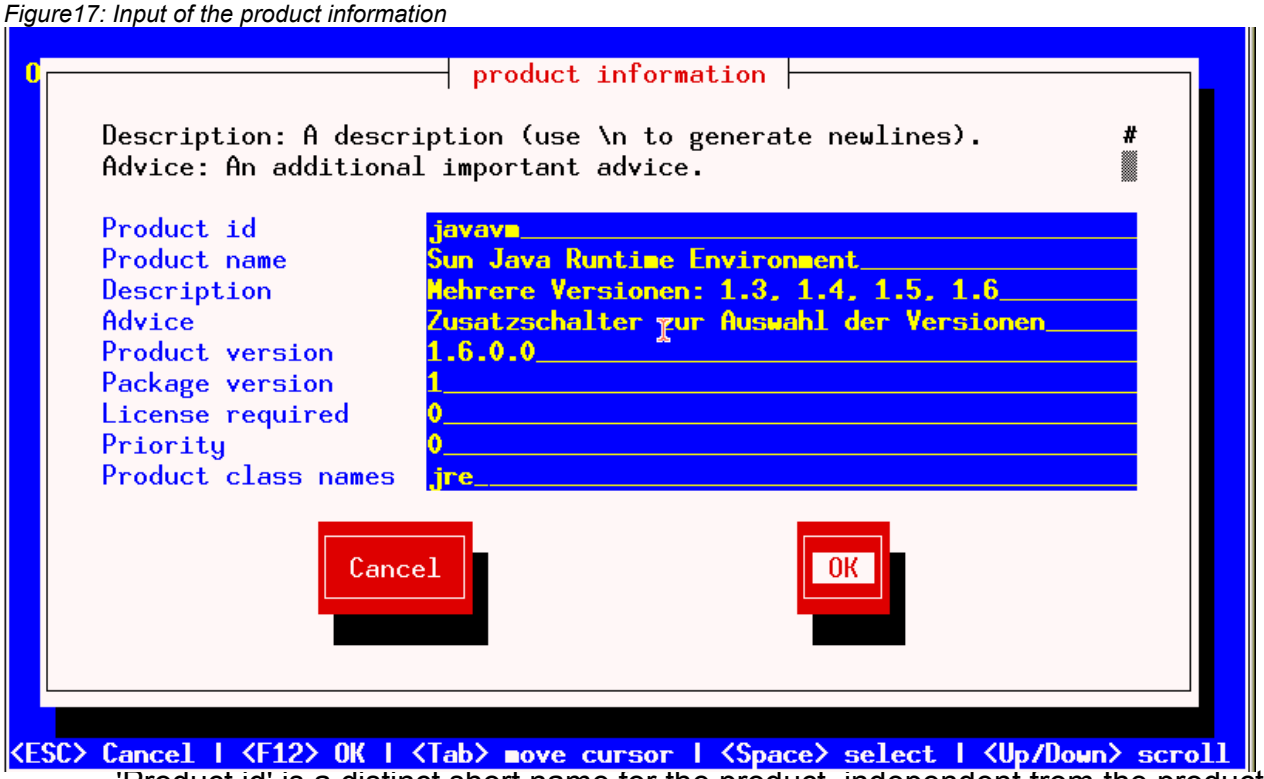



*Figure 16: Choose the product type: localboot*

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.

*Figure17: Input of the product information*




- 'Product id' is a distinct short name for the product, independent from the product version (in opsi V2 this has been the product name)
- '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.
- '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 wInst script.
- 'Priority' is for future use (regarding the installation order).
- 'Product class names' is for future use.

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

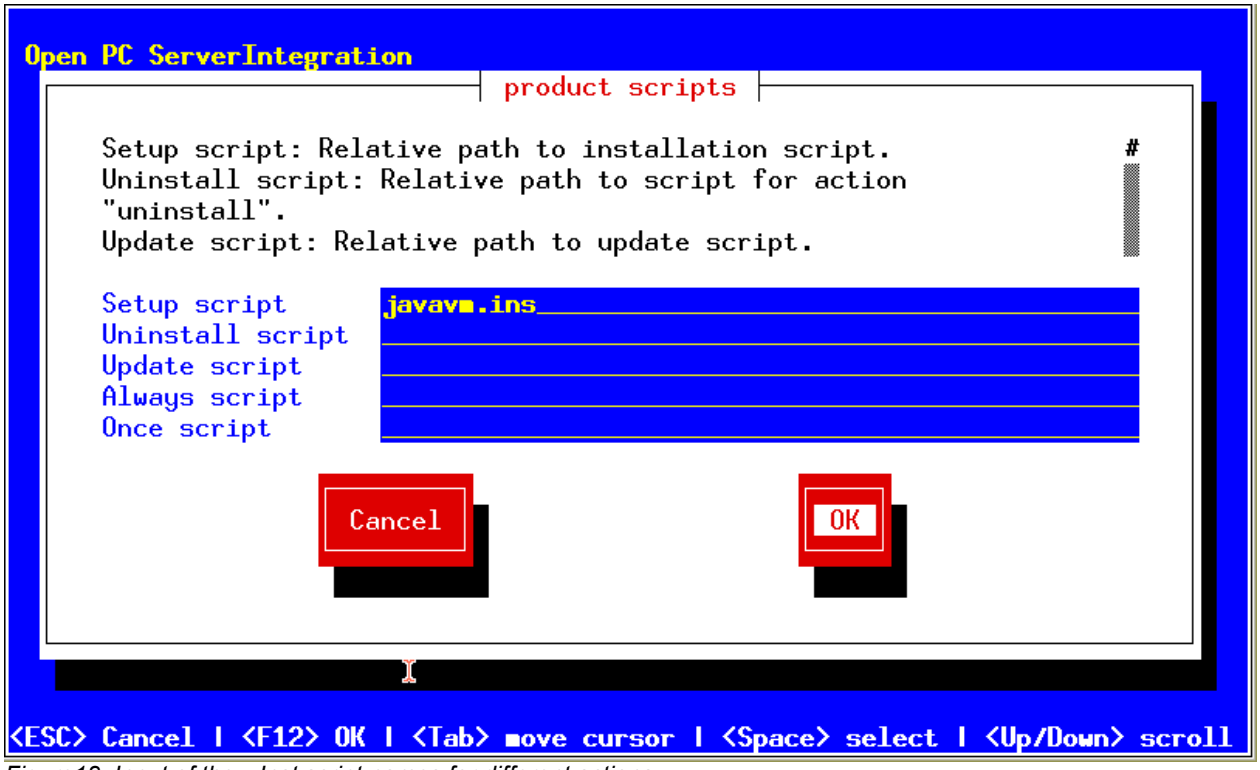



*Figure18: Input of the wInst script names for different actions*

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

| *Type* | *resulting state* | *resulting action* |
|---|---|---|
| setup | installed | none |
| uninstall | not_installed | none |
| update | installed | none |
| always | installed | always |
| once | not_installed | none |

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


*Figure 19: Create product dependency: No/Yes*

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

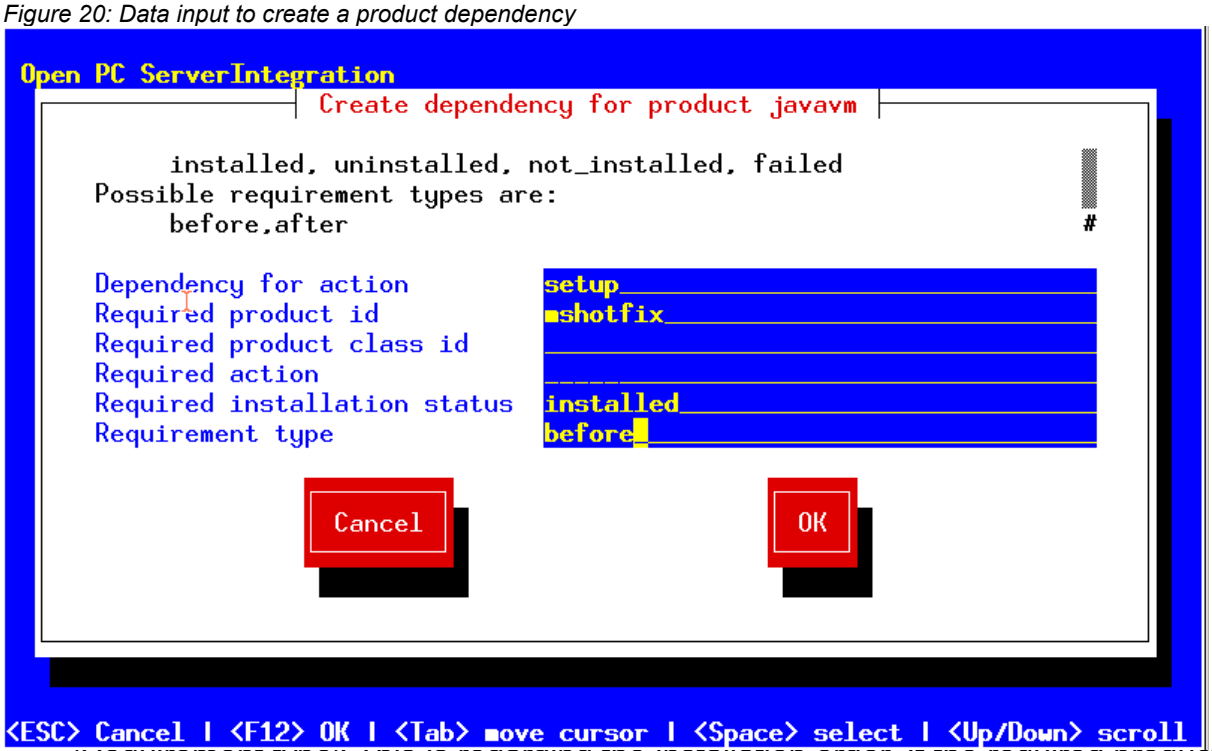
'Dependency for action' : For which product action shall the dependency be created (setup, deinstall,...)

- '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

*Figure 20: Data input to create a product dependency*




- '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.

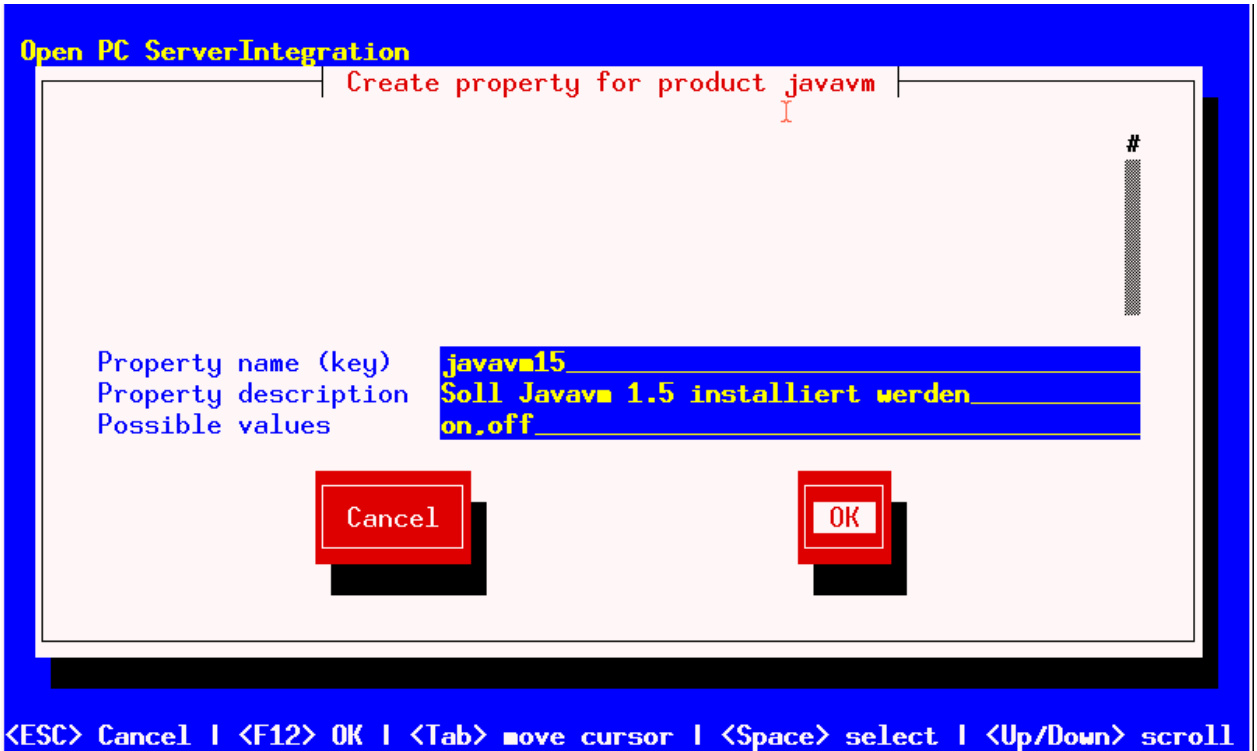
Notice: The possibiliy to define deinstall actions or dependecies is broken.

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.


*Figure 21: 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 wInst-script and result in installing



different options at installation time. 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.

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


*Figure 23: Default value of the product 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', 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:

```
[Product]
type: localboot
id: javavm
name: Sun Java Runtime Environment
description:
Mehrere Versionen: 1.3, 1.4, 1.5, 1.6
advice: Additional choice switch for different versions
version: 1.6.0.0
packageVersion: 1
priority: 0
licenseRequired: True
productClasses: jre
setupScript: javavm.ins
```

```
uninstallScript:
updateScript:
alwaysScript:
onceScript:

[ProductDependency]
action: setup
requiredProduct: mshotfix
requiredStatus: installed
requirementType: before

[ProductProperty]
name: default13
description: on=Version 1.3 are default JRE; off=The actual installed Javavm
are the default JRE
values: on, off
default: off

[ProductProperty]
name: javavm15
description: Should Javavm 1.5 be installed
values: on, off
default: on

[ProductProperty]
name: javavm16
description: Should Javavm 1.6 be installed
values: on, off
default: off
```

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

Then you can pack the package. Change to the root directory of the product and start 'opsi-makeproductfile'. The product will be packed.

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

```
# opsi-makeproductfile --help

Usage: opsi-makeproductfile [-h] [-v|-s] [-f] [-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
   -s          silent
   -l          log-level 0..6
   -f          fast, no topicality test
   -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)
  -t           temp dir
```

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

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

## 5. More Informations

A öot of more and more detailed informations you will find in the opsi manual: (http://download.uib.de/opsi3.4/doku/opsi-manual-v34-en.pdf)

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