Dokumentation

opsi
Version 3.3.1

open pc server integration

Handbuch

Stand: 03.04.2009

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

Inhaltsverzeichnis

1. Einführung

1.1. Für wen ist dieses Handbuch?

Diese Handbuch richtet sich an alle, die sich näher für die automatische Softwareverteilung opsi interessieren. Der Schwerpunkt der Dokumentation ist die Erläuterung der technischen Hintergründe, um so zu einem Verständnis der Abläufe beizutragen.

Damit soll dieses Handbuch nicht nur den praktisch mit opsi arbeitenden Systemadministrator unterstützen sondern auch im Vorfeld den Interessenten einen konkreten Überblick über opsi geben.

1.2. Konventionen zu Schrift und Grafiken

In <spitzen Klammern> werden Namen dargestellt, die im realen Einsatz durch ihre Bedeutung ersetzt werden müssen.
Beispiel: Der Fileshare, auf dem die opsi Softwarepakete liegen, wird <opsi-depot-share> genannt und liegt auf einem realen Server z.B. auf /opt/pcbin/install.
Das Softwarepaket: <opsi-depot-share>/ooffice liegt dann tatsächlich unter /opt/pcbin/install/ooffice.

Beispiele aus Programmcode oder Konfigurationsdateien stehen in Courier-Schrift und sind grau hinterlegt .

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

2. Überblick opsi

Werkzeuge zur automatischen Softwareverteilung und Betriebssysteminstallation sind bei größeren PC-Netz-Installationen ein wichtiges Werkzeug zur Standardisierung, Wartbarkeit und Kosteneinsparung. Während die Verwendung solcher Werkzeuge für gewöhnlich mit erheblichen Linzenzkosten einher geht, bietet opsi als Opensource Werkzeug deutliche Kostenvorteile. Hier fallen nur die Kosten an, die von Ihnen durch tatsächlich angeforderte Dienstleistungen, wie Beratung, Schulung und Wartung, entstehen.

Auch wenn Software und Handbücher kostenlos sind, ist es die Einführung eines Softwareverteilungswerkzeuges nie. Um die Vorteile ohne Rückschläge und langwierige Lernkurven nutzen zu können, ist die Schulung und Beratung der Systemadmistratoren durch einen erfahrenen Partner dringend geboten. Hier bietet Ihnen uib seine Dienstleistungen rund um opsi an.

Das von uib entwickelte System basiert auf Unix-/Linux-Servern, über die das Betriebssystem und Software-Pakete auf den PC-Clients installiert und gewartet werden (PC-Server-Integration). Es basiert weitestgehend auf frei verfügbaren Werkzeugen (*GNU*-tools, *SAMBA* etc.). Dieses **opsi** (Open PC-Server-Integration) getaufte System ist durch seine Modularität und Konfigurierbarkeit in großen Teilen eine interessante Lösung für die Probleme der Administration eines großen PC-Parks.

2.1. Erfahrung

opsi ist die Fortschreibung eines Konzepts, das seit Mitte der 90er Jahre bei einer Landesverwaltung auf über 2000 Clients in verschiedenen Lokationen kontinuierlich im Einsatz ist und stetig weiterentwickelt wurde. Als Produkt opsi ist es nun auch einem breiten Kreis von Interessenten zugänglich.
Eine Übersicht registrierter opsi-Installationen finden Sie unter: http://www.opsi.org/map/

2.2. Features von opsi

Die wesentlichen Features von opsi sind:

- automatische Softwareverteilung
- automatische Betriebssysteminstallation
- Hard- und Softwareinventarisierung mit Historyfunktion
- Komfortable Steuerung über das opsi Managementinterface
- Unterstützung von mehreren depot-servern

Die Funktionalität von opsi basiert dabei auf dem opsi-server welcher die serverseitigen Dienste zur Verfügung stellt.

2.3. Was ist neu in opsi 3.3.1

- Unterstützung für Debian Lenny
- Überarbeitete Pakete zur Betriebssysteminstallation
  Darin:
  - Vereinheitlichter Aufbau der Pakete
  - Unterstützung bei der automatisierten Treiberintegration für HD-Audio- und USB-Treiber
  - Unterstützung mehrere i386 Zweige
- Für unsere Vista-Support Kunden:
  - Verbesserte Unterstützung der 64-Bit Versionen von Vista/2008
  - Neuer Preloginloader 3.4 nicht nur für Vista
  - Pakete für Vista Service-Packs und Hotfixes
- Auch bei Netbootprodukten ist gibt es nun die Möglichkeit von 'always' als angeforderte Aktion
- Neue Virtual Maschine auf Basis von Debian Lenny
- Aktualisiertes Installations- und opsi-Handbuch

- Diverse Bugfixes
- Die Unterstützung des opsi 2.x/3.0 File-Backends ist eingestellt.
    - Anwender dieser Versionen sollten dringend **vor dem Update auf opsi 3.3.1** zu dem File31-Backend migrieren.

2.4. Was sollten Sie bei einem Upgrade auf opsi 3.3.1 unbedingt lesen

In diesem Handbuch:

•3.4 Werkzeug opsi-package-manager: opsi-Pakete (de-) installieren Seite 25

•4.2 opsi Standardprodukte Seite 41

•5.1.8 Aufbau der Produkte zur unattended Installation Seite 88

•5.1.9 Vereinfachte Treiberintegration in die automatische Windowsinstallation Seite 91

•10 Anpassen des preloginloaders an die Corporate Identity (CI) Seite 121

Im opsi-Vista-Installationshandbuch:

- preloginloader 3.4

3. opsi-Konfiguration und Werkzeuge

3.1. Übersicht

Die Konfiguration von opsi benötigt eine Datenhaltung. Während in opsi Version 2 nur eine Datei basierte Datenhaltung möglich war sind ab Version 3 verschiedene Datenhaltungen möglich. Während die alten Werkzeuge direkt auf den Dateien gearbeitet haben (und bei Verwendung des 'File' Backends auch weiter Ihren Dienst tun) kommunizieren die neuen Werkzeuge mit dem opsiconfd über einen Webservice. Der opsiconfd übergibt die Daten dem Backendmanager der die Daten in das Konfigurierte Backend schreibt. Mehr zur Datenhaltung finden Sie im Kapitel 'Datenhaltung von opsi'.

Per Default wird ein File31-Backend angelegt.

3.2. Werkzeug: opsi V3 opsi-Configed

3.2.1. Voraussetzungen und Aufruf

Der opsi-Configed setzt Java 1.6 voraus und benötigt einen laufenden opsiconfd auf der Serverseite.

Der opsi-Configed ist Bestandteil des Clientproduktes 'opsi-adminutils' und kann über die entsprechende Gruppe im Startmenü gestartet werden.

Serverseitig wird der opsi-configed als Debianpaket (opsi-configed.xxxxx.deb) installiert und ist über einen Menüeintrag im Desktopmenü sowie über **`/usr/bin/opsi-configed`** aufrufbar.

Der Aufruf kann auch erfolgen über **`java -jar configed.jar`**.

Der Aufruf **`java -jar configed.jar --help`** zeigt die Kommandozeilenoptionen.

```
P:\install\opsi-adminutils>java -jar configed.jar --help
starting configed
default charset is windows-1252
server charset is configured as UTF-8

configed [OPTIONS]...
```

```
Options:
     -l, --locale    Set locale (format: <language>_<country>)
     -h, --host      Configuration server to connect to
     -u, --user      Username for authentication
     -p, --password  Password for authentication
     -d, --logdirectory Directory for the log files
         --help      Show this text
```

Soll ein anderer Port als der Standardport 4447 verwendet werden, so kann beim Hostnamen der Port mit angegeben werden in der Form: host:port.

3.2.2. Login

Abbildung 1: opsi-Configed: Loginmaske

Beim Login versucht sich der opsi-configed per https auf den Port 4447 (opsiconfd) zu verbinden. Das Login geschieht unter Angabe des Servernamens[:Port], des usernamens auf dem opsi-config-server und dessen Unix-Passwortes. Damit der Login erfolgreich ist muss der user in der Unix-Gruppe opsiadmin sein.

3.2.3. Depotauswahl

Unterstützt Ihr opsi-server mehrere Depots, so werden diese Links oben im configed angezeigt. Per default ist das Depot auf dem config-server markiert und die zu diesem Depot gehörigen Clients werden angezeigt. Sie können mehrere Depots gleichzeitig markieren und deren Clients gleichzeitig bearbeiten - allerdings nur dann, wenn die unterschiedlichen Depots synchron zum Depot auf dem config-server sind. Nachdem Sie die Auswahl der Depots geändert haben, müssen Sie den Refresh-Button klicken.

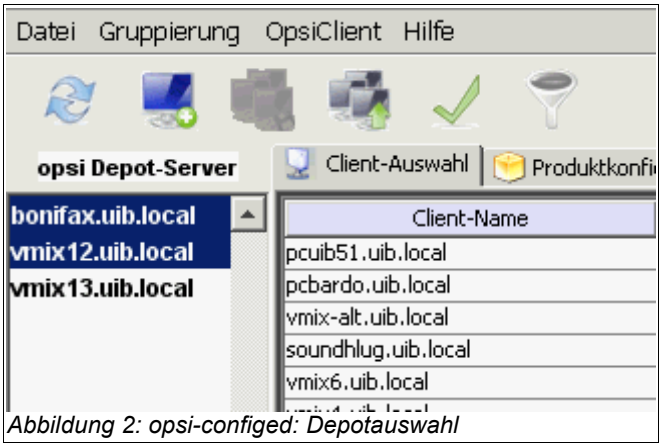


Abbildung 2: opsi-configed: Depotauswahl

Dieses Feature wird nur im Rahmen eines entsprechenden Supportvertrages unterstützt.

3.2.4. Clientauswahl und Gruppenbildung

Nach erfolgreichem Login zeigt sich das Hauptfenster mit dem aktiviertem Karteireiter 'Client-Auswahl'. In der Client-Auswahl sehen Sie die Liste der bekannten Clients mit den Spalten 'Client-Name', 'Beschreibung' und 'Zuletzt gesehen'.

- 'Client-Name' ist der 'full qualified hostname' also der Clientname mit Domainnamen.
- 'Beschreibung' ist eine frei wählbare Beschreibung die im rechten oberen Teil des Fensters editiert werden kann.
- 'Zuletzt gesehen' gibt Datum und Uhrzeit an zu der der Client zum letzten mal sich bei der Softwareverteilung (über den Webservice) gemeldet hat.

Die Clientliste lässt sich durch anklicken der Spaltentitel nach der entsprechenden Spalte sortieren.

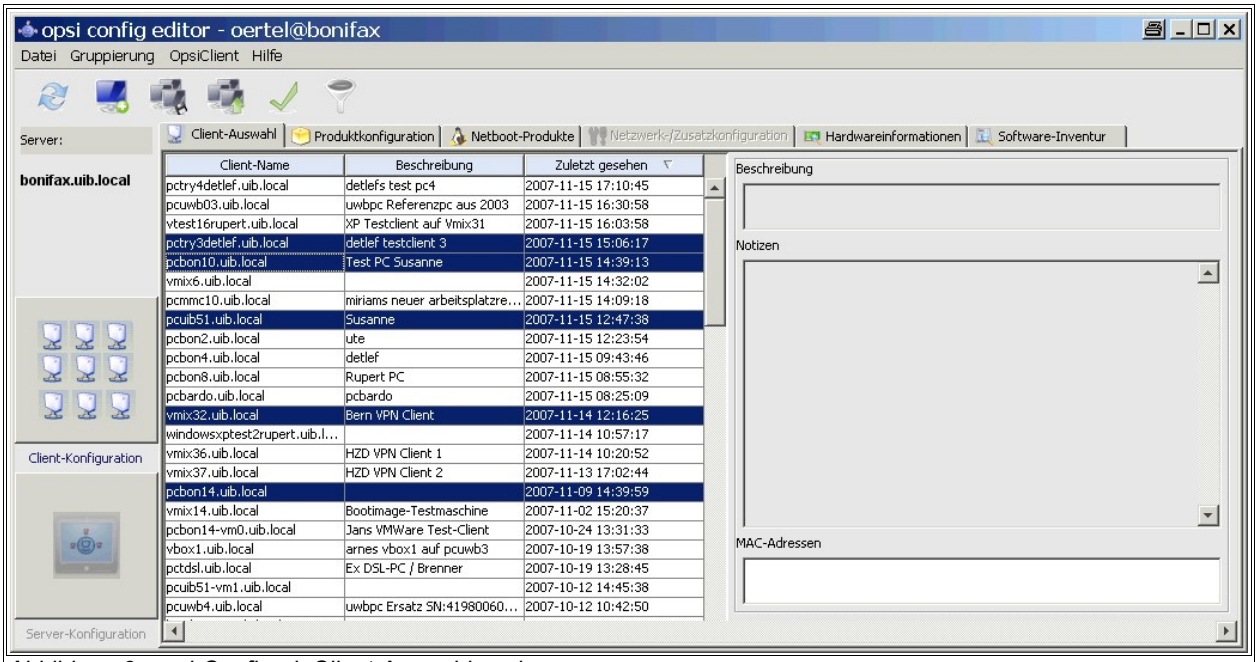


Abbildung 3: opsi-Configed: Client Auswahlmaske

Es lassen sich ein oder mehrere Clients markieren und gemeinsam bearbeiten. Die Ansicht der Clients lässt sich durch das Trichter-Icon bzw. über 'Gruppierung / Nur ausgewählte Clients anzeigen' auf die markierten Clients beschränken.

Eine markierte Gruppe von Clients lässt sich über das Icon 'Gruppe sichern' bzw. über 'Gruppierung / Diese Gruppe abspeichern' unter einem frei wählbaren Namen speichern.

Abbildung 4: opsi-Configed: Maske: Gruppe setzen

Über das Icon 'Gruppe setzen' bzw. über Gruppierung / 'Gruppe setzen' lassen sich abgespeicherte Gruppen wieder laden.

Über die Funktion 'Gruppe setzen' lassen sich auch Clientgruppen aufgrund aktuell vor liegender Bedingungen wie z. B. alle Clients, bei denen das Produkt 'acroread' den Installationsstatus 'installed' hat, bilden.

3.2.5. Client Bearbeitung / WakeOnLan / Client erstellen / Client umziehen

Haben sie einen oder mehrere Clients gewählt, so können Sie diesen über den Menüpunkt OpsiClient ein 'WakeOnLan' Signal senden. Sie haben hier auch die Möglichkeit, Clients zu löschen oder neu anzulegen.

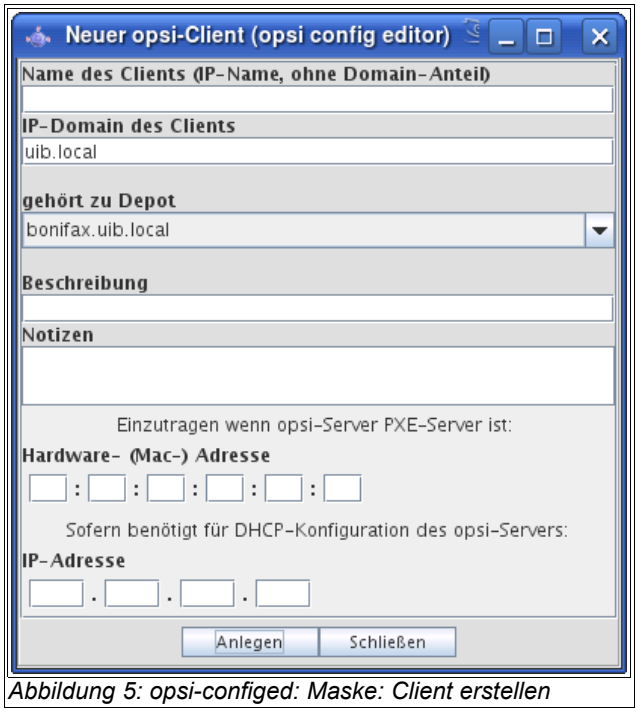


Abbildung 5: opsi-configed: Maske: Client erstellen

Über den Menü-Punkt 'Neuen OpsiClient erstellen', erhalten Sie eine Maske zur Eingabe der nötigen Informationen zur Erstellung eines Clients.

Diese Maske enthält auch Felder für die optionale Angabe der IP-Nummer und der Hardware- (MAC)-Adresse. Wenn das Backend für die Konfiguration eines lokalen dhcp-Servers aktiviert ist (dies ist nicht der default), werden diese Informationen genutzt, um den neuen Client auch dem dhcp-Server bekannt zu machen. Ansonsten

wird die MAC-Adresse im File31-Backend in der <pcname>.ini gespeichert und die IP-Nummer verworfen.

Client zu einem anderen Depot umziehen
(nur supported im Rahmen eines entsprechenden Supportvertrages)

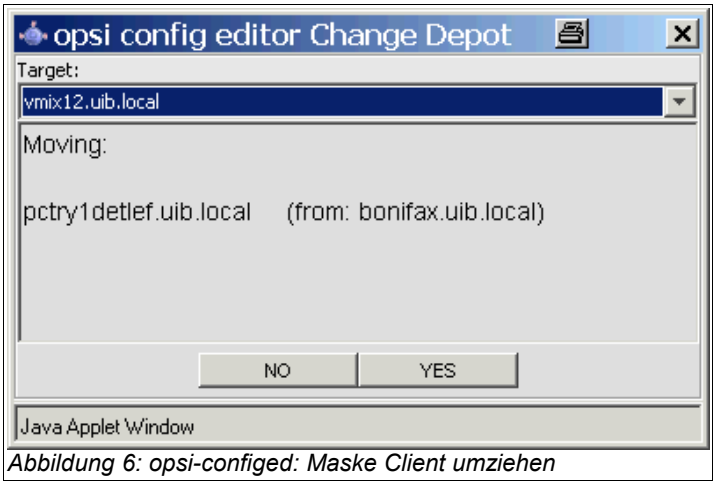


Abbildung 6: opsi-configed: Maske Client umziehen

3.2.6. Produktkonfiguration

Wechseln Sie auf den Karteireiter 'Produktkonfiguration' so erhalten Sie die Liste der zur Softwareverteilung bereit stehenden Produkte und den Installations- und Aktionsstatus zu den ausgewählten Clients. Stati die für die ausgewählten Clients unterschiedlich sind werden grau ('undefined') angezeigt. Die Liste der ausgewählten Clients wird rechts oben angezeigt. Sie können auch hier die Produktliste durch anklicken der Spaltentitel sortieren lassen.

- 'Installationsstatus' ist der letzte der Softwareverteilung gemeldete Status zu diesem Produkt und kann die Werte 'installed', 'not installed', 'installing', 'undefined' und 'failed' einnehmen. 'Failed' bedeutet das ein Installationsskript eine gescheiterte Installation gemeldet hat. 'Undefined' bedeutet das die Stati bei den gewählten Clients unterschiedlich sind. 'installing' ist der Produktstatus während der Produktinstallation.

- 'Anstehende Aktion' ist die Aktion die beim nächsten Boot ausgeführt werden soll. Mögliche Werte sind immer 'none' (keine Anzeige) und 'undefined'. Darüber

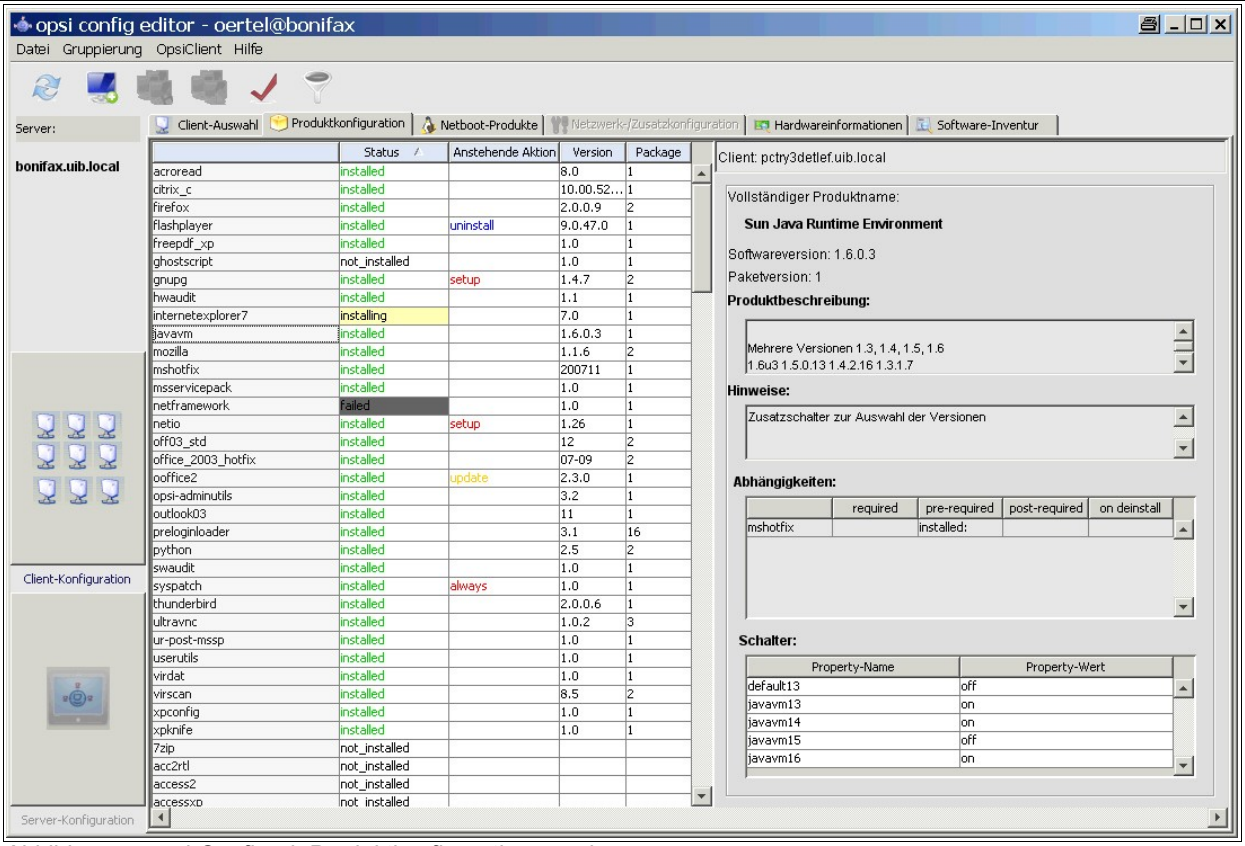


Abbildung 7: opsi-Configed: Produktkonfigurationsmaske

hinaus die Aktionen für die für diese Produkt Skripte hinterlegt wurden: 'setup', 'deinstall', 'once' 'always'.

- 'Version' ist die Versionsnummer der auf dem Client installierten Software so wie sie im entsprechenden opsi-Paket angegeben war.
- 'Package' ist die Paket-Nummer des opsi-Paketes der auf dem Client installierten Software so wie sie im entsprechenden opsi-Paket angegeben war.

Durch das Anwählen eines Produktes erhalten Sie auf der rechten Seite des Fensters weitere Informationen zu diesem Produkt:

'Vollständiger Produktname': Klartextname des Produktes

'Softwareversion': Die Versionsnummer der zur Verteilung bereitstehenden Software (wie sie der Paketierer angegeben hat).

'Paketversion': Version des Pakets zu der obenstehenden Softwareversion

'Produktbeschreibung': Freier Text zur im Paket enthaltenen Software.

'Hinweise': Freier Text mit Angaben zum Umgang mit diesem Paket.

'Abhängigkeiten': Eine Liste von Produkten zu denen das ausgewählte Produkt Abhängigkeiten aufweist so wie die Angabe der Art der Abhängigkeit. 'required' bedeutet dabei das ausgewählte Produkt benötigt das angezeigte Produkt es besteht aber keine notwendige Installationsreihenfolge. 'pre-required' bedeutet das angezeigte Produkt muss vor dem ausgewählten installiert werden. 'post-required' bedeutet das angezeigte Produkt muss nach dem ausgewählten installiert werden. 'on deinstall' bedeutet diese Aktion soll bei der Deinstallation des ausgewählten Produktes durchgeführt werden.

'Schalter': zur Clientspezifischen Anpassung der Installation können für ein Produkt zusätzliche Schalter definiert sein. Hier sehen Sie die Liste der zur Verfügung stehenden Schalter. Die Bedeutung der Schalter wird in einem Hinweis angezeigt wenn Sie mit dem Mauszeiger über den Schalternamen fahren. Unter Value bekommen Sie eine Liste der erlaubten Werte für diesen Schalter. Falls nicht so hat der Paketierer keine Werteliste angegeben und die Eingabe ist frei.

3.2.7. Netboot-Produkte

Die Produkte unter dem Karteireiter 'Netboot-Produkte' werden analog zum Karteireiter 'Produktkonfiguration' angezeigt und konfiguriert.

Die hier angeführten Produkte versuchen, werden sie auf setup gestellt, zu den ausgewählten Clients den Start von bootimages beim nächsten Reboot festzulegen.

Dies dient üblicherweise der OS-Installation.

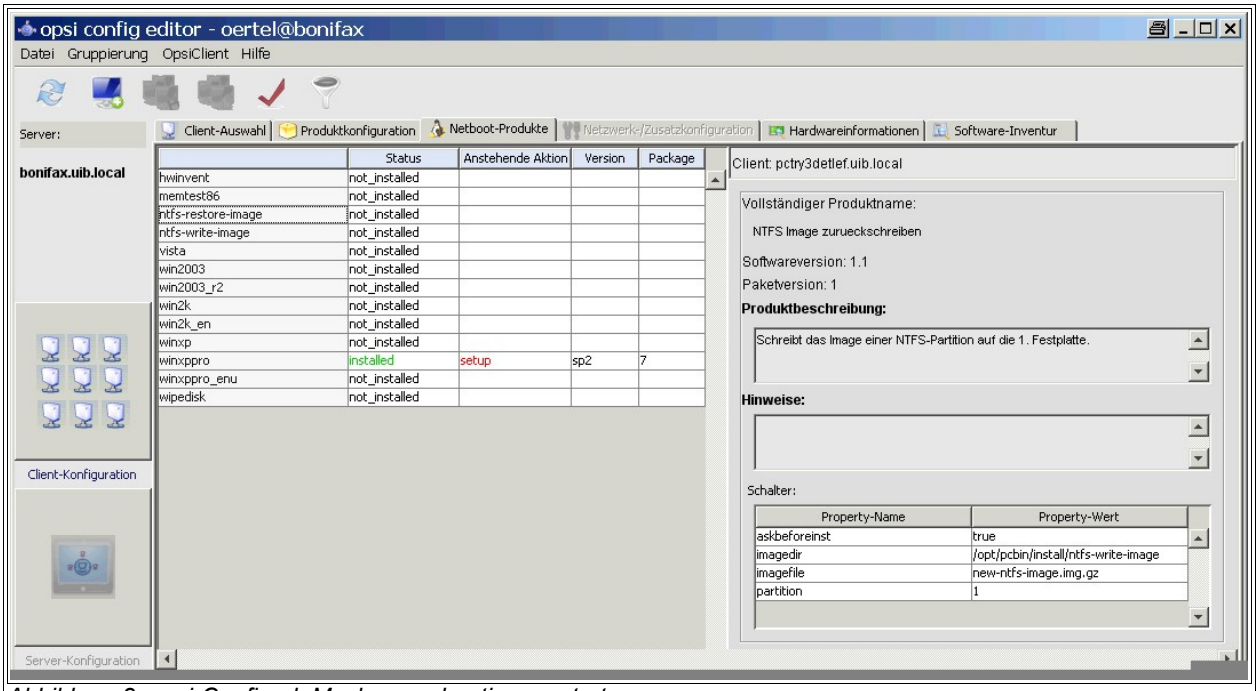


Abbildung 8: opsi-Configed: Maske zum bootimage starten

3.2.8. Hardwareinformationen

Unter diesem Karteireiter erhalten Sie die letzten gemeldeten Hardwareinformationen zu diesem Client.

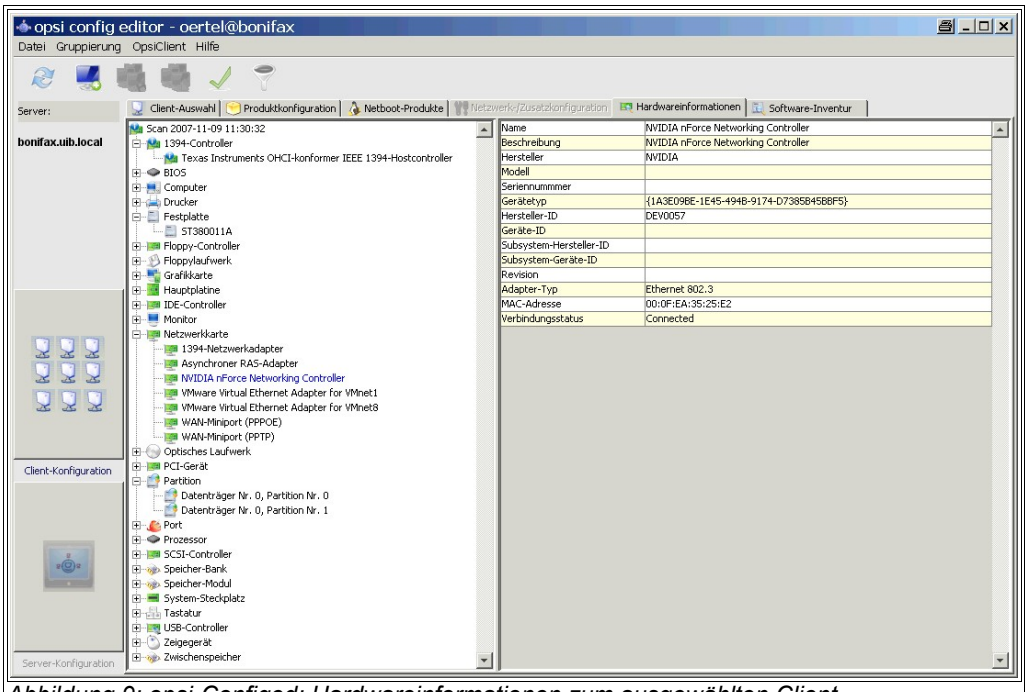
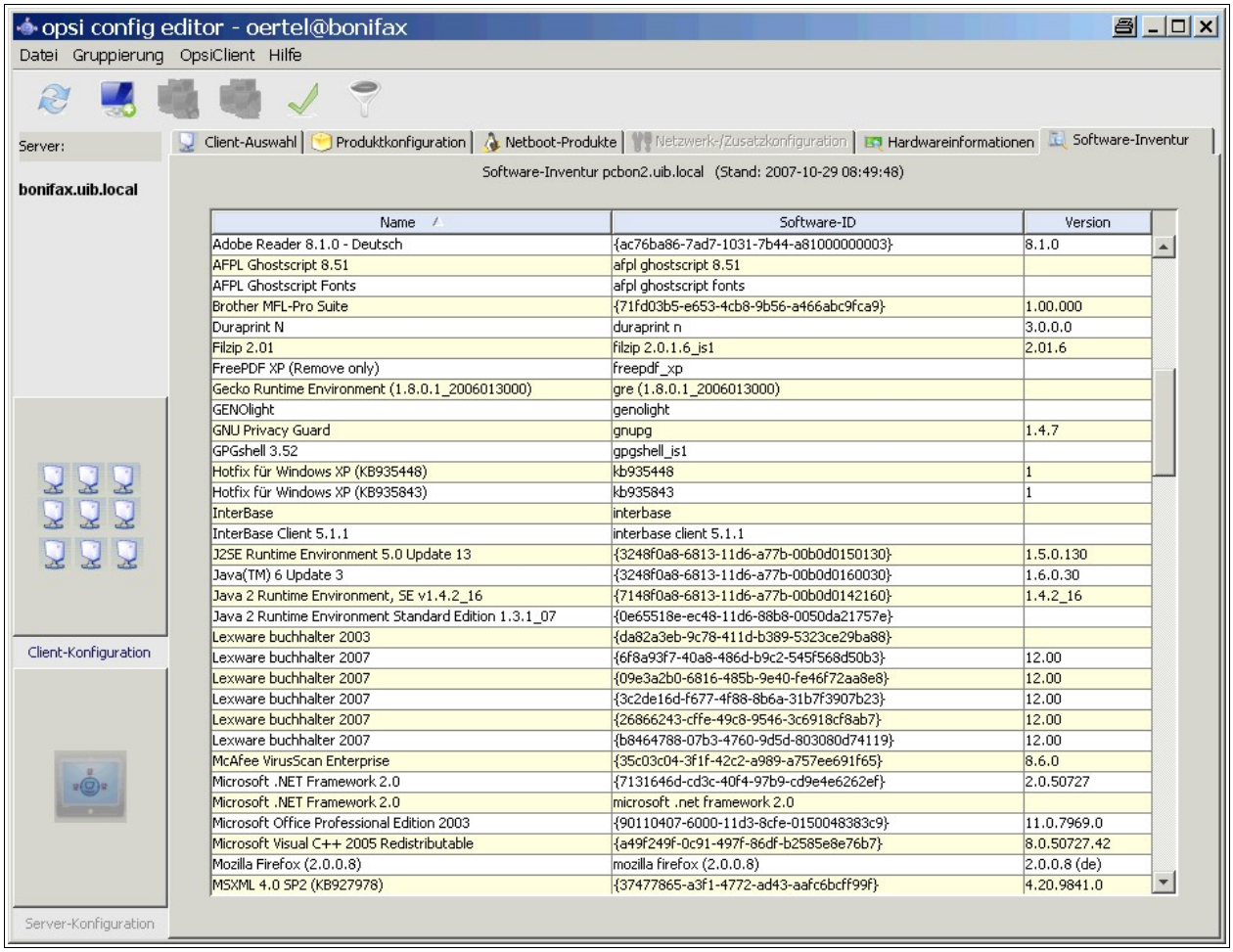


Abbildung 9: opsi-Configed: Hardwareinformationen zum ausgewählten Client

3.2.9. Software Inventur

Unter diesem Karteireiter erhalten Sie die letzten mit swaudit ausgelesene Information über installierte Software zu diesem Client.



Name	Software-ID	Version
Adobe Reader 8.1.0 - Deutsch	{ac76ba86-7ad7-1031-7b44-a81000000003}	8.1.0
AFPL Ghostscript 8.51	afpl ghostscript 8.51	
AFPL Ghostscript Fonts	afpl ghostscript fonts	
Brother MFL-Pro Suite	{71fd03b5-e653-4cb8-9b56-a466abc9fca9}	1.00.000
Duraprint N	duraprint n	3.0.0.0
Filzip 2.01	filzip 2.0.1.6_is1	2.01.6
FreePDF XP (Remove only)	freepdf_xp	
Gecko Runtime Environment (1.8.0.1_2006013000)	gre (1.8.0.1_2006013000)	
GENOlight	genolight	
GNU Privacy Guard	gnupg	1.4.7
GPGshell 3.52	gpgshell_is1	
Hotfix für Windows XP (KB935448)	kb935448	1
Hotfix für Windows XP (KB935843)	kb935843	1
InterBase	interbase	
InterBase Client 5.1.1	interbase client 5.1.1	
J2SE Runtime Environment 5.0 Update 13	{3248f0a8-6813-11d6-a77b-00b0d0150130}	1.5.0.130
Java(TM) 6 Update 3	{3248f0a8-6813-11d6-a77b-00b0d0160030}	1.6.0.30
Java 2 Runtime Environment, SE v1.4.2_16	{7148f0a8-6813-11d6-a77b-00b0d0142160}	1.4.2_16
Java 2 Runtime Environment Standard Edition 1.3.1_07	{0e65518e-ec48-11d6-88b8-0050da21757e}	
Lexware buchhalter 2003	{da82a3eb-9c78-411d-b389-5323ce29ba88}	
Lexware buchhalter 2007	{6f8a93f7-40a8-486d-b9c2-545f568d50b3}	12.00
Lexware buchhalter 2007	{09e3a2b0-6816-485b-9e40-fe46f72aa8e8}	12.00
Lexware buchhalter 2007	{3c2de16d-f677-4f88-8b6a-31b7f3907b23}	12.00
Lexware buchhalter 2007	{26866243-cffe-49c8-9546-3c6918cf8ab7}	12.00
Lexware buchhalter 2007	{b8464788-07b3-4760-9d5d-803080d74119}	12.00
McAfee VirusScan Enterprise	{35c03c04-3f1f-42c2-a989-a757ee691f65}	8.6.0
Microsoft .NET Framework 2.0	{7131646d-cd3c-40f4-97b9-cd9e4e6262ef}	2.0.50727
Microsoft .NET Framework 2.0	microsoft .net framework 2.0	
Microsoft Office Professional Edition 2003	{90110407-6000-11d3-8cfe-0150048383c9}	11.0.7969.0
Microsoft Visual C++ 2005 Redistributable	{a49f249f-0c91-497f-86df-b2585e8e76b7}	8.0.50727.42
Mozilla Firefox (2.0.0.8)	mozilla firefox (2.0.0.8)	2.0.0.8 (de)
MSXML 4.0 SP2 (KB927978)	{37477865-a3f1-4772-ad43-aafc6bcff99f}	4.20.9841.0

Abbildung 10: opsi-Configed: Softwareinformationen zum ausgewählten Client

3.2.10. Logdateien: Logs von Client und Server

Ab dem Update vom 1.9.08 werden unter opsi 3.3 die Logdateien auch der Clients zentral auf dem Server abgelegt und sind über den opsi-configed einsehbar.

Dabei kann auch in den Logdateien gesucht werden (Fortsetzung der Suche mit 'F3' oder 'n'.

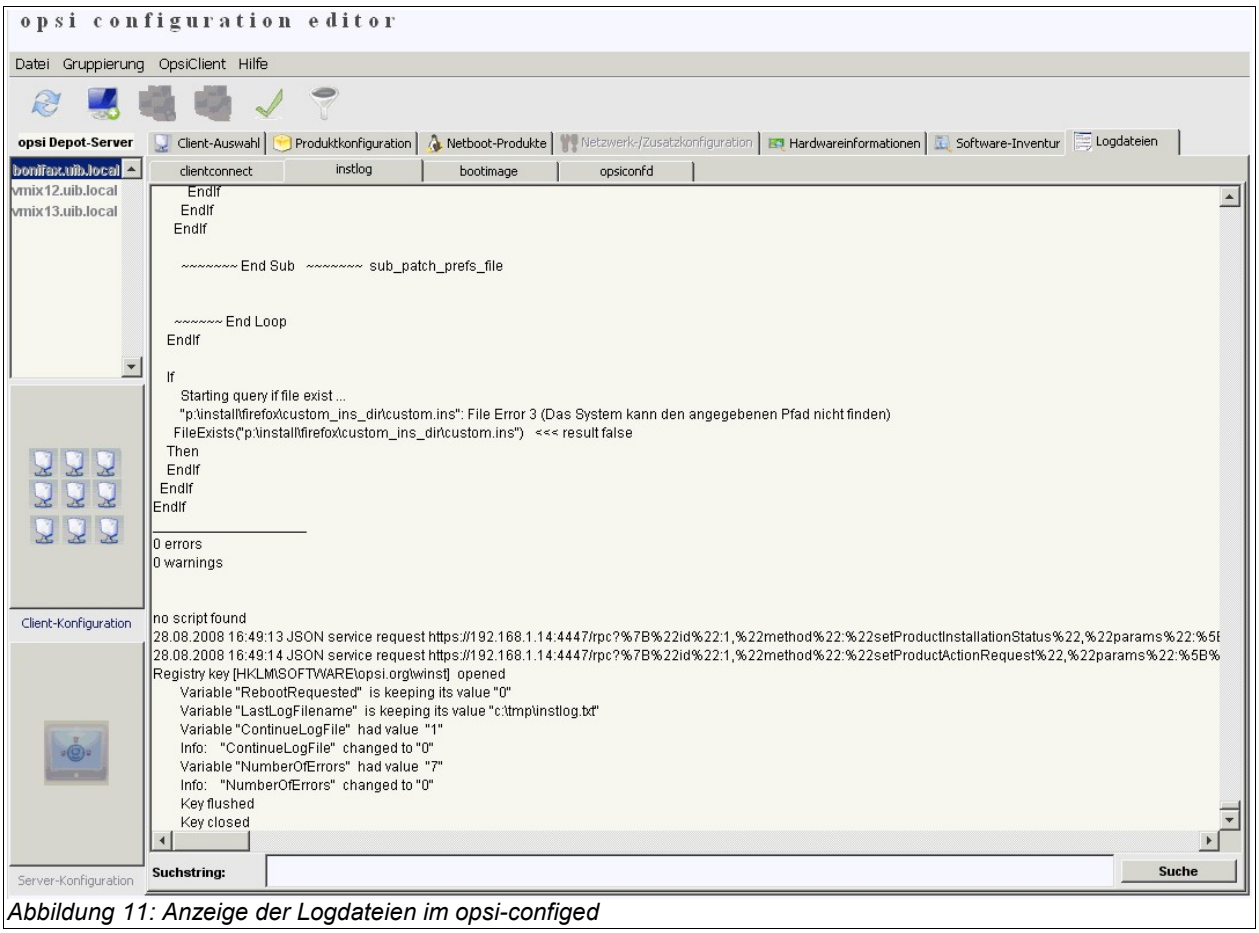


Abbildung 11: Anzeige der Logdateien im opsi-configed

3.2.11. Serverkonfiguration: Netzwerk- und Zusatzkonfiguration

Über den Karteireiter 'Netzwerk-/Zusatzkonfiguration' können Einstellungen zur Netzwerkkonfiguration von opsi und weiteren Optionalen Konfigurationen vorgenommen werden. Die Bedeutung der möglichen Einträge ist im Kapitel über die Dateien des Filebackends / File31 / <pcname>.ini beschrieben.

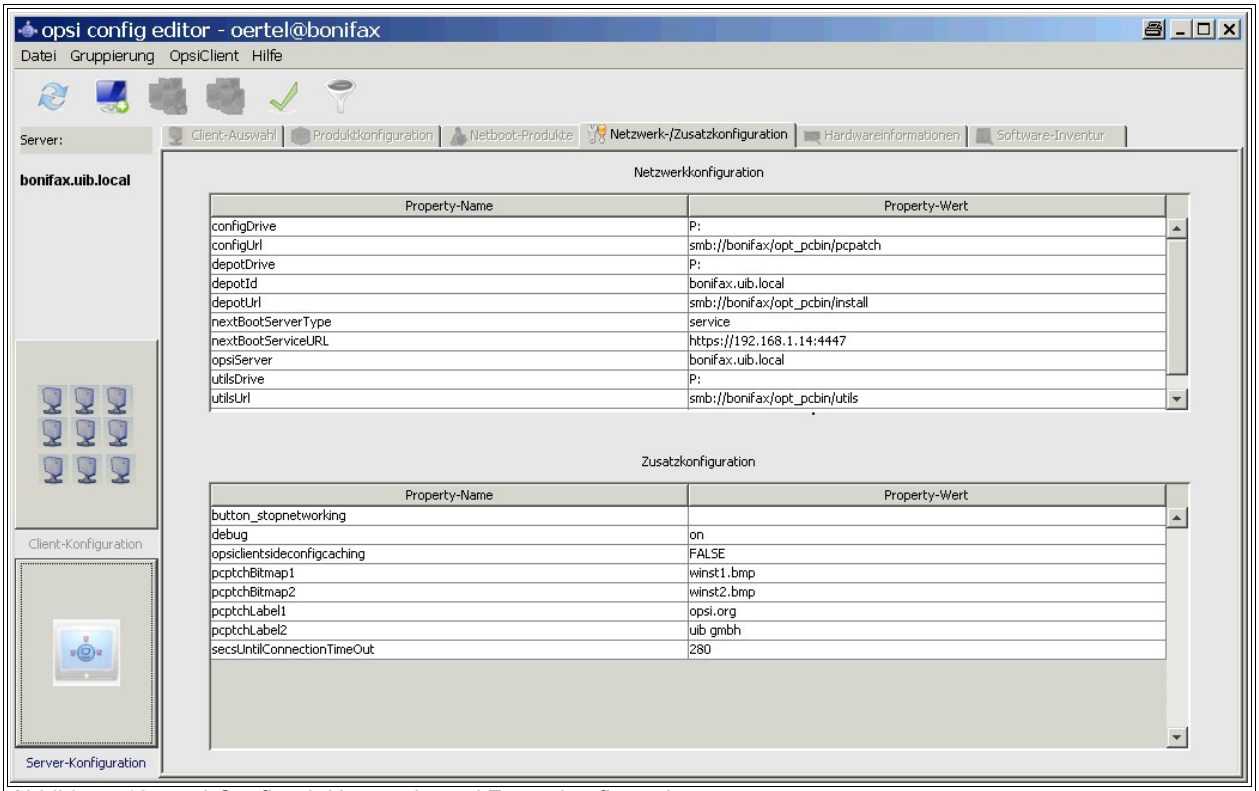


Abbildung 12: opsi-Configed: Netzwerk- und Zusatzkonfiguration

3.3. Werkzeug: opsi V3 opsi-Webconfiged

Der oben beschriebene opsi-configed steht auch als Applet zur Verfügung, wenn das Debian-Paket opsi-configed auf dem Server installiert ist.

Aufruf: http(s)://<servername>:<port>/configed/
Beispiel: https://dpvm03:4447/configed/

3.4. Werkzeug opsi-package-manager: opsi-Pakete (de-) installieren

Der opsi-package-manager dient zur (De-) Installation von opsi-Paketen auf einem opsi-Server. Die Befehle opsiinst und opsiuninst werden durch den opsi-package-manager ersetzt und sollten möglichst nicht mehr verwendet werden.

Beim Aufruf von opsi-package-manager zur Installation muss das zu installierende Paket für den Systemuser opsiconfd lesbar sein. Es wird daher dringend empfohlen,

opsi-Pakete von /home/opsiproducts bzw. einem Unterverzeichnis hiervon zu installieren.

Paket installieren (ohne Fragen neu installieren):

```
opsi-package-manager -i softprod_1.0-5.opsi
```

Paket installieren (mit Fragen nach Properties):

```
opsi-package-manager -p ask -i softprod_1.0-5.opsi
```

Paket installieren (und für alle auf Setup stellen, bei denen es installiert ist):

```
opsi-package-manager -S -i softprod_1.0-5.opsi
```

Paket deinstallieren:

```
opsi-package-manager -r softprod
```

Paket extrahieren und umbenennen:

```
opsi-package-manager -x opsi-template_<version>.opsi --new-productid myprod
```

Eine Übersicht über alle Optionen liefert die Option -h.
Die Optionen -d bzw. --depots sind für den multi-depotserver Betrieb und werden nur mit einem entsprechenden Supportvertrag unterstützt. Bei der Verwendung von -d wird das zu installierende Paket zunächst nach /var/lib/opsi/ kopiert. Dort muss ausreichend Platz zur Verfügung stehen. Siehe:
7 opsi-server mit mehreren Depots Seite 102.

```
svmopside:~# opsi-package-manager -h

Usage: opsi-package-manager [options] <command>

Manage opsi packages

Commands:
  -i, --install     <opsi-package> ...  install opsi packages
  -u, --upload      <opsi-package> ...  upload opsi packages to repositories
  -l, --list        <regex>             list opsi packages matching regex
  -D, --differences <regex>             show depot differences of opsi
                                        packages matching regex
  -r, --remove      <opsi-product-id>   uninstall opsi packages
  -x, --extract     <opsi-package> ...extract opsi packages to local directory
  -V, --version                         show program's version info and exit
```

```
  -h, --help                              show this help message and exit

Options:
  -d, --depots      <depots>       comma separated list of depots to process
                                   (default: <this-host>.<myDomain>)
                                   use keyword ALL to process all known depots
  --direct-install          install package directly without repository upload
  -p, --properties  <mode>         mode for default product property values
                      ask                        display dialog
                      package                    use defaults from package
                      keep                       keep depot defaults (default)
  -f, --force               force install/uninstall (use with extreme caution)
  -U, --update                       set action "update" on hosts where
                                        installation status is "installed"
  -S, --setup                        set action "setup" on hosts where
                                        installation status is "installed"
  --max-transfers   <num>            maximum number of simultaneous uploads
                                        0=unlimited (default)
  -o, --overwrite           overwrite existing package even if size matches
  -k, --keep-files                   do not delete client data dir on uninstall
  -t, --temp-dir    <path>         tempory directory for package install
  --new-product-id  <product-id>
                           set an new product id when extracting opsi package
  --interface       <type>                   type of user interface
                      text                       text based interface
                      snack                      newt interface (default)
  -v, --verbose             increase verbosity (can be used multiple times)
  -q, --quiet                                  do not display any messages
  --log-file        <log-file>                 path to debug log file
```

3.5. Werkzeug: opsi V3 opsi-admin

3.5.1. Übersicht

Mit opsi 3 wurde eine opsi eigene Python-Bibliothek eingeführt. Diese bietet eine API zur Konfiguration von opsi. Während der opsiconfd diese API als Webservice zur Verfügung gestellt, dient opsi-admin als Kommandzeilen Interface zu dieser API.

Dabei bietet opsi-admin einen Interaktiven Modus und einen nicht-interaktiven z.B. zum Einsatz in Skripten.

Der Aufruf von `opsi-admin -h` zeigt eine kleine Hilfe zu den Optionen:

```
# opsi-admin -h

Usage: opsi-admin [-u -p -a -d -l -f -i -c -s] [command] [args...]
```

```
  -h, --help            Display this text
  -u, --username        Username (default: current user)
  -p, --password        Password (default: prompt for password)
  -a, --address         URL of opsiconfd (default: https://localhost:4447/rpc)
  -d, --direct          Do not use opsiconfd
  -l, --loglevel        Set log level (default: 2)
                        0=nothing, 1=critical, 2=error, 3=warning, 4=notice,
5=info, 6=debug
  -f, --log-file        Path to log file
  -i, --interactive     Start in interactive mode
  -c, --colorize        Colorize output
  -S, --simple-output   Simple output (only for scalars, lists)
  -s, --shell-output    Shell output
```

opsi-admin kann auf einen opsi Webservice zugreifen oder direkt auf der Datenhaltung arbeiten. Für die Arbeit über den Webservice müssen neben der URL auch username und password angegeben werden. Dies wird man in Skripten üblicherweise nicht tun wollen. Stattdessen bietet sich hier der direkte Datenzugriff über Aufruf **`opsi-admin -d`** an.

Im interaktiven Modus (Start mit **`opsi-admin -i`** bzw. **`opsi-admin -d -i -c`**) erhalten Sie Eingabe-Unterstützung durch die TAB-Taste. Nach Eingabe der Tabtaste erhalten Sie eine Auswahl bzw. die Angabe des Datentyps der nächsten erwarteten Eingabe.

Die Optionen -s und -S erzeugen eine Form der Ausgabe welche sich leichter in Skripten weiterverarbeiten lässt.

Es gibt die Methodenaufrufe welche direkt auf die API-Aufrufe aussetzen. Darüberhinaus gibt es 'Tasks' welche eine Kombination von Methodenaufrufen zur Erledigung einer bestimmten Aufgabe darstellen.

3.5.2. Typische Verwendung

3.5.2.1. Löschen eines Produktes:

Die Methode ist deleteProduct productId. Der Aufruf in einen Skript zum Löschen z. B. des Produktes softprod ist dann:

```
opsi-admin -d method deleteProduct "softprod"
```

3.5.2.2. Ein Produkt für alle Clients auf setup stellen, welche dieses Produkt installiert haben:

```
opsi-admin -d task setupWhereInstalled "softprod"
```

3.5.2.3. Client löschen

```
opsi-admin -d method deleteClient <clientname>
z.B.:
opsi-admin -d method deleteClient pxevm.uib.local
```

3.5.2.4. Client anlegen

```
opsi-admin -d method createClient <clientname> <domain>
z.B.:
opsi-admin -d method createClient pxevm uib.local
```

3.5.2.5. Client Bootimage aktivieren

```
opsi-admin -d method setProductActionRequest <productId> <clientId>
<actionRequest>
```

z.B.:

```
opsi-admin -d method setProductActionRequest win2k pxevm setup
```

3.5.2.6. Beschreibungen den Clients zuordnen:

```
opsi-admin -d method setHostDescription "dpvm02.uib.local" , "Client
unter VMware"
```

3.5.2.7. Pcpatch Passwort setzen

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

Setzt das Passwort von pcpatch für Unix, samba und opsi.

3.5.3. Liste der Methoden

Hier eine Liste der Methoden mit einer kurzen Beschreibung. Diese dient zur Orientierung und nicht als Referenz. Das bedeutet die Beschreibung muss nicht alle Informationen enthalten die Sie benötigen um diese Methode Tatsächlich zu verwenden.

```
method addHardwareInformation hostId, info
```

Fügt Hardwareinformationen zum Rechner 'hostid' hinzu. Übergeben wird der Hash 'info'. Vorhandene Informationen werden überschrieben wenn die Keys über einstimmen. Es sind nur bestimmte Keys zulässig

```
method authenticated
```

Überprüfen ob die Authentifizierung am Service erfolgreich war.

```
method checkForErrors
```

Überprüft auf Inkonsistenzen im Backend (bisher nur implementiert für Backend File)

```
method createClient clientName, domain, description=None, notes=None
```

Erzeugt einen neuen Client.

```
method createGroup groupId, members = [], description = ""
```

Erzeugt eine Gruppe von Clients wie sie vom opsi-Configedit verwendet wird.

```
method createLicenseKey productId, licenseKey
```

Weist dem Produkt 'produktid' einen (weiteren) Lizenzkey zu.

```
method createLocalBootProduct productId, name, productVersion, packageVersion,
       licenseRequired=0, setupScript="", uninstallScript="", updateScript="",
       alwaysScript="", onceScript="", priority=10, description="", advice="",
       productClassNames=('localBoot')
```

Legt ein neues 'localboot' Produkt (Winst-Produkt) an.

```
method createNetBootProduct productId, name, productVersion, packageVersion,
       licenseRequired=0, setupScript="", uninstallScript="", updateScript="",
       alwaysScript="", onceScript="", priority=10, description="", advice="",
       productClassNames=('netboot')
```

Legt ein neues bootimage Produkt an

```
method createOpsiBase
```

Nur für interne Verwendung beim LDAP-Backend

```
method createProduct productType, productId, name, productVersion,
       packageVersion, licenseRequired=0,setupScript="", uninstallScript="",
       updateScript="", alwaysScript="", onceScript="", priority=10,
       description="", advice="", productClassNames=""
```

Legt ein neues Produkt an

```
method createProductDependency productId, action, requiredProductId="",
       requiredProductClassId="", requiredAction="",
       requiredInstallationStatus="", requirementType=""
```

Erstellt Produktabhängigkeiten

```
method createProductPropertyDefinition productId, name, description=None,
       defaultValue=None, possibleValues=[]
```

Erstellt eine Produkteigenschaft

```
method createServer serverName, domain, description=None
```

Erstellt im LDAP-Backend einen neuen Server

```
method createServerProduct  productId, name, productVersion, packageVersion,
       licenseRequired=0,setupScript="", uninstallScript="", updateScript="",
       alwaysScript="", onceScript="", priority=10, description="", advice="",
       productClassNames=('server')
```

Noch nicht implementiert, für zukünftige Verwendung

```
method deleteClient clientId
```

Löscht einen Client

```
method deleteGeneralConfig objectId
```

Löscht Konfiguration eines Clients oder einer Domain

```
method deleteGroup groupId
```

Löscht eine Gruppe von Clients

```
method deleteHardwareInformation hostId
```

Löscht sämtliche Hardwareinfos zum Rechner 'hostid'

```
method deleteLicenseKey productId, licenseKey
```

Löscht einen Lizenzkey

```
method deleteNetworkConfig objectId
```

Löscht Netzwerkkonfiguration (z.B. depotshare Eintrag) für Client oder Domain

```
method deleteOpsiHostKey hostId
```

Löscht einen pckey aus der pckey-Datenbank

```
method deleteProduct productId
```

Löscht ein Produkt aus der Datenbasis

```
method deleteProductDependency productId, action, requiredProductId="",
       requiredProductClassId="", requirementType=""
```

Löscht Produktabhängigkeit

```
method deleteProductProperties productId *objectId
```

Löscht alle Properties eines Produkts.

```
method deleteProductProperty productId property *objectId
```

Löscht ein Property eines Produkts.

```
method deleteProductPropertyDefinition productId, name
method deleteProductPropertyDefinitions productId
```

Löscht alle Produkteigenschaften zum Produkt 'productid'.

```
method deleteServer serverId
```

Löscht die Serverkonfiguration

```
method exit
```

Verläßt den opsi-admin

```
method getBackendInfos_listOfHashes
```

Liefert eine Beschreibung der auf dem opsi-server konfigurierten Backends und welche davon aktiviert sind.

```
method getBootimages_list
```

Liefert die Liste der zur Auswahl stehenden bootimages

```
method getClientIds_list serverId = None, groupId = None, productId = None,
       installationStatus = None, actionRequest = None
```

Liefert die Liste der Clients welche den angegebenen Kriterien entsprechen.

```
method getClients_listOfHashes serverId = None, groupId = None, productId =
       None, installationStatus = None, actionRequest = No
```

Liefert die Liste der Clients welche den angegebenen Kriterien entsprechen zusammen mit Beschreibung, Notizen und 'Lastseen'.

```
method getDefaultNetBootProductId clientId
```

Liefert das Netboot Produkt (z.B. Betriebssystem) welches beim Aufruf des bootimages 'install' installiert wird.

```
method getDomain hostId
```

Liefer die Domain zu einem Rechner

```
method getGeneralConfig_hash objectId
```

Liefert Allgemeine Konfiguration zu einem Client oder einer Domain

```
method getGroupIds_list
```

Liefert die Liste der gespeicherten Clientgruppen

```
method getHardwareInformation_listOfHashes hostId
```

Liefert die Hardwareinformationen zu dem angegebenen Rechner

```
method getHostId hostname
```

Liefert hostid zu dem angegebenen Hostnamen

```
method getHost_hash hostId
```

Liste der Eigenschaften des angegebenen Rechners.

```
method getHostname hostId
```

Liefert hostname zur hostid

```
method getInstallableLocalBootProductIds_list clientId
```

Liefert alle localboot Produkte die auf diesem Client installiert werden können.

```
method getInstallableNetBootProductIds_list clientId
```

Liefert alle netboot Produkte die auf diesem Client installiert werden können.

```
method getInstallableProductIds_list clientId
```

Liefert alle Produkte die auf diesem Client installiert werden können.

```
method getInstalledLocalBootProductIds_list hostId
```

Liefert alle localboot Produkte die auf diesem Client installiert sind.

```
method getInstalledNetBootProductIds_list hostId
```

Liefert die Liste der installierten netboot Produkte für einen Client oder Server

```
method getInstalledProductIds_list hostId
```

Liefert die Liste der installierten Produkte für einen Client oder Server

```
method getIpAddress hostId
```

Liefert IP-Adresse zur hostId

```
method getLicenseKey productId, clientId
```

(Noch nicht in Verwendung) Liefert einen freien Lizenzkey zu dem angegebenen Produkt bzw. liefert den der clientId zugeordneten Lizenzkey

```
method getLicenseKeys_listOfHashes productId
```

(Noch nicht in Verwendung) Liefert eine Liste der Lizenzkeys für das angegebene Produkt

```
method getLocalBootProductIds_list
```

Liefert alle (z.B. im LDAP-Baum) bekannten localboot Produkte

```
method getLocalBootProductStates_hash clientIds = []
```

Liefert für die angegebenen Clients Installationsstatus und Actionrequest für alle localboot Produkte

```
method getMacAddresses_list hostId
```

Liefert die MAC-Adresse zum angegebenen Rechner

```
method getNetBootProductIds_list
```

Liefert Liste der NetBoot Produkte.

```
method getNetBootProductStates_hash clientIds = []
```

(Noch nicht in Verwendung) Liefert für die angegebenen Clients Installationsstatus und Actionrequest für alle netboot Produkte

```
method getNetworkConfig_hash objectId
```

Liefert die Netzwerk spezifischen Konfigurationen für einen Client oder eine Domain.

```
method getOpsiHostKey hostId
```

Liefert den pckey zur angegeben hostid

```
method getPcpatchPassword hostId
```

Liefert das mit dem pckey von hostid verschlüsselte Passwort von pcpatch

```
method getPossibleMethods_listOfHashes
```

Liefert die Liste der aufrufbaren Methoden (in etwa so wie in diesem Kapitel beschrieben)

```
method getPossibleProductActionRequests_list
```

Liefert die Liste der in opsi prinzipiell zulässigen Action-Requests.

```
method getPossibleProductActions_hash
```

Liefert zu allen Produkten die möglichen Aktionen (setup, deinstall,....)

```
method getPossibleProductActions_list productId=None
```

Liefert zum angegebenen Produkt die möglichen Aktionen (setup, deinstall,....)

```
method getPossibleProductInstallationStatus_list
```

Liefert die möglichen Installationsstati (installed, not installed,...)

```
method getPossibleRequirementTypes_list
```

Liefert die möglichen Typen von Produktabhängikeiten (before, after,...)

```
method getProductActionRequests_listOfHashes clientId
```

Liefert die anstehenden ausführbaren Aktionen für den angegebenen Client

```
method getProductDependencies_listOfHashes productId = None
```

Liefert die bekannten Produktabhängigkeiten (zum angegebenen Produkt)

```
method getProductIds_list productType = None, hostId = None,
       installationStatus = None
```

Liefert die Liste der Produkte die den angegebenen Kriterien entsprechen.

```
method getProductInstallationStatus_hash productId, hostId
```

Liefert den Installationsstatus zum angegebenen Client und Produkt

```
method getProductInstallationStatus_listOfHashes hostId
```

Liefert den Installationsstatus zum angegebenen Client

```
method getProductProperties_hash productId, objectId = None
```

Liefert die Schalterstellungen (product properies) zum angegebenen Produkt und Client

```
method getProductPropertyDefinitions_hash
```

Liefert alle bekannten product properties mit Beschreibung, erlaubten Werten,....

```
method getProductPropertyDefinitions_listOfHashes productId
```

Liefert die product properties zum angegebenen Produkt mit Beschreibung, erlaubten Werten,....

```
method getProductStates_hash clientIds = []
```

Liefert Installationsstati und Actionrequests der einzelnen Produkte (zu den agegebenen Clients)

```
method getProduct_hash productId
```

Liefert die Metadaten (Beschreibung, Version,...) zum angegebenen Produkt

```
method getProvidedLocalBootProductIds_list serverId
```

Liefert die Liste der auf dem angegebenen Server Bereitgestellten localboot Produkte.

```
method getProvidedNetBootProductIds_list serverId
```

Liefert die Liste der auf dem angegebenen Server Bereitgestellten netboot Produkte.

```
method getServerId clientId
```

Liefert den zuständigen opsi-server zum angegebenen Client

```
method getServerIds_list
```

Liefert die Liste der bekannte opsi-server

```
method getServerProductIds_list
```

Liste der Server-Produkte

```
method getUninstalledProductIds_list hostId
```

Liefert die deinstallierten Produkte

```
method powerOnHost mac
```

Sendet ein WakeOnLan Signal an die angegebene MAC

```
method setBootimage bootimage, hostId, mac=None
```

Setzt einen bootimage start für den angegebenen Client und bootimage

```
method setGeneralConfig config, objectId = None
```

Setzt für Client oder Domain die GenaralConfig (z.B. Sektion [general] in *.sysconf Dateien)

```
method setHostDescription hostId, description
```

Setzt für einen Client die Beschreibung

```
method setHostLastSeen hostId, timestamp
```

Setzt für einen Client den Zeitstempel für LastSeen

```
method setHostNotes hostId, notes
```

Setzt für einen Client die Notiz-Angaben

```
method setMacAddresses hostId, macs
```

Trägt für einen Client seine MAC-Adresse in die Datenbank ein.

```
method setNetworkConfig objectId, serverId='', configDrive='', configUrl='',
       depotDrive='', depotUrl='', utilsDrive='', utilsUrl='', winDomain='',
       nextBootServiceURL=''
```

Setzt für einen Client die angegebene Netzwerkdaten für den opsi-preloginloader

```
method setOpsiHostKey hostId, opsiHostKey
```

Setzt für einen Rechner den pckey

```
method setPXEBootConfiguration hostId *args
```

Schreibt Pipe für PXE-Boot mit *args in der 'append'-Liste

```
method setPcpatchPassword hostId password
```

Setzt das verschlüsselte (!) password für hostid

```
method setProductActionRequest productId, clientId, actionRequest
```

Setzt für den angegebenen Client und Produkt einen ActionRequest

```
method setProductInstallationStatus productId, hostId, installationStatus,
       policyId="", licenseKey=""
```

Setzt für den angegebenen Client und Produkt einen Installationsstatus (policyId und Licensekey noch nicht in Verwendung)

```
method setProductProperties productId, properties, objectId = None
```

Setzt product property für das angegebene Produkt (und den angegebenen Client)

```
method unsetBootimage hostId
```

Setzt einen bootimage start für den angegebenen Client zurück

```
method unsetPXEBootConfiguration hostId
```

Löscht PXE-Boot Pipe.

```
method unsetProductActionRequest productId, clientId
```

Setzt Actionrequest auf undefined, so das im LDAP übergreifende Policies für diesen Client wirken können.

4. Localboot Produkte: Automatische Softwareverteilung mit opsi

4.1. Der opsi-preloginloader

4.1.1. Überblick

Damit die Verteilung von Software nicht zur 'Turnschuh-Administration' wird, muss ein Client-PC selbstständig erkennen, dass neue Softwarepakete oder Updates für ihn bereit stehen und diese installieren. Bei der Installation ist auf jede Form von Anwender-Interaktion zu verzichten, damit diese unbeaufsichtigt erfolgen kann und nicht durch verunsicherte Anwender notwendige Installationen abgebrochen werden.

Diese Anforderungen werden bei opsi durch zwei Softwarekomponenten realisiert:

Auf dem Client wird ein opsi-PreLoginLoader installiert. Dieser überprüft nach jedem Boot und vor dem Login des Anwenders, anhand von Konfigurationsdateien auf einem zentralen Fileshare, ob für diesen Client ein Update installiert werden soll.

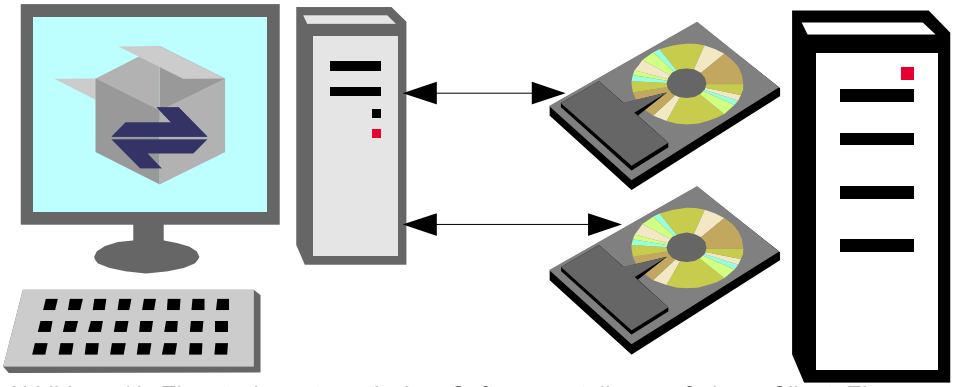


Abbildung 13: Einsatz der automatischen Softwareverteilung auf einem Client. Ein Fileserver stellt Shares für Konfigurationsdateien und Softwarepakete bereit.

Soll Software installiert werden, wird das skriptgesteuerte Installationsprogramm Winst gestartet. Auf einem Fileshare stehen die dafür notwendigen Skripte und Softwarepakete bereit. Während dieser Zeit hat der Anwender keine Notwendigkeit und keine Möglichkeit in den Installationsprozess einzugreifen.

Um zu verhindern, dass sich ein Anwender vor dem Abschluss der Installation einloggen und so den Installationsprozess stören kann, kann zusätzlich ein 'Loginblocker' installiert werden, der eine Anmeldung erst nach beendeter Installation zulässt.

Damit Softwarepakete mit dem Programm Winst ohne Interaktion installiert werden können, müssen sie dafür vorbereitet werden. Das Programm Winst bietet hierfür unterschiedliche Möglichkeiten:

- Vorhandene Setup-Programme können über Parameter im sogenannten 'silent' oder 'unattended' Modus gestartet werden.
- Das Standard-Setup kann 'aufgezeichnet' und die entsprechenden Tätigkeiten direkt durch das opsi Installationsprogramm Winst durchgeführt werden.
- Die Anworten zum Orginal Setupprogramm können mit Hilfe des freien Werkzeugs autoit (http://www.hiddensoft.com/AutoIt/) automatisiert gegeben werden.

In der Praxis wird eine Kombination der verschiedenen Varianten Ihren Bedürfnissen am ehesten entgegen kommen.

4.1.2. Einbindung der Softwareinstallation über den opsi-PreLoginLoader

Ziel der Einbindung der Softwareverteilung ist es, Softwareinstallationen ohne die Notwendigkeit von Anwenderinteraktionen durchzuführen. Diese Installationen müssen unabhängig von den Rechten des Anwenders auf diesem PC mit Administratorrechten ausgeführt werden. Softwareinstallation und Anwendertätigkeit sollen hierbei klar getrennt sein. Das heißt, weder soll der Anwender während einer Softwareinstallation am Rechner etwas manipulieren können, noch soll der Anwender durch eine Softwareinstallation gestört werden, wenn er bereits eingeloggt ist.

Der opsi-PreLoginLoader besteht aus vier Komponenten: prelogin.exe, pcptch.exe, winst32.exe und dem optionalen Loginblocker. Die prelogin.exe startet beim Boot des PC's als Service. Das bedeutet, die prelogin.exe wird noch vor dem Login eines Anwenders aktiv.

Die Aufgabe der prelogin.exe ist es, nach dem Start des Betriebssystems die automatische Softwareinstallation mit den notwendigen Zugriffsrechten auf Netz und auf die grafische Oberfläche zu starten. Als erste Stufe wird dazu das Programm pcptch.exe gestartet. Es verwendet die Rechte des bei der Installation des opsi-PreLoginLoaders angelegten und mit Administratorrechten ausgestatten Pseudo-Anwenders pcpatch. Die pcptch.exe stellt anhand von Konfigurationsinformationen eine Netzwerkverbindung zu den Fileshares her, die die Softwarepakete und die PC-Konfigurationsdateien beinhalten. Die hierzu nötigen Informationen sind entweder nur in der Registry oder - bei Verwendung des opsi-deposervers - zum Teil in dessen Konfigurationsdateien gespeichert. Eine Beschreibung dieser Konfigurationsinformationen finden Sie in den entsprechenden Kapiteln am Ende dieses Handbuchs.

Nun startet die pcpatch.exe das opsi-Installationsprogramm Winst, das anhand der zu dem PC gehörenden Konfigurationsdatei (<pcname>.ini) erkennt, ob Installationen durchgeführt werden müssen und führt diese aus. Nach Beendigung der Installationen werden die Informationen über die durchgeführten Installationen in die Konfigurationsdateien (<pcname>.ini) eingetragen. Danach beenden sich Winst und pcptch.exe und der Anwender kann sich einloggen.

Eine Installation kann einen Reboot beinhalten. In diesem Fall ist die Installation noch nicht abgeschlossen, sondern wird nur für einen System-Neustart unterbrochen. Der Reboot erfolgt vollkommen automatisch und ohne dass der Anwender sich vorher anmelden kann. Entsprechend setzt die Softwareinstallation nach dem Reboot wieder vollautomatisch an der richtigen Stelle auf.

4.1.3. Nachträgliche Installation des opsi-PreLoginLoaders

Möchte man nachträglich, d.h. wenn das Betriebssystem bereits anderweitig installiert wurde, einen PC in die Softwareverteilung aufnehmen, so muss der opsi-PreLoginLoader installiert werden.

Dabei muss darauf geachtet werden, dass auf dem PC und dem Server der pckey zur Verschlüsselung des Passwortes korrekt installiert wird.

Hierzu empfehlen sich die nachfolgenden Methoden.

4.1.3.1. Verwendung von opsi-deploy-preloginloader

Das opsi-deploy-preloginloader Skript verteilt den opsi-preloginloader direkt vom opsi-server auf die Clients. Voraussetzung hierfür sind bei den Clients:

- ein offener c$ share
- ein offener admin$ share
- ein administrativer account

Das Skript erzeugt serverseitig den Client, kopiert die Installations-Dateien und Konfigurationsinformationen wie den pckey auf den Client und startet dort die Installation.

Mit dem opsi-deploy-preloginloader Skript kann auch eine ganze List von Clients bearbeitet werden. Das Skript findet sich unter /opt/pcbin/install/preloginloader.

```
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 accessable on every client.
Simple File Sharing (Folder Options) should be disabled on the Windows
machine.
Options:
    -h        show this help text
    -V        show version information
    -v        increase verbosity (can be used multiple times)
    -u        username for authentication (default: Administrator)
    -p        password for authentication
    -c        use fqdn instead of hostname for smb/cifs connection
    -r        reboot computer after installation
    -s        shutdown computer after installation
    -f        file containing list of clients (one hostname per line)
```

4.1.3.2. Verwendung von service_setup.cmd

Unter /opt/pcbin/install/preloginloader (bzw. auf dem share opt_pcbin) findet sich auch das Skript service_setup.cmd. Dieses kann mit administrativen Rechten vom Client aus gestartet werden. Das Skript nimmt per opsi-Webservice Kontakt zum Server auf um serverseitig den Client zu erzeugen und den pckey zu erfahren. Dies erfolgt zunächst mit der in der config.ini eingetragenen user/password Kombination. Schlägt dies fehl, so

erscheint eine Art Login-Fenster mit Service-URL user und password. Dort kann die Operation mit dem Accountdaten eines Mitglieds der Gruppe opsiadmin authorisiert werden.

Achtung: Der Client rebootet nach der Installation.

4.1.4. Sperrung des Anwender Logins mit dem opsi-Loginblocker

Um zu verhindern, dass sich ein Anwender schon vor dem Abschluss der Installation einloggt, kann zusätzlich ein Loginblocker installiert werden. Dieser gibt den Zugriff auf das Login erst frei, wenn der Installationsprozess beendet ist.

Der Loginblocker ist als gina.dll realisiert. Gina steht dabei für „Graphical Identification and Authentication“, und stellt die seitens Microsoft offiziell unterstützte Möglichkeit dar, in den Loginprozess von Windows einzugreifen. Die pgina.dll, die vom opsi-PreLoginLoader verwendet werden kann, basiert auf der pgina des Projektes http://pgina.xpasystems.com. Gelegentlich ist es der Fall, dass bereits andere Softwareprodukte (z.B. Client für Novellnetzwerke) eine gina.dll auf dem System hinterlassen haben und empfindlich auf Eingriffe reagieren. Generell sind mehrere nacheinander aufgerufene gina.dll (gina chaining) durchaus möglich. Auch die pgina.dll des Loginblockers ist für das genannte chaining vorbereitet. Sollte der beschriebene Fall bei Ihren Clients eintreten, informieren Sie sich bitte auf der o.g Webseite nach den bestehenden Anpassungsmöglichkeiten, oder kontaktieren Sie die Firma uib.

Ob der Loginblocker installiert wird, ist festgelegt in der <pcname>.ini in der Sektion [preloginloader-install] mit dem Schalter LoginBlockerStart=on/off.

4.2. opsi Standardprodukte

4.2.1. opsi-preloginloader 3.3.1

Das Produkt preloginloader dient zur Installation und Aktualisierung des opsi-preloginloaders auf den Clients.

4.2.2. opsi-preloginloader 3.4

Der opsi-preloginloader 3.4 ist eine Neuentwicklung, welche im Moment nur unseren Vista-Kunden zur Verfügung steht. Eine detaillierte Beschreibung finden Sie im opsi-Vista-Installationshandbuch.

4.2.3. opsi-winst

Das Produkt opsi-winst ist ein Spezialfall. Es enthält den aktuellen opsi-winst. Dieser muss zur Aktualisierung nicht auf setup gestellt werden. Vielmehr prüft ein Teil der preloginloaders bei jedem Start, ob auf dem Server eine andere Version des Winst verfügbar ist und holt sich diese im Zweifelsfall.

4.2.4. javavm: Java Runtime Environment

Das Produkt javavm stellt die für den opsi-configed benötigte Java 1.6 Laufzeitumgebung für die Clients zur Verfügung.

4.2.5. opsi-adminutils

Das Produkt opsi-adminutils bietet neben einigen Hilfsprogrammen vor allem eine lokale Installation des opsi-configed.

4.2.6. swaudit + hwaudit: Produkte zur Hard- und Software-Inventarisierung

Die Produkte hwaudit und swaudit dienen der Hard- bzw. Software-Inventarisierung.

Bei der Hardware-Inventarisierung werden die Daten über WMI erhoben und über den opsi-Webservice an den Server zurück gemeldet.

Bei der Software-Inventarisierung werden die Daten aus der Registry (HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall) erhoben und über den opsi-Webservice an den Server zurück gemeldet.

Beide Produkte verwenden die Sprache Python und setzten daher die Installation des Produktes python voraus.

4.2.7. opsi-template

Template zur Erstellung eigener opsi-Scripts.

Sie können das Template extrahieren mit

```
opsi-package-manager -x opsi-template_<version>.opsi
```

oder auch dabei gleich umbenennen mit

```
opsi-package-manager -x opsi-template_<version>.opsi --new-productid myprod
```

4.2.8. python

Python Laufzeitumgebung z.B. für die Produkte swaudit und hwaudit.

4.2.9. xpconfig

Paket zum Customizing der Grundeinstellungen von Oberfläche, Explorer usw., nicht nur für XP.

4.3. Einbindung eigener Software in die Softwareverteilung von opsi

4.3.1. Erstellung eines opsi-Winst Skriptes

4.3.1.1. Überblick

Prinzipiell gibt es drei Verfahren der Einbindung eines Softwarepaketes in die automatische Softwareverteilung für Windows-Betriebssysteme, zuzüglich einer Variante, die sich auf die Pakete für den Microsoft Installer Service bezieht.

1. Unattended / Silent Setup:
   Das Original-Setupprogramm wird verwendet und über Kommandozeilenargumente in einen nicht-interaktiven Modus versetzt.

   Der wichtigste Spezialfall davon ist der

2. „stille“ Aufruf eines MSI-Pakets:

   Ein Paket für den Microsoft Installer Service ist vorhanden und wird mit einer „quiet“-Option aufgerufen.

3. Interaktives Setup mit automatisierten Antworten:
   Zur Vorbereitung wird bei einem Lauf des Original-Setupprogramms festgestellt, welche Fenstertitel das Programm zeigt und welche Fragen und Antworten beim Setup anfallen. Dies wird in einem Skript niedergeschrieben. Im Prozess der Softwareverteilung läuft das Setupprogramm dann unter Kontrolle eines Automatisierungs-Programms, welches das Setupprogramm gemäß dem Skript steuert.

4. Analysieren und Neu-Paketieren:
   Es wird (teil-automatisiert) untersucht, welche Komponenten auf einem Testpc, auf dem nur das Betriebssystem bzw . allgemeine Basissoftware verfügbar ist, installiert werden müssen, damit die Software wie gewünscht läuft. Diese Analyse dient als Basis, um ein neues Verteilungspaket zu bauen. Das Paket kann dabei direkt mit wInst-Mitteln erstellt werden. Es kann aber auch als MSI-Paket ausgeführt werden, das dann in einen beliebigen Verteilungsmechanismus eingebunden werden kann.

Im Einzelnen:

4.3.1.2. Einbindung mit Unattended bzw. Silent Setup

Beim „unattended“ oder „silent setup“ wird das Original-Setupprogramm über Kommandozeilen Argumente in einen nicht interaktiven Modus gestellt.

Hat man die passenden Kommandozeilenparameter gefunden, so bettet man den Aufruf in ein Winstskript ein.

Hier ein Beispiel, bei dem das Setupprogramm mit dem Argument /silent aufgerufen wird:

```
; Copyright (c) uib umwelt informatik büro gmbh (www.uib.de)
; This sourcecode is owned by uib
; and published under the Terms of the General Public License.

[Initial]
Message=installiere Produkt xyz ...
StayOnTop=false

[Aktionen]
  Winbatch_produkt_silent_install

[Winbatch_produkt_silent_install]
%SCRIPTPATH%\setup_xyz.exe /silent
```

Der hier zu empfehlende Aufruf von 'stayOnTop=false' dient dazu, dem aufgerufenen Setupprogramm zu erlauben, sich vor das Winst-Fenster zu setzen. Dadurch werden die Fortschrittsanzeigen und andere Meldungen des Setupprogramms überhaupt erst sichtbar.

Das Problem dieser Installationsmethode ist, die geeigneten Kommandozeilenargumente zu finden.

Gängige Parameter sind '/s' oder '/silent' oder '/s /v"/qb-!"'

Eine allgemein gültige Lösung gibt es nicht, aber hilfreiche Tipps, die zu einer Lösung führen.

4.3.1.2.1. Suche bei unattended.sourceforge.net und anderen

Bevor man sich in Forschungen stürzt, ist dringend zu empfehlen, bei unattended.sourceforge.net zu schauen, ob jemand das Problem bereits gelöst hat:

Unter

http://unattended.sourceforge.net/installers.php

http://www.german-nlite.de/index.php?act=module&module=pages&pg=schalterse

finden sich Hinweise zu Schaltern bei gängigen Setupprogrammen und auch zu gängigen Produkten.

Selbst wenn Sie nicht fündig werden, bieten diese Seiten eine Fülle von Anregungen.

Und bei unattended.sourceforge (oder bei uib.de) können Sie Ihre Lösung auch wieder anderen zur Verfügung stellen.

Für viele Produkte gibt es schon Integrationen bei unattended Sourceforge

Beispiele finden sich Im wiki siehe http://ubertechnique.com/unattended/Scripts

oder im CVS siehe http://cvs.sourceforge.net/viewcvs.py/unattended/unattended/install/scripts/

In diesen ist beschrieben, wie man z.B. als "msi-Paket" oder "silent" eine exe installiert. Diese kann man dann modifiziert als Winbatch-Aufrufe verwenden.

Weitere lohnende Quellen sind:

http://www.appdeploy.com/packages/browse.asp?cat=all

http://www.german-nlite.de/index.php?autocom=custom&page=ug-schaltertabelle

http://www.windows-unattended.de/component/option,com_appbase/Itemid,160/

http://www.msfn.org/board/lofiversion/index.php/f80.html

4.3.1.2.2. Suche beim Hersteller des Programms

Da das Problem einer automatischen Installation inzwischen vielen Herstellern bewusst ist, finden sich häufig in der Produktdokumentation oder auf der Homepage des Herstellers Anleitungen oder Hinweise hierzu.

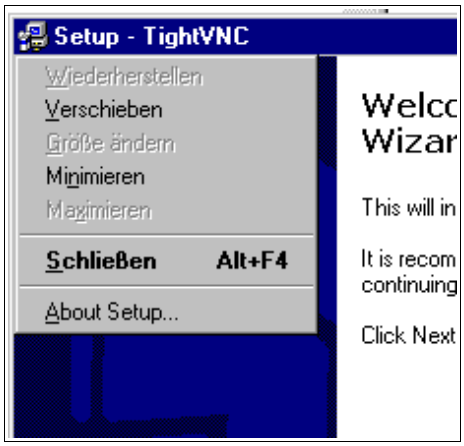
4.3.1.2.3. Suche beim Hersteller des Setup-Programms

Setupprogramme werden in der Regel von den Herstellern der diversen Softwareprodukte nicht selbst geschrieben. In den meisten Fällen bedienen sich vielmehr die Produkthersteller selbst spezieller Softwareprodukte, mit denen Setupprogramme relativ einfach erstellt werden können. Die verwendbaren Kommandozeilenargumente sind daher zumeist typisch für das verwendete Produkt zur Erstellung von Setupprogrammen.

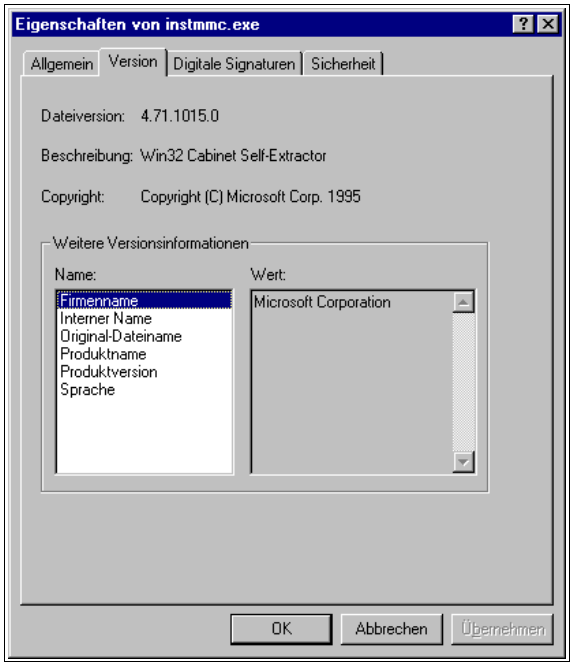
Wie lässt sich nun der Hersteller des Setupprogramm-Generators in Erfahrung bringen? Häufig steht der betreffende Name des Herstellers in der Titelleiste des

Begrüßungsfensters. Im folgenden Beispiel hat man es mit einem Programm zu tun, das von einem Produkt der Firma Installshield entwickelt wurde:

Teilweise werden die Hinweise auch dezent versteckt wie im 'About' Fenster von Innosetup.



Finden sich im Begrüßungsfenster keine Hinweise, so können vielleicht die Versionsangaben hilfreich sein. Diese erhält man, wenn man im Explorer mit der rechten Maustaste das Kontextmenü zum Setupprogramm aufruft und dort 'Eigenschaften' wählt.



Auf der Homepage des Herstellers des Setupprogramms wird man zumeist mit der Suche nach Stichworten wie 'silent' , 'silent Install' oder 'unattended' fündig.

Hier einige Links zu Internetseiten von gängigen Herstellern:

http://unattended.sourceforge.net

http://helpnet.installshield.com/robo/projects/InstallShieldXFAQ/FAQDeploymentSilent.htm

http://helpnet.installshield.com/robo/projects/InstallShieldXHelpLib/IHelpSetup_EXECmdLine.htm#sParam

http://helpnet.installshield.com/robo/projects/InstallShieldXHelpLib/SetupIss.htm

http://www.jrsoftware.org/isfaq.php#silent

http://nsis.sourceforge.net/

http://nsis.sourceforge.net/index.php?id=19&backPID=15&tx_faq_faq=39

http://www.wise.com/ (normalerweise /s)

4.3.1.2.4. Installation mit eingeloggtem user

Als Ausgangspunkt wird vorausgesetzt, das Sie ein Winst-Script mit einer unattended Installation haben, das von einem Administrativen user aus aufgerufen funktioniert aber im Rahmen der automatischen Installation beim Boot scheitert.

Eine mögliche Ursache ist dann, daß dieses Setupprogramm einen eingeloggten user bzw. den Zugriff auf ein user-Profil benötigt.

Handelt es sich um eine MSI-Installation hilft evtl. die Option ALLUSERS=2.
Beispiel:

```
[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
```

Eine weitere Möglichkeit ist, dass das Installationsprogramm nicht ordentlich terminiert, z.B. weil ein ein Subprozess gestartet wird. Dann kann man den Winbatch-Aufruf mittels /WaitSeconds [AnzahlSekunden]
oder aber mit /WaitForProcessEnding program /TimeOutSeconds seconds
probieren.

Eine andere aufwendigere Möglichkeit dieses Problem zu lösen, die Möglichkeit einen administrativen user temporär anzulegen, und diesen zur Installation des Programms zu Verwenden. Dies ist im Winst-Handbuch im Kapitel Kochbuch / Skript für Installationen im Kontext eines lokalen Administrators' dokumentiert.

4.3.1.3. Arbeiten mit MSI-Paketen

Microsoft hat mit Windows 2000 ein eigenes Installationskonzept vorgestellt, das auf dem Microsoft Installer Service, kurz „MSI“ beruht. Inzwischen sind viele Setup-Programme MSI-konform.

MSI-Konformität bedeutet, dass die eigentliche Installation darin besteht, dass an den MSI ein Paket von Installations-Anweisungen übergeben wird (im Prinzip eine Datei mit einen Namen der Form „produkt.msi“) und der MSI dieses Paket dann ausführt.

In der Praxis sieht dies meist so aus, dass die zu einem Produkt gehörige „setup.exe“ eine Datei „produkt.msi“ und ein zusätzliches Steuerprogramm für die Installation enthält. Das Steuerprogramm packt „produkt.msi“ aus und fragt, ob eine Installation

starten soll. Wird dies bestätigt, prüft das Steuerprogramm, ob der MSI schon eingerichtet ist und übergibt bei positivem Ergebnis der Prüfung diesem die „produkt.msi“. Ist der MSI nicht eingerichtet und wird insbesondere das Programm „msiexec.exe“ nicht gefunden, so startet das Steuerprogramm zuerst eine Installationsprogramm für den MSI.

Klickt man bei der Frage, ob die Installation starten soll, nicht auf „weiter“, sondern ruft den Explorer auf, so findet sich das ausgepackte MSI-Paket meist in einem temporären Verzeichnis.

Dieses Paket kann nun dazu verwendet werden, eine Installation „unattended“ - also „unbewacht“, d.h. ohne dass ein Benutzereingriff“ erforderlich ist - auszuführen. Dazu ist bei vorhandener msiexec.exe aufzurufen:

```
msiexec /qb-! ALLUSERS=2 /i Product.msi
```

bzw.

```
msiexec /i Product.msi /qn
```

Zusätzlich können häufig noch weitere, produktspezifische Argumente übergeben werden. Eine Übersicht über die Kommandozeilen Argumente der msiexec.exe gibt:

http://helpnet.installshield.com/robo/projects/InstallShieldXHelpLib/IHelpCmdLineMSI.htm

Weitere Informationen zu MSI unter:

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/de/library/ServerHelp/9361d377-9011-4e21-8011-db371fa220ba.mspx

englisch

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/msiexec.mspx

http://www.microsoft.com/windows2000/techinfo/howitworks/management/installer.asp

http://www.microsoft.com/technet/prodtechnol/windows2000serv/maintain/featusability/winmsi.mspx

4.3.1.4. Customizing nach einer silent/unattended Installation

Häufig will man nach einer erfolgreichen silent Installation noch Anpassungen an der Installation vornehmen. Hierzu bietet der opsi-Winst ein mächtiges Werkzeug. Doch bevor diese Eingesetzt werden kann muss oft ermitteltet werden, welche in der graphischen Oberfläche getätigten Änderungen zu welchen Veränderungen in Dateien und der Registry geführt haben.

Hierzu kann man die unter 'Analyse und Neu-Paketieren' vorgestellten Werkzeuge einsetzen. Häufig führen aber auch kleinere Werkzeuge schneller zum Erfolg.
Hier einige Links zu entsprechenden Werkzeugen:

http://www.sysinternals.com/

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

4.3.1.5. Einbindung mit automatisierten Reaktionen des Setup-Programms

Eine weitere schnelle Möglichkeit zur Einbindung in die automatische Softwareverteilung ist das 'Setup mit automatisierten Antworten'. Hierzu wird eine Steuerungssoftware verwendet, die z.B. auf das Erscheinen eines bestimmten Fensters warten kann und dann in dieses Fenster skriptgesteuert eine Antwort gibt. Wir empfehlen hier den Einsatz der Software autohotkey (http://de.autohotkey.com/) oder AutoIt (http://www.hiddensoft.com/autoit3/).

Folgendes (sehr einfache) Beispiel verdeutlicht das Prinzip von AutoIt:

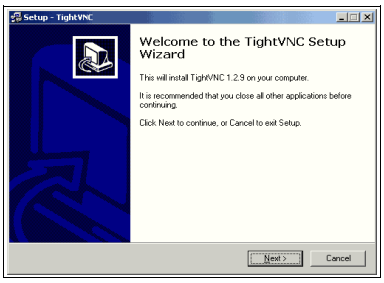
Automatisiert wird hier mit Hilfe von AutoIt das Setup von TightVNC. Das AutoIt-Skript sieht so aus:

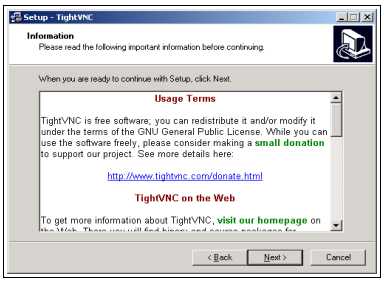
```
; starte das Setupprogramm
Run, tightvnc-1.2.9-setup.exe
; warte auf das erste Fenster mit dem Titel Setup - TightVNC
WinWait, Setup - TightVNC
; Next Button mit Enter bestätigen
Send, {ENTER}
; usw.
WinWait, Setup - TightVNC
Send, {ENTER}
WinWait, Setup - TightVNC
Send, {ENTER}
WinWait, Setup - TightVNC
Send, {ENTER}
WinWait, Setup - TightVNC
```

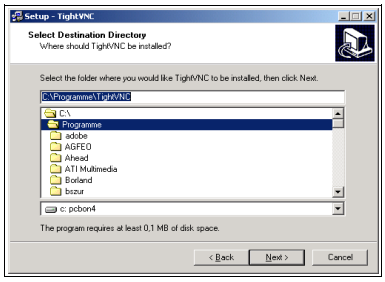
```
Send, {ENTER}
WinWait, Setup - TightVNC
Send, {ENTER}
WinWait, Setup - TightVNC
Send, {ENTER}
; hier läuft das Installieren
WinWait, Setup - TightVNC
; Finish bestätigen
Send, F
exit
```

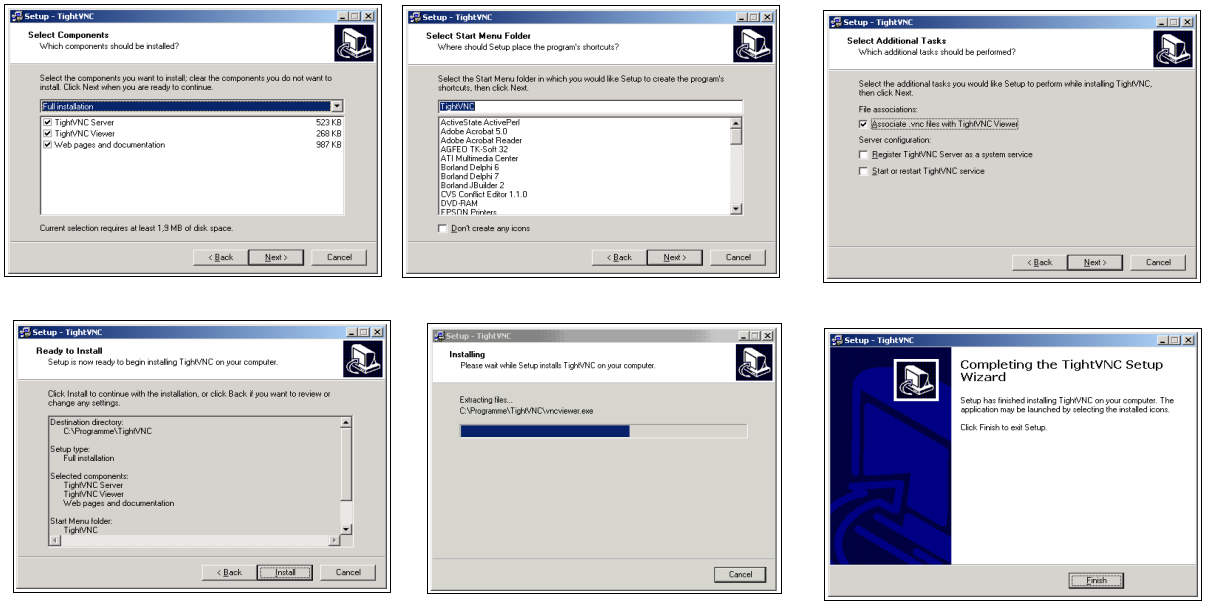
Dabei wird autoit.exe mit dem Skriptnamen als Parameter aufgerufen.

Die zu dem Skript gehörenden Screenshots des Setupprogramms sehen Sie in den folgenden Abbildungen:









AutoIt bietet eine ganze Reihe zusätzlicher Möglichkeiten, den Setupprozess zu steuern. Auch eventuelle Fehlerzustände können (so vorher bekannt) mit dem Einsatz von [ADLIB] Sektionen im Skript abgefangen werden.

Trotzdem bleibt ein prinzipielles Problem bestehen: Nicht vorhergesehene (und im Skript berücksichtigte) Fehlerfenster können das Skript zum Stoppen bringen. Außerdem kann der Anwender mit Maus und Tastatur 'dazwischen funken', wenn diese nicht gesperrt sind. Von daher ist aus unserer Sicht ein unattended oder silent setup die bessere Lösung.

Sehr gut kann auch eine Kombination aus beidem funktionieren: Das Silent-Setup übernimmt die eigentliche Installation und das AutoIt-Skript fängt bekannte Sonderbedingungen ab.

Beispiel: Winstskript zur Installation von TightVNC

```
[Initial]
Message=installiere tightvnc 1.2.9 ......

[Aktionen]
; starte AutoIt als Hintergrund-Prozess um Fenster abzufangen,
; das erscheint wenn tightvnc während der Installation als Service läuft
winbatch_tightvnc_autoit_confirm  /LetThemGo
; starte das setup Programm als silent setup
winbatch_tightvnc_silent_install

[winbatch_tightvnc_autoit_confirm]
%SCRIPTPATH%\autoit %SCRIPTPATH%\confirm.aut
```

```
[winbatch_tightvnc_silent_install]
%SCRIPTPATH%\tightvnc-1.2.9-setup.exe /silent
```

4.3.1.6. Analyse und Neu-Paketieren

Wenn der Entwickler einer Anwendung ein Paket zur Auslieferung der Anwendung schnürt, kennt er die benötigten Komponenten. Im nach hinein, wenn schon ein Paket existiert, das mittels eines Setup-Programm zu installieren ist, kann die Kenntnis, welche Komponenten installiert werden müssen, damit eine Anwendung wie gewünscht auf einem Arbeitsplatzrechner lauffähig ist, aus der Studie der Effekte bei der Ausführung des vorhandenen Setup-Programms gewonnen werden.

Zur Durchführung derartiger Analysen hatte Microsoft für Windows NT 4 und Windows 2000 das Programm **sysdiff** zur Verfügung gestellt. Dieses Programme konnte den Zustand der Softwarekomponenten eines Rechners vor und nach einer Installation jeweils in einem „Schnappschuss“ festhalten. Aus den beiden „Schnappschüssen“ konnte sysdiff automatisch deren Differenz und damit im Prinzip die Gesamtheit aller Veränderungen auf dem Arbeitsplatzrechner, die durch die Ausführung des Setup-Programms bewirkt wurden, ermitteln. Das Ergebnis der Differenzbildung konnte sysdiff in einem Skript-Format darstellen. Ein so produziertes Skript konnte schließlich als Basis für die Reproduktion der ursprünglichen Installation dienen.

Die mit sysdiff Installationsroutinen und -Pakete entsprechen allerdings nicht dem neuen MSI-Standard. Vermutlich aus diesem Grund wird sysdiff von Windows XP nicht mehr unterstützt. Als Ersatz hat Microsoft als Tool zum Erstellen von von MSI-Paketen das Produkt **WinINSTALL LE** der Firma OnDemand Software in seine Server Edition integriert.

WinINSTALL LE ist (wieder) als Freeware verfügbar.

http://www.ondemandsoftware.com

WinINSTALL LE führt weitgehend automatisiert unter Windows XP ein Verfahren ähnlich dem für sysdiff geschilderten durch. Ein „Schnappschuss“ des Systems vor und nach einer Probeinstallation mündet in einer anschließenden Differenzbildung, deren

Ergebnis nun in einem MSI-Paket festgehalten wird. Dieses Paket sollte sich zur Installation mit dem Microsoft Installer eignen.

Weiter wird direkt von Microsoft das Programm Orca zur Verfügung gestellt, mit dem MSI-Pakete einer Inspektion unterzogen werden können. Im Prinzip (wenn die Aufgabe nicht so komplex wäre) können Pakete mit Orca auch bearbeitet oder sogar erstellt werden. Orca kann auch die formale Korrektheit von MSI-Paketen, die mit anderen Mitteln erstellt wurden, automatisch prüfen. Orca ist Teil des Platform SDK für den Windows Server 2003, das bezogen werden kann unter der URL:

http://www.microsoft.com/msdownload/platformsdk/sdkupdate/

Ein interessantes freies Tool zur eigenständigen Erstellung von MSI-Paketen ist das Programm Installer2GO der Firma Dev4PC. Das Programm ist erhältlich unter:

http://dev4pc.com

Im folgenden werden Hinweise für das Arbeiten mit WinINSTALL LE sowie die Nutzung von Orca gegeben.

4.3.1.6.1. Hinweise zur Anwendung von WinINSTALL LE

WinINSTALL LE muss zunächst auf einem als **Server** dienenden Rechner installiert werden. Auf diesem soll das **gleiche Betriebssystem** wie auf dem Ziel-Client laufen, also in der Regel Windows XP.

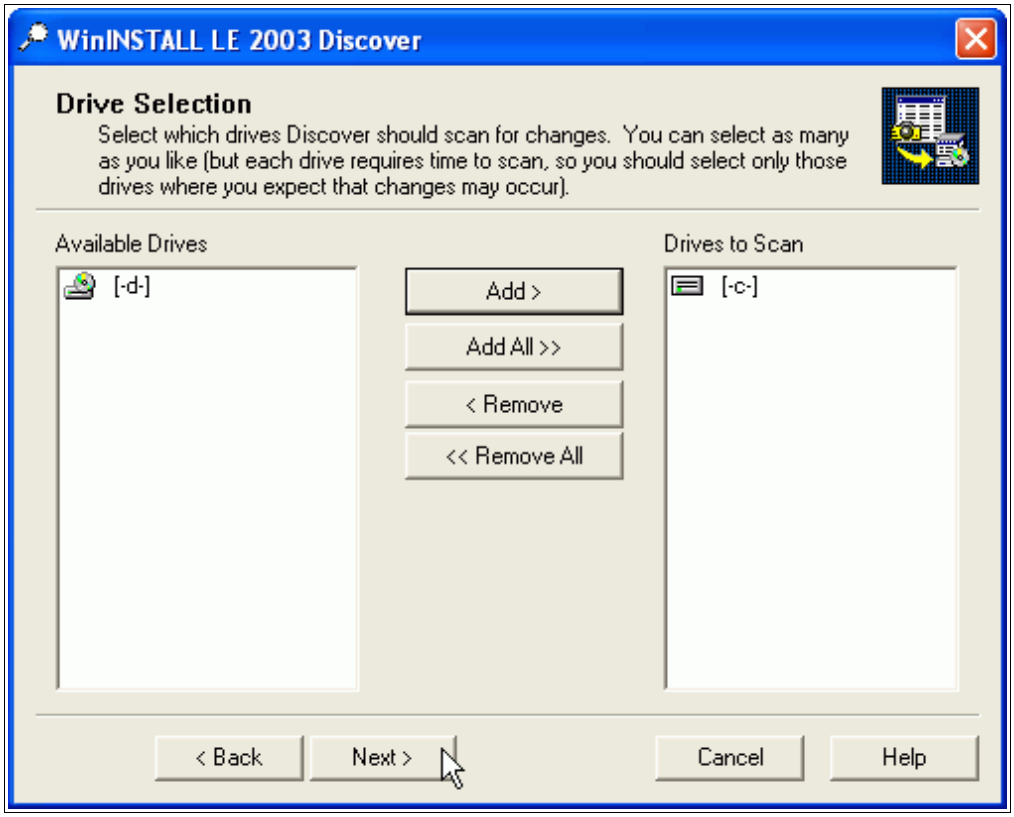
Für die Probeinstallation der zu analysierenden Software wird als Test-Ziel-Client ein frisch installierter **Arbeitsrechner** benötigt. Auf diesem sollte vorab nur eine Minimalkonfiguration eingespielt sein, in der Regel nur das Betriebssystem samt Updates.

Bei der Installation von WinINSTALL LE auf dem Serverrechner fragt das Installationsprogramm nach einer Freigabe (einem Share), an dem es die Installationspakete ablegen kann und den es ggfs. auch einrichtet.

Gestartet wird der mehrschrittige Prozess der Analyse und MSI-Paketierung auf dem Test-Ziel-Client. Dort muss der freigegebene Share aufgesucht werden und das Programm disco32.exe, das „Discover“-Programm gestartet werden. Es führt durch den weiteren Prozess.

Der erste Schritt ist, dass die Anwendung, für die ein Paket erstellt wird, einen Namen erhält – z.B. „Programm 1.0“ - und Pfad und Name des Pakets festgelegt werden. Für jedes Paket und sollte dabei unbedingt ein eigenes Verzeichnis vorgesehen sein.

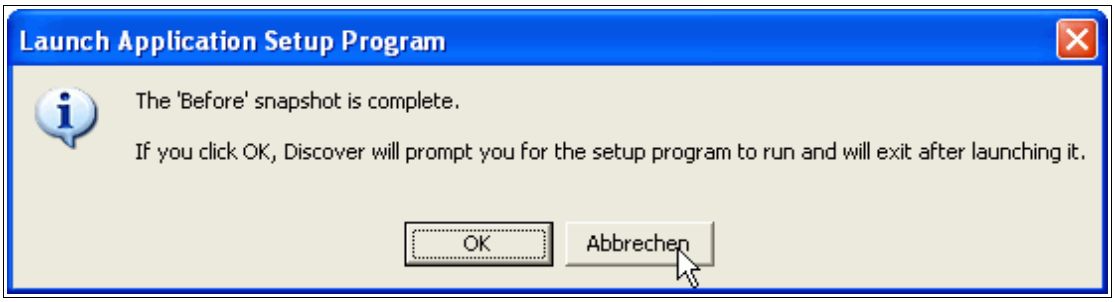
Das Discover-Programm bereitet dann den „Schnappschuss“ des ursprünglichen Systemzustandes vor. Dazu erfragt es, welche Laufwerke in den Schnappschuss einzubeziehen sind. In der Regel wird lediglich C: zu betrachten sein, da Setup-Programme normalerweise andere Laufwerke nicht anfassen (sofern C: das Systemlaufwerk ist).



Weiterhin präsentiert der Discover-Agent eine Liste von Unterverzeichnissen, Registry-Abschnitten und Ini-Dateien, die aus der Analyse ausgeklammert werden können, weil sie temporäre Dateien oder mit hoher Wahrscheinlichkeit keine für die Anwendung relevanten Daten und Einstellungen enthalten. Die Default-Liste kann geändert werden, ohne genaue Kenntnis der Anforderungen der Anwendung belässt man sie am besten zunächst.

Anschließend kann die Erfassung des Ist-Zustand des Systems (1. Schnappschuss) gestartet werden.

Ist dieser Vorgang – der einige Zeit in Anspruch nimmt – abgeschlossen, meldet dies das Discover-Programm und fordert auf, das Original-Setup-Programm der neu zu paketierenden Anwendung zu starten:



Wenn die Anwendung installiert ist – ob dabei ein Reboot, zu dem ggfs. das Original-Setup-Programm auffordert, auch auszuführen ist, muss im jeweiligen Einzelfall abgewogen werden, ohne dass Microsoft hierfür klare Hilfen gibt –, kann die disco32.exe erneut aufgerufen werden. Sie führt dann den zweiten Schnappschuss durch, ermittelt automatisch die Unterschiede zwischen den beiden Systemzuständen. Sodann erstellt sie das MSI-Paket, dessen Installation künftig genau die Unterschiede reproduzieren soll, d.h. als Installationspaket für die gewünschte Anwendung dienen kann.

4.3.1.6.2. Orca

Microsoft fasst die Aufgabe des Programms Orca folgendermaßen zusammen:

Wegen der Beschränkungen existierender Werkzeuge für den Windows Installer, ist es u.U. erforderlich, MSI-Pakete direkt zu editieren. Orca wird vom Windows Installer SDK für diese Aufgabe zur Verfügung gestellt.

Dies weist bereits darauf hin, dass man im Prinzip alles, was mit MSI-Paketen zu tun hat, mit Hilfe von Orca machen kann – dass die Arbeit mit Orca für die meisten Aufgabenstellungen aber zu mühsam zu benutzen ist und sich Orca eher zu Prüfungs- und kleineren Korrekturarbeiten eignet.

Ein Blick mit Orca in ein fertiges MSI-Paket kann auf jeden Fall dazu dienen, das Prinzip der datenbankorientierten Ablage der Setup-Informationen, die die Grundlage der MSI-Architektur bildet, zu beobachten.

Nach Öffnen eines MSI-Pakets mit Orca wird eine Liste von Tabellen angezeigt, die durchaus als Tabellen im Datenbank-Sinn zu verstehen ist.

Klickt man dann etwa die Tabelle „Files“ an, so erscheint ein Bild wie das folgende:

Jede Datei, die für die Installation der Anwendung benötigt wird, ist durch eine Zeile in der tabellarischen Darstellung rechts repräsentiert und eindeutig gekennzeichnet durch den Schlüssel, der in der ersten Spalte steht. Theoretisch kann man nun hier Änderungen an Dateinamen o.a. Vornehmen.

Ähnlich werden Registry-Einträge und zahlreiche andere Komponenten der Installation der Anwendung beschrieben.

Interessant ist, dass auch die Abfolge der Installationsschritte, die ja zentral ist für die Durchführung der Installation, in der speziellen Tabelle „InstallExecuteSequence“ niedergelegt ist. Die Reihenfolge wird dabei durch den Wert in der Spalte „Sequence“ beschrieben, d.h. sortiert man die Tabelle nach dieser Spalte, ist die Reihenfolge erkennbar.

4.3.1.7. Aufbau eines eingebundenen Produkts

Im opsi-Modell wird ein kommerzielles oder freies Softwareprodukt entsprechend den Ergebnissen der Analyse in ein "integriertes" bzw. eingebundenes Produkt überführt. Dieses opsi-Produkt wird durch ein spezifisches Installationsskript beschrieben, das von dem Programm **winst** ausgeführt wird.

4.3.1.7.1. Die Aufgabe des opsi-Windows-Installationsprogramms winst

Im Vergleich zu Windows-eigenen Installationsverfahren oder anderen kommerziell erhältlichen Installern muss die Installation mit **winst** noch folgende Aufgaben erfüllen:

- Das Installationsprogramm muss unproblematisch mit den opsi-Konfigurationsdateien arbeiten, die die Vorgaben für die Produktinstallation auf dem PC enthalten (PCNAME.INI-Dateien, **produkte.txt** u.a.).
- Die Installation sollte entsprechend den Anforderungen der Software-Integration in allen Details gesteuert werden.
- Der **winst** zeichnet sich durch eine – abstellbare – genaue Protokollierung aller Schritte einer Software-Installation aus. Dies ist sowohl für die Integration als auch später für den Support sehr hilfreich.

4.3.1.7.2. Allgemeine Hinweise zum Aufbau eines Winstskriptes

4.3.1.7.2.1. Wenn Installationen einen Reboot erfordern

Wer heutzutage ein Produkt auf einem PC mit dem mitgelieferten Setup-Programm installiert, kennt es zu Genüge: Meist leitet das Setup-Programm von sich aus irgendwann einen Reboot des PCs ein oder es weist zumindest am Ende seiner Tätigkeit darauf hin, dass Einstellungen, die bei der Installation gemacht wurden, erst nach einem Neustart des Systems wirksam werden. Der Grund dafür ist, dass oft nur ein Neustart sichern kann, dass alle Programme auf dem System eventuell geänderte Systemwerte einlesen bzw. mit erneuerten Bibliotheksmodulen arbeiten.

Dieser Erfordernis müssen auch die Installationen mit dem **winst** Rechnung tragen. Dazu wird im **winst-**Skript (in der Aktionen-Sektion) **ExitWindows** aufgerufen. Der Befehl muss durch eine der Optionen **/RebootWanted**, **/Reboot** oder **/ImmediateReboot** näher spezifiziert werden:

- **/RebootWanted** (abgekündigt) ist die schwächste Form und bedeutet, dass die Reboot-Anforderung in der Registry vorgemerkt wird und irgendwann am Ende des Installationsprozesses (möglicherweise auch, wenn noch weitere Installationsskripte gelaufen sind), eingeleitet wird,
- **/Reboot** heißt, dass der Reboot nach vollständigem Abarbeiten des Skripts, in dem der Befehl steht, ausgelöst wird,
- **/ImmediateReboot** stoppt unmittelbar die weitere Abarbeitung des Skripts und stößt den Reboot an. Damit das Skript nach dem Neustart des PCs und Wiederaufruf des

Skripts (was im Rahmen eines opsi-Systems automatisch passiert) nicht wieder auf die selbe Anweisung stößt und in eine Schleife läuft, sondern jetzt den zweiten Teil des Skripts abarbeitet, ist eine Konstruktion in etwa wie folgt erforderlich:

```
Set $WinstRegKey = "HKLM\SOFTWARE\opsi\winst"
Set $RebootFlag = GetRegistryStringValue ("[" + $WinstRegKey + "] " +
"RebootFlag")

if not ($RebootFlag = "1")
     ;=========================
     ;
     ; Anweisungen vor Reboot ...
     ;
     ; hier können jetzt beliebige Anweisungen stehen
     ; Reboot initialisieren
     Set $RebootFlag = "1"
     Registry_SaveRebootFlag
     ExitWindows /ImmediateReboot

else
     ;=========================
     ;
     ; Anweisungen nach Reboot ...
     ;
     ; Rebootflag zurücksetzen
     Set $RebootFlag = "0"
     Registry_SaveRebootFlag

     ; hier steht der Skriptteil nach Reboot
endif
```

Dazu wird noch die folgende Registry-Sektion im Bereich Aktionen benötigt:

```
[Aktionen]
[Registry_SaveRebootFlag]
openKey [$WinstRegKey]
set "RebootFlag" = "$RebootFlag"
```

4.3.1.7.2.2. Dateien kopieren

Die möglichen Kopieroptionen sind im wInst-Handbuch dargestellt.

4.3.1.7.2.3. Startmenü-Einträge

Im **winst-**Skript können die Startmenü-Einträge standardkonform mittels einer Linkfolder-Sektion eingerichtet sowie dem entsprechenden Aufruf eingerichtet werden, Beispiel:

```
[LinkFolder_adminutils]
set_basefolder common_programs
set_subfolder "Admin Utils"
```

```
set_link
  name: Ini-Editor
  target: javaw.exe
  parameters:  -jar %ProgramFilesDir%\opsi.org\inied\inied.jar
  working_dir: $TEMP$
  icon_file: %ProgramFilesDir%\opsi.org\inied\config_prog.ico
  icon_index: 0
end_link

set_link
  name: WinMerge
  target: %ProgramFilesDir%\WinMerge\WinMerge.exe
  parameters:
  working_dir: $TEMP$
  icon_file: %ProgramFilesDir%\WinMerge\WinMerge.exe
  icon_index: 0
end_link
```

4.3.1.7.2.4. Betriebssystem-Abhängigkeiten

Zu Beginn des **winst-**Skripts ist abzusichern, für welches Betriebssystem das Skript geeignet ist bzw. es muss eine Fallunterscheidung für verschiedene Betriebssysteme (bzw. Betriebssystem-"Familien") vorgenommen werden.

In der Aktionen-Sektion des **winst-**Skriptes kann das Betriebssystem bzw. die Betriebssystemfamilie mit der Funktion **GetOS** abgefragt werden. **GetOS** nimmt einen der folgenden Werte an:

```
"Windows_16"
"Windows_95"
"Windows_NT"
```

Windows_95 steht dabei für Win95, Win98 und WinME, Windows_NT für NT 4.0, Windows 2000 und Windows XP.

Mit der Funktion **GetNTVersion** kann anschließend die NT-Version abgefragt werden. Folgende Werte werden angenommen:

```
"NT4"
"Win2k"
"WinXP"
"Win NT 5.2" (z.B. Windows Server 2003 R2 Enterprise Edition)
```

Ein Skript, das alle heute noch faktisch vorkommenden Windows-Betriebssystemfamilien berücksichtigen soll, kann z.B. mit folgender Fallunterscheidung arbeiten:

```
DefVar  $OS$
set $OS$ = GetOS

DefVar $MinorOS$
set $MinorOS$ = GetNTVersion


; gemeinsame Anweisungen für Win95- und WinNT-Familie


; Fallunterscheidungen
if  $OS = "Windows_NT"
      if $MinorOS$ = "NT4"
            ; Anweisungen nur für WinNT
      else
            ; Anweisungen für Win2000/XP
      endif
else
      ; Anweisungen nur für Win95-Familie
endif
```

In einem Skript, das nur für PCs gedacht ist, die dem opsi-Standard NT (also Windows NT, Windows 2000 oder Windows XP) genügen, reicht es, sich gegen versehentliche Installation auf einem ungeeigneten Rechner abzusichern:

```
if GetOS = "Windows_NT"
; es folgen alle Anweisungen des Skripts
endif
; bei einem Nicht-NT-Betriebssystem geschieht gar nichts
```

4.3.1.7.2.5. Optionen im winst-Skript

Für manche Produkte ist es erforderlich, Optionen zur Verfügung zu stellen. So soll der Internet Explorer eigentlich nur als Betriebssystemdateienupdate dienen. Manche Nutzer benötigen aber doch die Browser-Funktionalität des Internet Explorers. Daher gibt es einen Schalter, der es ermöglicht, den Browser mittels opsi mit zu installieren. Dazu wird in der PCNAME.INI ein Schalter eingerichtet, der im wInst-Skript abgefragt wird. Dies kann dann so aussehen:

```
if IniVar ("ie6_exe") = "on"
      Files_CopyC_hiddenexefiles
      Registry "%SCRIPTPATH%\hiddenexe.rgm"
      Registry "%SCRIPTPATH%\hiddenexe.rgu" /AllNTUserDats
endif
```

Es werden dann die notwendigen Dateien kopiert und die Registry-Einträge gemacht.

Es besteht aber auch die Möglichkeit, mit solchen Schaltern ein Programm zu konfigurieren. Ein Beispiel für diese Einsatzform ist im Installationsskript des Virenscanners zu finden. So sind in der PCNAME.INI folgende Schalter eingerichtet:

```
[virscan-install]
....
WriteScan=on
NetworkScan=on
Reinst451=on
```

Der zugehörige Skriptteil in der Installationsdatei lautet:

```
;Konfigurations-Datei patchen und importieren
....
if IniVar ("ReadScan") = "off"
  Set $ReadScan="0"
else
  Set $ReadScan="1"
endif
if IniVar ("WriteScan") = "off"
  Set $WriteScan="0"
else
  Set $WriteScan="1"
endif
if IniVar ("NetworkScan") = "off"
  Set $NetworkScan="0"
else
  Set $NetworkScan="1"
endif
....
```

Diese drei Parameter werden benutzt, um den Vshield zu konfigurieren. Hierzu wird die Konfigurationsdatei vsconfig.ini entsprechend den Schaltern gepatcht und auf den Vshield angewandt.

4.3.1.8. Verfahren zur Deinstallation von Produkten

Um ein Produkt auch wieder von einem Rechner löschen zu können, muss ein Deinstallationsskript existieren. Grundsätzlich besteht bei einer Deinstallation die Schwierigkeit, dass nicht immer klar ist, wie das Produkt auf dem Rechner vorliegt und

was alles entfernt werden muss. Es können neue Dateien oder neue Registry-Einträge zum Produkt nach der Installation hinzugekommen sein. Weiterhin muss darauf geachtet werden, nicht zu viel zu entfernen um nicht die Systemstabilität zu gefährden. Meist weiß nur der Hersteller genau, wie mit seinem Produkt bei der Deinstallation umzugehen ist. Ähnlich wie bei der Installation existieren zu diesem Zweck Deinstallationsroutinen die dem Produkt beiliegen. Wenn es die Möglichkeit gibt, diese ohne Benutzerinteraktion auszuführen, kann dies schon ein entscheidender Schritt sein. Ist eine solche Routine nicht vorhanden oder muss diese erweitert werden, so kennt der Winst Befehle, die zur Deinstallation nützlich sein können. Im Folgenden soll nun ein Überblick über Möglichkeiten zur Deinstallation gegeben werden, die durch Beispiele verdeutlicht werden.

4.3.1.8.1. Verwenden einer Deinstallationsroutine

Liefert der Hersteller des Produkts ein Programm (oder ein MSI-Paket) zur Deinstallation, so muss zunächst geprüft werden, ob dies auch ohne Benutzerinteraktion ausgeführt werden kann (silent-mode). Sollte dies nicht von Hause aus Möglich sein, kann der Einsatz eines autoit-Skriptes zusammen mit der Deinstallationsroutine hilfreich sein. Der Aufruf der ausführbaren Datei kann im Winst-Skript dann in einer Winbatch-Sektion geschehen, z.B.:

```
[Winbatch_start_ThunderbirdUninstall]
%SYSTEMROOT%\UninstallThunderbird.exe /ma
```

Trotz dieser Unterstützung des Herstellers sollte man sich jedoch nicht auf die korrekte Beseitigung des Produkts verlassen und prüfen ob das System nach der Deinstallation auf einem Testsystem weiter stabil läuft oder welche Dateien/Einträge zurückgeblieben sind.

Falls man das Produkt zuvor mittels MSI installiert hat, ist es meist möglich an diesem Paket auch die Deinstallation aufzurufen. Dazu übergibt man das MSI-Paket mit dem Schalter **`/x`** an die **`msiexec.exe`**. Um die Benutzerfragen zu deaktivieren (das Skript läuft dann ohne Benutzerinteraktion durch) existiert der Schalter **`/qb-!`**. Dies ergibt nun folgende Anweisung:

```
msiexec.exe /x meinPaket.msi /qb-!
```

Statt den Namen des Pakets zu übergeben, gibt es auch die Möglichkeit die GUID an **msiexec.exe** zu übergeben. Diese Nummer identifiziert das Produkt im System – sie ist als produktspezifisch. Sie findet sich zum Beispiel im Zweig

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall
```

der Registry. Ein Aufruf mit dieser GUID sieht dann folgendermaßen aus:

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

Sollten diese Methoden nicht oder nicht vollständig funktionieren, so muss mit einem Winst-Skript nachgeholfen werden, wie es der nächste Abschnitt beschreibt.

4.3.1.8.2. Nützliche Winst-Befehle zur Deinstallation

Wurde ein Produkt mit den Winst-Funktionen installiert oder gibt es keine Deinstallation vom Hersteller, so muss ein eigenes Winst-Skript zu Deinstallation geschrieben werden. Um den Programmierer bei dieser Arbeit zu unterstützen kennt der Winst einige Funktionen, die speziell bei der Deinstallation notwendig oder hilfreich sind. Es soll an dieser Stelle ein Überblick gegeben werden, eine genaue Beschreibung der Befehle und deren Parameter findet sich im Winst-Handbuch.

Der einfachste Fall ist das Löschen einer oder mehrerer Dateien vom System. Dies geschieht in einer Files-Sektion mit dem Befehl

```
delete -f Dateiname
```

oder für ein Verzeichnis mit Unterverzeichnissen

```
delete -sf Verzeichnisname
```

Der Parameter f steht dabei für force um die Datei wirklich zu löschen, auch wenn diese schreibgeschützt ist, der Paramtere s für subdirectories (Unterverzeichnisse). Soll eine Datei oder ein Verzeichnis aus allen User-Profilen gelöscht werden so kann diese Files-Sektion mit dem Parameter /AllNTUserProfiles aufgerufen werden. (siehe Winst-Handbuch)

Möchte man einen Verzeichnisbaum löschen in dem sich auch Dateien mit dem Attribut „versteckt“ oder „systemdatei“ befinden, muss im Moment der Umweg über eine

DosInAnIcon-Sektion gegengen werden in der nun der Dos-Befehl ausgeführt werden kann:

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

Muss vor dem Löschen evtl. ein laufender Prozess beendet werden, so kann dies mit dem Namen des Prozesses (zu sehen im Task-Manager) und dem Winst-Befehl killtask geschehen

```
killtask "thunderbird.exe"
```

Sollte das Produkt – oder Teile davon – als Service laufen, so muss dieser vor der Deinstallation beendet werden. Man kann dazu den Service in der Registry auf “inaktiv“ schalten und den Rechner neustarten oder aber man benutzt den Befehl “net stop“ um den Service sofort zu stoppen um anschließend – ohne Neustart – die zugehörigen Dateien zu löschen.

```
net stop
```

Besondere Vorsicht ist beim Löschen von .dll-Dateien geboten, die noch von anderen Produkten verwendet werden könnten. Sie müssen individuell behandelt werden, weshalb hier leider kein allgemein gültiges Rezept gegeben werden kann.

Um einzelne Einträge aus der Registry mit dem Winst zu löschen kommt der Befehl **`DeleteVar`** zum Einsatz der nur in einer Registry-Sektion eines Winst-Skripts verwendet werden kann. Er löscht Einträge aus dem momentan geöffneten Key:

```
      DeleteVar <VarName>
```

Möchte man einen Registry-Key samt seiner Unterschlüssel und Registry-Variablen löschen, so geschieht dies mit dem Winst-Befehl DeleteKey, z.B.:

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

4.3.2. Erstellen eines opsi-Pakets

In opsi werden die Installationsdateien, das Winst-Script zur Installation auf den Client und die Metadaten zu einem Paket zusammengefasst, welches zur Installation dieses Softwareproduktes auf einem opsi-server ident.

Die wesentlichen Vorteile dieses Paketformates sind:

- Vereinfachte menügeführte Erstellung mit dem Programm opsi-newprod.
- Ablage aller relevanten Metadaten in einer einfach zu editierenden Datei.
- Optional menügeführtes Auspacken des Paketes mit der Möglichkeit Vorgaben zu ändern.
- Informationen über die im Paket enthaltene Produktversion, Paketversion und eventueller Kundenspezifischer Erweiterungen werden abgespeichert und sind am Paketnamen erkennbar, werden im Installationsverzeichnis abgelegt und im opsi-Configeditor angezeigt. Auf diese Weise wird der Überblick über unterschiedliche Versionen erleichtert (Productlifecycle Management).
- Zur Erstellung und zum Auspacken von Produkten sind keine root-Rechte erforderlich. Es langen hierzu die Rechte der Gruppe 'pcpatch'.

Das Paket selber besteht aus einem per Gzip komprimierten cpio Archiv. In diesem Archiv befinden sich drei Verzeichnisse:

CLIENT_DATA

Hier liegen die Dateien die im Produktverzeichnis (z.B. /opt/pcbin/install/<productid> landen sollen.

SERVER_DATA

Hier können Verzeichnisse abgelegt werden die nach / ausgepackt werden. (Dann sind zum Auspacken allerdings evtl. auch root Rechte erforderlich).

OPSI

Hier liegen in der Datei 'control' die Metadaten des Produkts wie z.B. Produktabhängigkeiten. Weiterhin finden sich hier die Dateien preinst und postinst die

vor bzw. nach der Installation ausgeführt werden. Hier können Sie soweit benötigt entsprechende Erweiterungen unterbringen.

4.3.2.1. Erstellen, Packen und Auspacken eines neuen Produktes

Zur Erstellung eines neuen Produktes benötigt man mindestens die Rechte der Gruppe pcpatch.

In diesem Beispiel werden die Produkte in dem Verzeichnis /home/opsiproducts erstellt welches der Gruppe pcpatch gehört und die Rechte 2770 hat (Setgroupid Bit für Gruppe pcpatch gesetzt).

Achtung: Im folgenden sollten bei Eingaben keine Umlaute verwendet werden, da die Umsetzung zwischen den verschiedenen Zeichensätzen noch nicht sauber funktioniert.

Zum Erstellen wechselt man in diese Verzeichnis und ruft 'newprod' auf. Das Programm fragt daraufhin nach dem Typ des zu Erstellenden Paketes. Dies ist üblicherweise der Typ 'localboot' für Produkte die über den Preloginloader/Winst installiert werden. Der Typ 'netboot' steht für Produkte die einen bootimage Start auslösen (wie Hardware-Inventarisierung) und der Typ 'server' für Produkte die nur Dinge auf dem Server installieren.

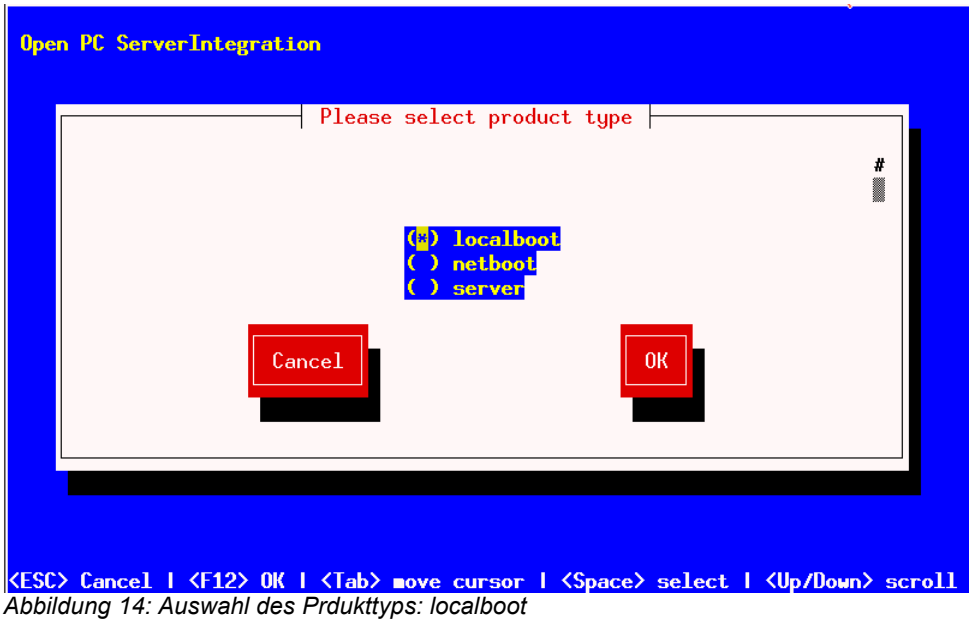


Abbildung 14: Auswahl des Prdukttyps: localboot

Wählen Sie nun mit Tab OK (oder bestätigen mit F12). Nun müssen Sie die wesentlichen Produktdaten eingeben. Am oberen Rand ist hierzu eine Hilfe die Erläutert was die Felder bedeuten.

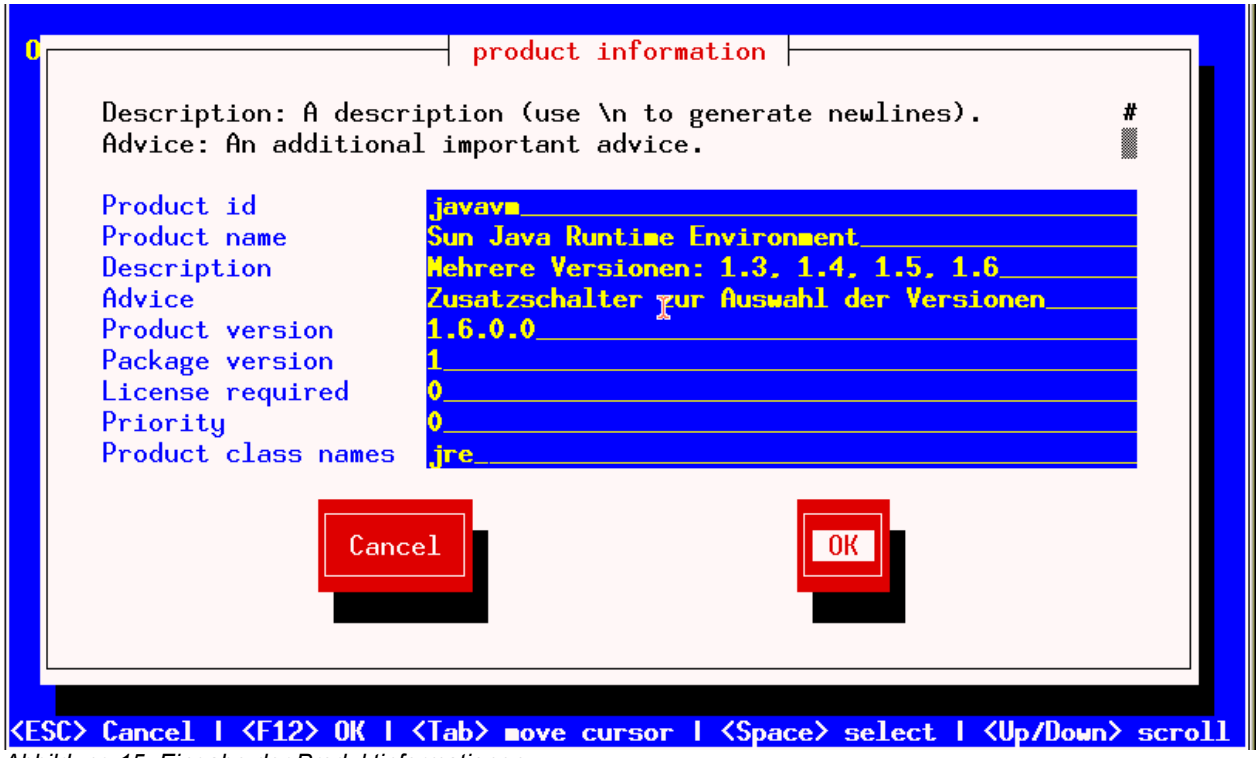


Abbildung 15: Eingabe der Produktinformationen

- 'Product Id' ist ein eindeutiger Bezeichner für das Produkt in der Regel unabhängig von der Version (In opsi 2 hieß das Produktname)
- 'Product name' ist der Klartextname des Produkts
- 'Desciription' ist eine ergänzende Beschreibung zum Produkt die z.B. im opsi-Configeditor unter 'Beschreibung' angezeigt wird.
- 'Advice' ist eine ergänzende Beschreibung in der Regel zum Umgang mit dem Produkt die zu Beachten ist und im im opsi-Configeditor unter 'Notiz' angezeigt wird.
- 'Product version' ist die Version der eingepackten Software
- 'Package Version' ist die Version des Paketes für die Produktversion. Die dient dazu um Pakete mit gleicher Produktversion aber z. B. korrigiertem Winst-Script zu unterscheiden.

- 'Priority' wird zur Zeit noch nicht verwendet. Soll neben Produktabhängikeiten die Installationsreihenfolge beeinflussen.
- 'Product class' wird zur Zeit noch nicht verwendet (und auch nicht angezeigt).

Nach Eingabe der Produktinformationen werden Sie aufgefordert die Skripte anzugeben die Sie für unterschiedliche mögliche Aktionen bereit stellen werden.

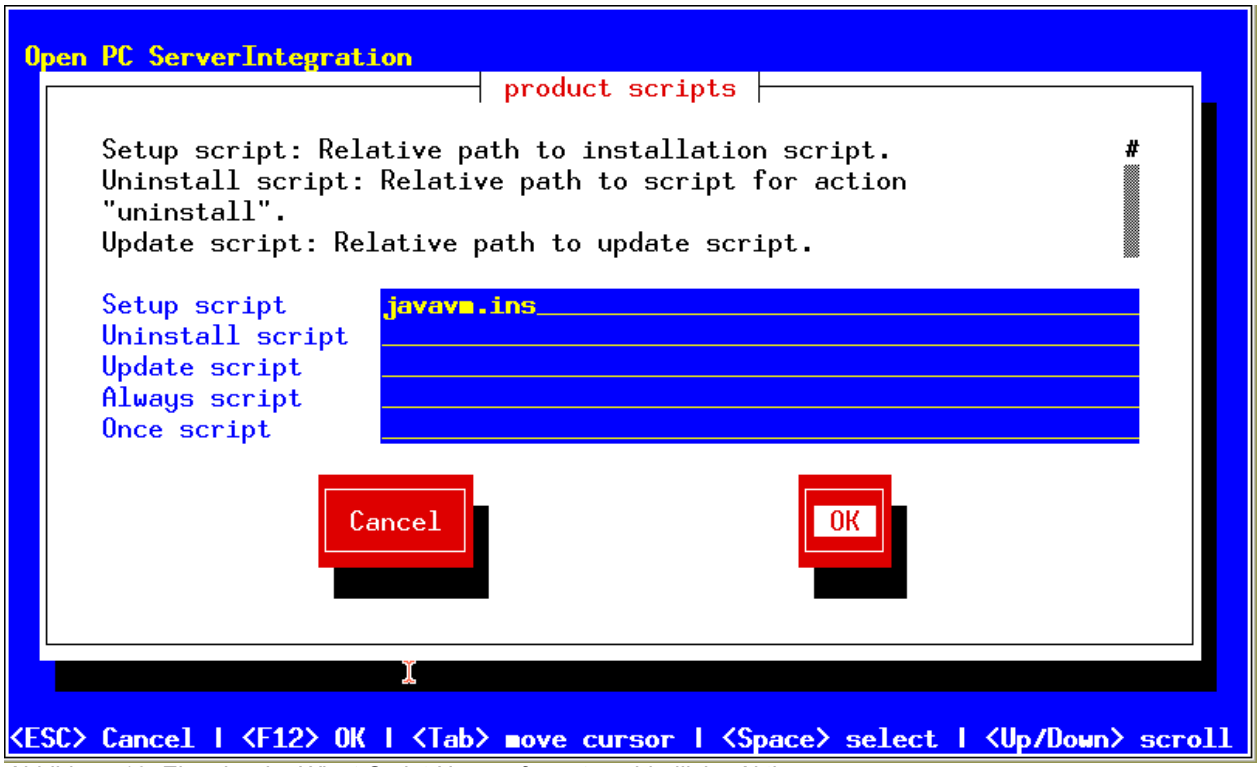


Abbildung 16: Eingabe der Winst-Script Namen für unterschiedliche Aktionen

Üblicherweise heißt das 'Setup script' gleich <product Id>.ins.

Nachdem nun das Produkt selber beschrieben ist, können Sie eine oder mehrere Produktabhängigkeiten definieren. Wollen Sie keine Produktabhängigkeit definieren so geben Sie 'No' ein.

Abbildung 17: Eine (weitere) Produktabhängigkeit definieren: Ja / Nein

Zur Erstellung einer Produktabhängigkeit geben Sie die folgenden Daten an. Beachten Sie auch die Hilfe im oberen Teil des Fensters:

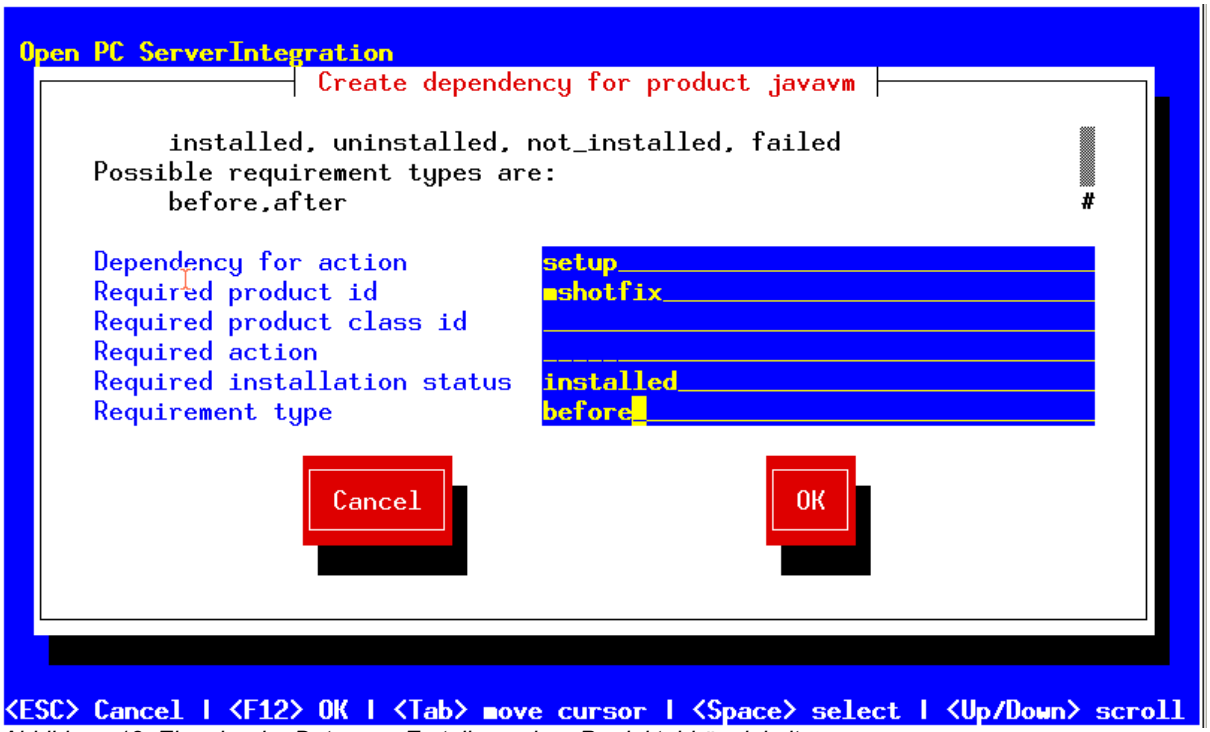


Abbildung 18: Eingabe der Daten zur Erstellung einer Produktabhängigkeit

- 'Dependency for Action' : Für welche Aktion des Produktes welches Sie gerade erstellen soll die Abhängigkeit gelten (setup, deinstall,...)
- 'Required product id': Productid (Bezeichner) des Produkts zu dem eine Abhängigkeit besteht.
- 'Required product class id': wird zur Zeit noch nicht verwendet. Leer lassen ! Bezeichner der Productklasse zu der eine Abhängigkeit besteht.
- 'Required action': Sie können entweder eine Aktion anfordern oder (siehe unten) einen Status. Aktionen können z.B. sein : setup, uninstall, update,...
- 'Required installation status': Status den das Produkt zu dem eine Abhängigkeit besteht haben soll. Typischerweise 'installed', liegt ein anderer Status vor so wird das Produkt auf setup gestellt.
- 'Requirement type': Installationsreihenfolge. Wenn das Produkt zu dem eine Abhängigkeit besteht installiert sein muss bevor mit der Installation des aktuellen Produkts begonnen werden kann dann ist dies 'before'. Muss es nach dem aktuellen Produkt installiert werden so ist dies 'after'. Ist die Reihenfolge egal so muss hier nichts eingetragen werden.

Hinweis:

Leider gibt es derzeit nicht wirklich einen generischen Mechanismus für Deinstallations-Produktabhängigkeiten. Zuverlässig ist der ProductDependency-Mechanismus nur für action: setup und die hierbei zu triggernden (before- oder after-) setup Aktionen und installed Status. Ein requiredAction: uninstall führt leider definitiv zu Fehlern.

Nach dem eine Produktabhängigkeit definiert ist, werden Sie wieder gefragt ob Sie eine (weitere) Produktabhängigkeit definieren wollen. Wenn ja wiederholt sich der Vorgang; wenn nein, so werden Sie gefragt ob sie eine Produkteigenschaft (Zusatzschalter) definieren wollen mit dem Sie die Installation des Produktes modifizieren können.

Antworten Sie ja so müssen Sie die Produkteigenschaft beschreiben:

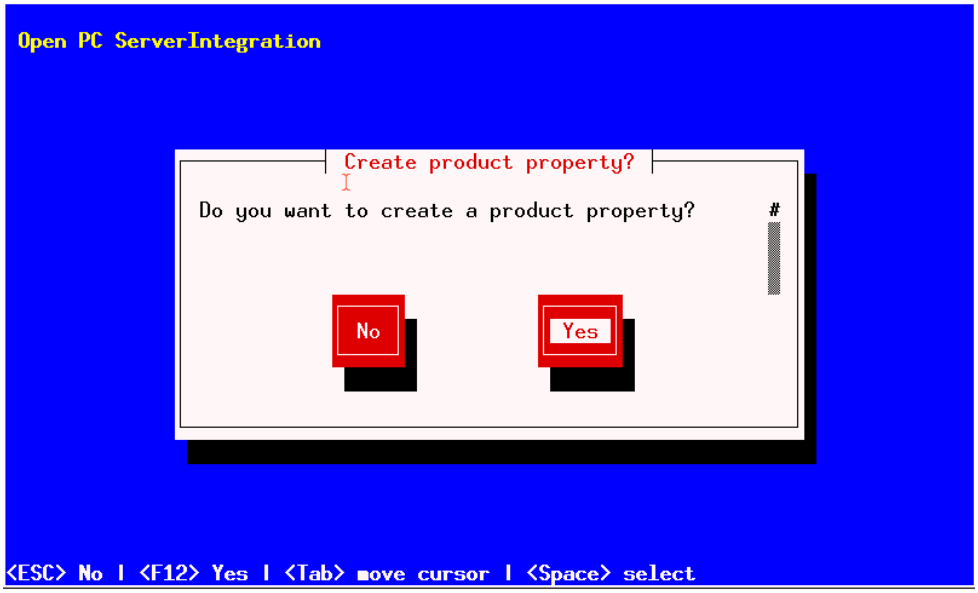


Abbildung 19: Eine (weitere) Produkteigenschaft definieren ?

Die Produkteigenschaft wird clientspezifisch gespeichert und besteht aus einem Namen (key) der verschiedene Werte (Values) zugeordnet bekommen kann und die dann vom Winst-Script aus abgefragt werden können. Weiterhin wird eine Beschreibung (Product description) benötigt die beim Auspacken des Produkts und im opsi-Configedit als Hilfe angezeigt wird. Weiterhin müssen Sie, durch Komma getrennt, alle Werte angeben, die der Key annehmen darf. Wird hier nichts angegeben so kann später im opsi-Configeditor ein beliebiger Wert eingegeben werden.

Im Folgefenster müssen Sie festlegen was der Defaultwert dieser Produkteigenschaft ist.

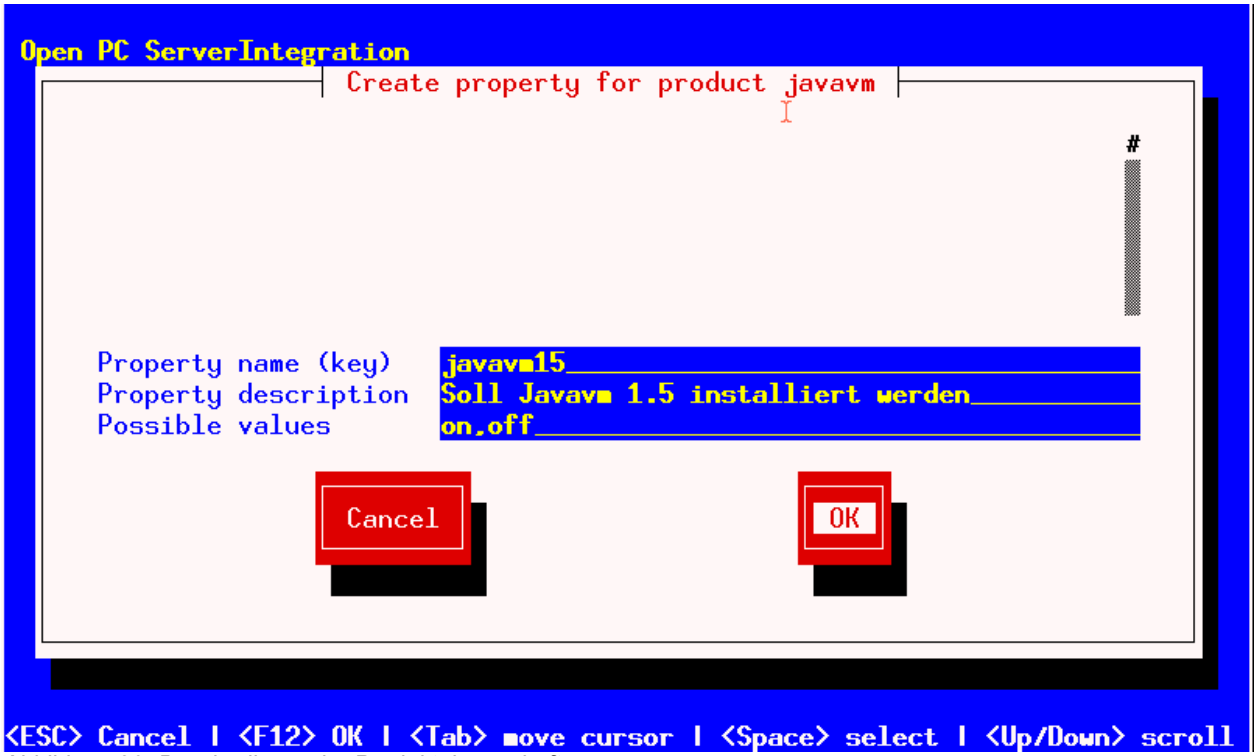


Abbildung 20: Beschreibung der Produkteigenschaft

Im folgenden Fenster entscheiden Sie welches der default Wert ist.

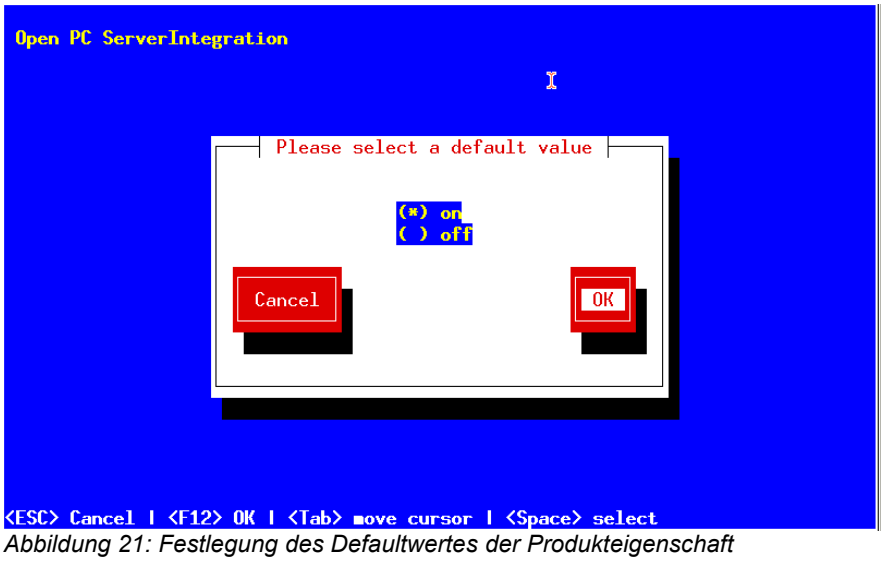


Abbildung 21: Festlegung des Defaultwertes der Produkteigenschaft

Nach dem eine Produkteigenschaft definiert ist, werden Sie wieder gefragt ob Sie eine (weitere) Produkteigenschaft definieren wollen. Wenn ja wiederholt sich der Vorgang; wenn nein, so ist das Grundgerüst des Produktes fertig gestellt.

Mit Hilfe des ls-Befehls finden Sie die oben beschriebene Verzeichnis Struktur. Wechseln Sie in den OPSI-Ordner und setzen Sie erneut den ls-Befehl ab. Hier befindet sich unter anderem die 'control'-Datei, welche die eben eingegebenen Daten enthält und Ihnen auch die Möglichkeit bietet, diese im Editor zu kontrollieren oder zu modifizieren.

Beispiel einer 'control' Datei:

```
[Product]
type: localboot
id: javavm
name: Sun Java Runtime Environment
description:
Mehrere Versionen: 1.3, 1.4, 1.5, 1.6
advice: Zusatzschalter zur Auswahl der Versionen
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 wird default JRE; off=Die aktuellste installierte
Javavm wird default JRE
values: on, off
default: off

[ProductProperty]
name: javavm15
description: Soll Javavm 1.5 installiert werden
values: on, off
default: on

[ProductProperty]
name: javavm16
description: Soll Javavm 1.6 installiert werden
values: on, off
default: off
```

Als nächstes müssen Sie Ihr für das Produkt erstellte Winst-Script und die entsprechenden Dateien nach CLIENT_DATA kopieren.

Danach können Sie das Produkt packen. Gehen Sie dazu in das Stammverzeichnis des Produkts und rufen Sie 'opsi-makeproductfile' auf. Es wird nun das Produkt gepackt.

opsi-makeproductfile kennt einige Optionen die sein Verhalten modifizieren:

```
# 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
```

Das gepackte Paket können Sie mit 'opsi-packet-manager -i <paketname>' auf dem Server installieren.

Weitere Informationen zu opsi-packet-manager siehe:
Kapitel 3.4 Werkzeug opsi-package-manager: opsi-Pakete (de-) installieren Seite 25

4.3.2.2. Erstellung kundenspezifischer opsi-Pakete

Tritt der Fall ein, das ein Produkt auf mehreren Servern unterschiedlich installiert werden soll, z.B. wegen Lizenzgründen oder wegen spezieller kundenspezifischer Anpassungen, müssen vor dem Packen des Produktes die spezifischen Dateien in getrennten Verzeichnissen gespeichert werden. Sie können dann mit **`makeproductfile -i`** kundenspezifisch gepackt werden. Am besten lässt sich das Vorgehen anhand eines Beispiels verdeutlichen.

Beispiel:
Das Beispielprodukt „softprod“ soll in 3 Versionen erstellt werden. Es gibt eine Grundversion und 2 weitere Versionen mit zusätzlicher Software, einmal für FirmaX und für KundeY. Dazu werden im Pfad:
`/home/opsiproduct/softprod/`

die Verzeichnise **CLIENT_DATA.FirmaX** und **CLIENT_DATA.KundeY** angelegt. In diese Verzeichnisse wird nun das Unterverzeichnis **custom_ins_dir** angelegt welche nun die *zusätzliche* Software zur jeweiligen Version aufnimmt. Es ist auch möglich dort ein weiteres Winst-Skript zu erzeugen, das dann vom eigentlichen Setup-Winst-Skript aufgerufen werden muss. Dies ermöglicht einen sehr leichten Umgang, denn wir können nun einfach das (Sub-)Winst-Skript starten mit:

```
if FileExists("%ScriptPath%\custom_ins_dir\custom.ins")
  sub "%ScriptPath%\custom_ins_dir\custom.ins"
endif
```

Zum Packen des Produktes wechseln wir nun, wie gewohnt, in das Verzeichnis Stammverzeichnis des Produkts (**/home/opsiproduct/softprod/**). Wollen wir nun das Produkt für FirmaX packen geben wir ein **makeproductfile -i FirmaX** (es ist unbedingt notwendig, das der Bezeichner den gleichen Namen trägt, auf den das dazugehörige **CLIENT_DATA**-Verzeichnis endet! Also: **CLIENT_DATA.<Versionsname>** => **makeproductfile -i <Versionsname>**) Der Aufruf zum packen des Pakets für KundeY ist analog dazu: **makeproductfile -i KundY**

Die Grundversion des Paketes (also ohne **custom_ins_dir**-Verzeichnisse) wird gepackt, indem makeproductfile ohne Parameter aufgerufen wird.

Das nun entstandene **softprod.<productversion>-<paketversion_<kunde>.opsi** kann nun wie gewohnt verteilt und installiert werden.

5. Netboot Produkte

5.1. Automatische Betriebssysteminstallation unattended

5.1.1. Überblick

Ablauf einer Reinstallation:

- Bei PXE-Boot:
  - Über den opsi-Configed oder opsi-admin wird der PC für die Neuinstallation ausgewählt.
- Der Client erkennt beim nächsten Bootvorgang mit Hilfe des PXE-Bootproms, dass er reinstalliert werden soll und lädt ein Bootimage vom opsi-server.
- Bei CD-Boot:
  - Der Client bootet von der opsi-bootcd das Bootimage.
- Das Bootimage stellt am Client die Rückfrage, ob der PC tatsächlich reinstalliert werden soll. Dies ist die einzige Interaktion des gesamten Prozesses.
- Das Bootimage partitioniert und formatiert die Festplatte.
- Das Bootimage überträgt die notwendigen Installationsdateien und Konfigurations-informationen vom opsi-server auf den Client und leitet einen Reboot ein.
- Nach dem Reboot installiert der Client selbstständig das Betriebssystem anhand der übertragenen Konfigurationsinformationen.
- Im Anschluss wird der opsi-PreLoginLoader zur Einbindung der automatischen Softwareverteilung installiert.
- Die automatische Softwareverteilung installiert die gesamte Software, die gemäß Konfigurationsdatei auf diesen Rechner gehört.

5.1.2. Voraussetzungen

Es muss ein opsi-server installiert sein.

Der Client-PC sollte mit einer bootfähigen Netzwerkkarte ausgestattet sein. Viele heute eingesetzte Netzwerkkarten verfügen über eine entsprechende PXE-Firmware. Diese kontrolliert den Bootvorgang vom Netz, falls nicht eine andere Reihenfolge im BIOS eingestellt ist. Ist kein PXE vorhanden kann alternativ kann auch von der opsi-client-bootcd das Bootimage gebootet werden.

Das opsi-Installationspaket für das zu installierende Betriebssystem muss auf dem opsi-server installiert sein. In den folgenden Abschnitten wird als Beispiel jeweils Windows 2000 angenommen.

5.1.3. PC-Client bootet vom Netz

Die Firmware des PXE wird beim Starten eines PC's aktiv: sie „kann“ dhcp und führt die Abfragen im Netz durch.

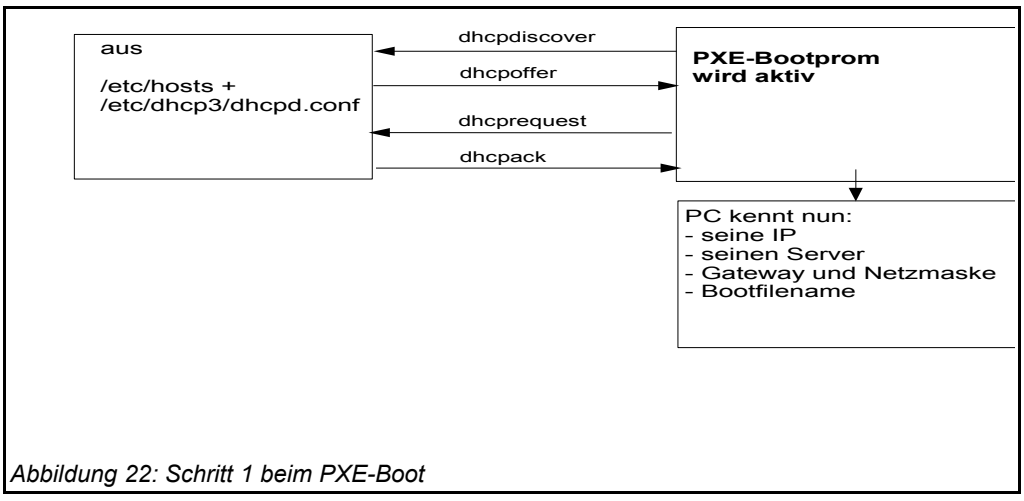


Abbildung 22: Schritt 1 beim PXE-Boot

Der PC kennt zu Beginn lediglich seine Hardware-Adresse (= hardware ethernet, MAC-Nummer der Netzwerkkarte), bestehend aus sechs zweistelligen Hexadezimalzeichen.

Die Firmware schickt damit eine Rundfrage ins Netz. Es ist eine **DHCPDISCOVER**-Anfrage über Standard-Port per Broadcast (= an alle Rechner im Netz): „Ich brauche eine IP-Nummer und wer ist mein dhcp-Server?“ (Discover= entdecken)

Mittels **DHCPOFFER** macht der dhcp-Server diesbezüglich einen Vorschlag. (offer=anbieten)

DHCPREQUEST ist die Antwort des Clients an den Server (wenn er die angebotene IP akzeptiert; Hintergrund ist hier: Es können in einem Netz mehrere dhcp-Server tätig sein.). Der Client fordert damit die angebotene Adresse an. (request=Anfrage)

Mit **DHCPACK** bestätigt der dhcp-Server diese Anforderung des Clients. Die Informationen werden an den Client übertragen. (acknowledge=bestätigen)

Am Bildschirm des bootenden PC's können diese Prozesse mitverfolgt werden. Nach den ersten Systeminformationen meldet sich das PXE-BOOTPROM mit seinen technischen Daten und stellt seine „CLIENT MAC ADDR“ dar. Im Anschluss zeigt ein sich drehendes Pipe-Zeichen die Dauer der Anfrage des Clients an. Wird das bewegliche Zeichen durch einen Backslash ersetzt, hat der dhcp-Server ein Angebot gemacht („CLIENT IP, MASK, DHCP IP, GATEWAY IP“).
Kurze Zeit später – wenn alles funktioniert hat – meldet das PXE: „My IP ADDRESS SEEMS TO BE .....“.

Nach dem Empfang und der Verarbeitung dieser Konfigurationsinformationen durch den PC ist dieser als Netzwerkteilnehmer ordentlich konfiguriert.

Der nächste Schritt ist, das in den Konfigurationsinformationen angegebene Bootfile (bootimage) zu laden.

5.1.3.1. pxelinux wird geladen

Das bootimage wird per tftp (trivial file transfer protocol) geladen. (Meldung auf dem PC-Bildschirm: „**LOADING**“). Das Protokoll tftp ist zum Übertragen von Dateien, bei dem sich der Client nicht authentifizieren muss. Das heisst, die über tftp ladbaren Dateien sind für alle im Netz verfügbar. Daher wird der Zugriff per tftp auf ein bestimmtes Verzeichnis (mit Unterverzeichnissen) beschränkt. Gewöhnlich ist dieses Verzeichnis

/tftpboot. Konfiguriert ist dies in der Konfigurationsdatei des inetd (/etc/inetd.conf), der den eigentlichen tftpd bei Bedarf startet. (z.B. **`tftpd -p -u tftp -s /tftpboot`**).

Der Ladevorgang gemäß dem PXE-Standard ist dabei mehrstufig:

In der ersten Stufe wird die per tftp übermittelte Datei (üblicherweise /tftpboot/linux/pxelinux.0) geladen und gestartet.

Das Programm pxelinux.0 sucht bei Ausführung im Verzeichnis /tftpboot/linux/pxelinux.cfg nach Konfigurations- bzw. Bootinformationen. Dabei wird zunächst nach PC-spezifischen Informationen gesucht. Eine solche PC-spezifische Datei basiert auf der Hardwareadresse (MAC-Adresse) der Netzwerkkarte im Dateinamen. Die Datei ist eine 'Einweg'-Datei (named pipe) und kann daher nur einmal gelesen werden. Der Hardwareadresse im Dateinamen werden dabei immer die zwei Ziffern 01 vorangestellt. Alle Zeichenpaare werden durch ein Minuszeichen verknüpft, z.B. 01-00-0c-29-11-6b-d2 für eine Netzwerkkarte mit MAC: 00:0C:29:11:6B:D2. Wird eine solche Datei nicht gefunden wird nach einer Datei gesucht deren Namen der Hexadezimaldarstellung der IP-Adresse entspricht. Ist auch keine solche PC-spezifische Datei vorhanden, wird pxelinux.0 den Dateinamen (von hinten beginnend) immer weiter verkürzt suchen, bis die Suche ergebnislos verlaufen ist und bei der Datei „default“ endet. Diese Datei enthält den Befehl hdboot. Lädt der PC diese Datei, findet also keine Installation statt, sondern das lokal installierte Betriebssystem wird gestartet.

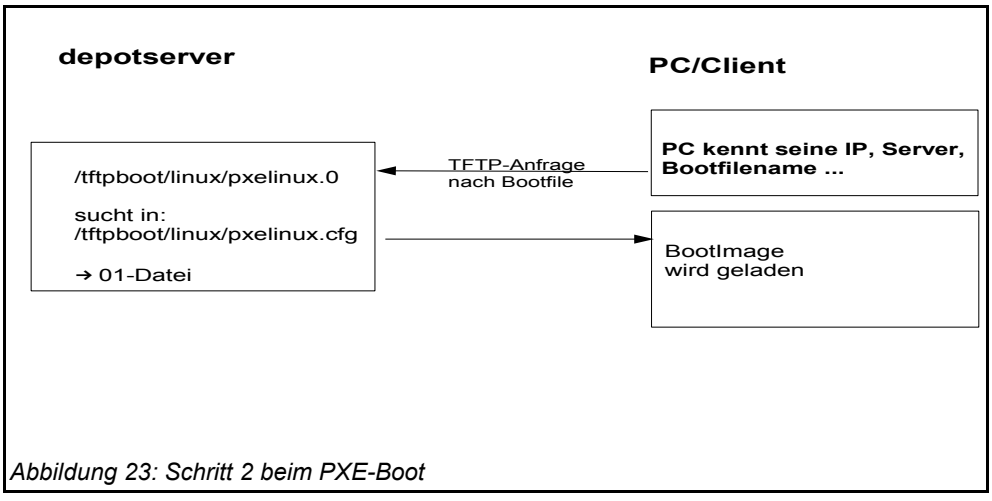


Abbildung 23: Schritt 2 beim PXE-Boot

Um für einen bestimmten PC eine Reinstallation einzuleiten, wird das Programm pxelinux.0 dazu gebracht, in einer zweiten Stufe ein Installationsbootimage zu laden. Dazu wird mit Hilfe des opsipxeconfd eine PC-spezifische Datei in /tftpboot/linux/pxelinux.cfg erzeugt, in der unter anderem der Befehl zum Laden des eigentlichen Installationsbootimages liegt. Weiterhin findet sich hier der PC-spezifische Schlüssel zur Entschlüsselung des pcpatch-Passwortes. Diese Datei wird als 'named pipe' erzeugt und ist damit eine 'Einweg'-Datei die durch einmaliges Lesen von selbst verschwindet. Details hierzu in den Kapiteln zur Absicherung der Shares und zum opsi-pxeconfd.

Linux Installationsbootimage wird geladen

Basierend auf den Informationen die das pxelinux.0 aus der named pipe gelesen hat, wird nun per tftp vom opsi-server das eigentliche Installationsbootimage geladen. Dieses besteht üblicherweise aus dem Kernel (/tftpboot/linux/install) in dem dazugehörigen initrd Filesystem (/tftpboot/linux/miniroot.gz).

Das Bootimage, das nun geladen wird, ist Linux basiert und hat etwa eine Größe von 40 MB.

5.1.4. PC-Client bootet von CD

Analog zu dem Bootvorgang per tftp mit Hilfe des PXE-bootproms kann das Installationsbootimage auch direkt von der opsi-bootcd geladen werden.

Diese Möglichkeit bietet sich bei folgenden Voraussetzungen an:

- der Client verfügt über kein PXE;
- es gibt kein dhcp;
- es gibt dhcp aber es sollen dort keine Einträge zu den Clients gemacht werden und die Hardwareadressen der Clients sind nicht bekannt;
- es gibt dhcp aber dieses ist nicht korrekt konfigurierbar

Entsprechend der unterschiedlichen Situationen müssen dem Bootimage auf der CD unterschiedlich viele Informationen interaktiv bereitgestellt werden. Im einfachsten Fall

müssen überhaupt keine Angaben gemacht werden. Evtl. muss der gewünschte IP-Name mit hn=<hostname> übergeben werden. Es können auch mit der Option ASK_CONF=1 eine ganze Reihe von Parametern abgefragt werden. Den genauen Syntax der Möglichkeiten gibt die Taste F1 am Bootprompt der opsi-bootcd an.

5.1.5. Das Linux Installationsbootimage bereitet die Reinstallation vor

Das Bootimage startet eine erneute dhcp-Anfrage und konfiguriert sich entsprechend sein Netzwerkinterface. Danach werden über den opsi-webservice die Konfigurationsdaten für diesen Client geladen.

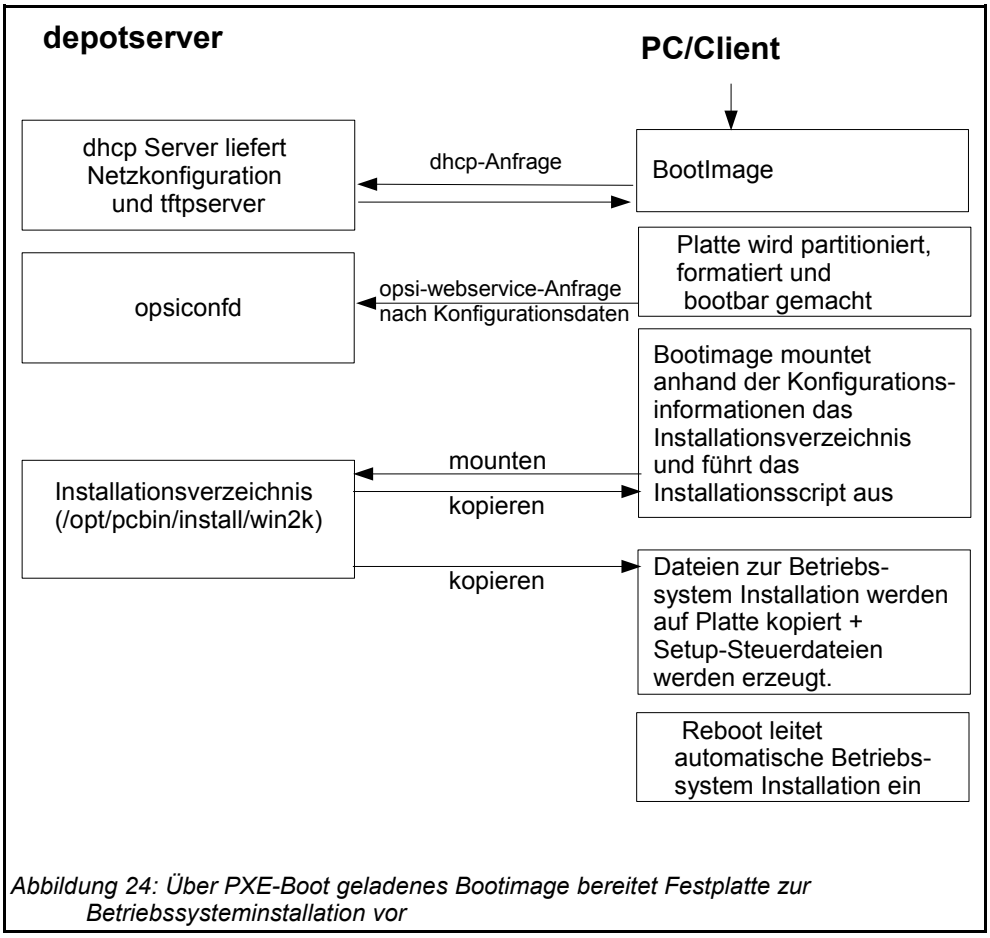


Abbildung 24: Über PXE-Boot geladenes Bootimage bereitet Festplatte zur Betriebssysteminstallation vor

Ergänzt wird dieses Informationspaket durch Angaben aus der dhcp-Antwort (z.B. wer ist der tftp-Server). Die gesammelten Informationen werden für die Weiterverarbeitung durch das eigentliche Installationsskript bereitgestellt.

Nun wird das Passwort des Installations-Users pcpatch mit Hilfe des übergebenen Schlüssels entschlüsselt und der angegebene Installationsshare gemountet. Jetzt kann das auf dem gemounteten Share liegende Installationsskript für das zu installierende Betriebssystem gestartet werden. Die Abläufe in diesem Skript sind abhängig von dem zu installierenden Betriebssystem. Im Folgenden werden beispielhaft die Abläufe für eine Windows-XP-Installation skizziert.

Vorbereitung der Festplatte: Auf der Platte wird eine 6 GB große FAT32 Partitionen angelegt, formatiert und bootfähig gemacht.

Kopieren der Installationsdateien: Die Dateien für die Installation des Betriebssystems werden von dem Installationsshare des Servers (z.B. /opt/pcbin/install/winxppro/i386) auf die lokale Platte kopiert. Das Gleiche gilt für das Setup-Programm des opsi-PreLoginLoaders zur Einrichtung der automatischen Softwareverteilung auf dem PC.

Einpflegen der Konfigurationsinformationen: Unter denen auf die lokale Platte kopierten Dateien finden sich auch Konfigurations- und Steuerdateien, die Platzhalter enthalten. Durch ein spezielles Skript (patcha) werden diese durch entsprechende Parameter aus dem Informationspaket ersetzt (gepatcht), welches das Bootimage zuvor aus Konfigurationsdateien und dhcp-Antwort bereitgestellt hat. Ein Beispiel für eine zu patchende Datei ist die unattend.txt. Sie steuert das „unbeaufsichtigte“ Installieren von Windows 2000/XP.

Reboot vorbereiten: Der Bootloader wir so konfiguriert, dass beim nächsten Boot der Rechner via ntloader in das Windows Setup-Programm startet. Der Aufruf ist dabei mit der Option versehen, die gepatchte unattend.txt als Steuerdatei zu verwenden.

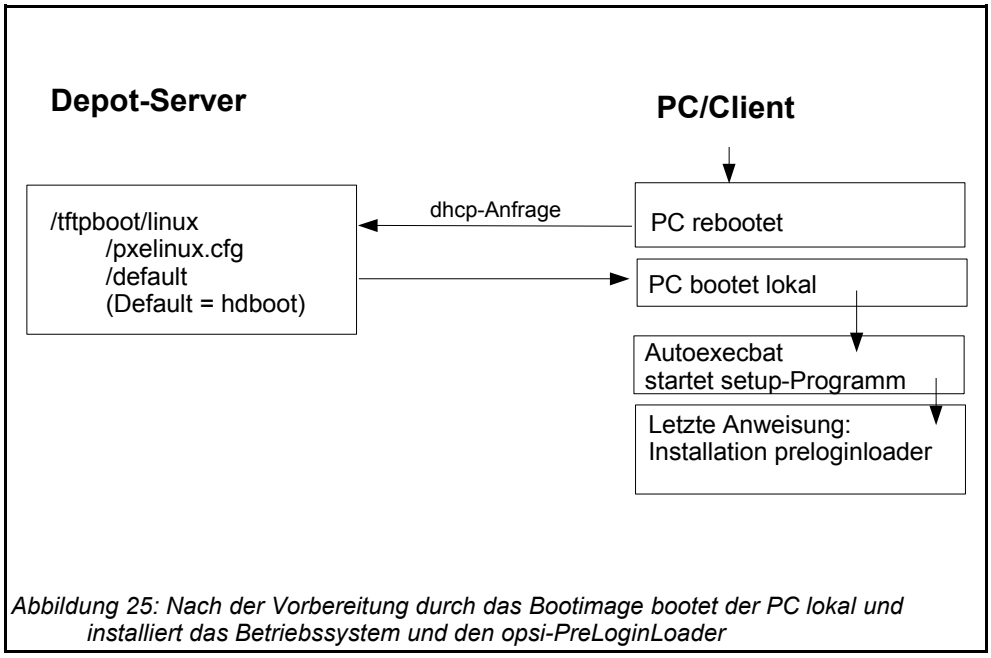


Abbildung 25: Nach der Vorbereitung durch das Bootimage bootet der PC lokal und installiert das Betriebssystem und den opsi-PreLoginLoader

Reboot: Da in /tftpboot/linux/pxelinux.cfg nun keine PC-spezifische Datei mehr vorhanden ist, wird in Stufe 1 des PXE-Boots der Befehl hdboot aus der Datei default geladen. Damit wird der lokale Bootloader gestartet und die Betriebssysteminstallation gestartet.

Die beschriebenen Abläufe werden von dem für diese Installation angegebenen Python-Script gesteuert (z.B. winxppro.py für die Installation von Windows XP). Hierzu stellt das bootimage eine Pythonbibliothek bereit die im gesonderten opsi-bootimage Handbuch beschrieben ist.

5.1.6. Die Installation von Betriebssystem und opsi-PreLoginLoader

Die Installation des Betriebssystems ist ein unattended Setup wie es von Microsoft vorgesehen ist. Dabei werden die Standardmöglichkeiten der Hardwareerkennung genutzt. Im Gegensatz zu einer normalen Installation von CD können auf dem Installations-Share schon aktualisierte Treiber und Servicepacks eingepflegt werden, damit diese schon direkt bei der Erstinstallation verwendet werden.

Zu einer unattended Installation gehört die Möglichkeit, nach Abschluss der eigentlichen Betriebssysteminstallation automatisch noch weitere Installationen starten zu können. Dieser Mechanismus wird genutzt, um das Setup des opsi-PreLoginLoaders aus-

zuführen und damit die automatische Softwareverteilung einzubinden. In der Registry wird eingetragen, dass sich der Rechner immer noch im Reinstallationsmodus befindet.

Nach dem abschließenden Reboot starten nun vor einem Login die opsi-Programme zur Softwareverteilung. Diese Software erkennt anhand der Registry den Reinstallationsmodus. Dieser Modus hat hier zur Folge, dass alle Softwarepakete, die in der Softwarekonfigurationsdatei (<pcname>.ini) für diesen PC auf **setup** und/oder **installed** stehen, nun installiert werden. Auf diese Weise werden sämtlich Pakete, die vor der Reinstallation des Betriebssystems auf diesem PC waren, automatisch wieder eingespielt. Erst nach Abschluss aller Installationen wird der Reinstallationsmodus zum Standard-Bootmodus zurückgeschaltet. (Im Gegensatz zum Reinstallationsmodus, bei dem alle Pakete installiert werden, die auf on oder setup stehen, werden im Standard-Bootmodus nur Pakete installiert, die auf setup stehen.) Damit ist der PC fertig installiert.

5.1.7. Funktionsweise des patcha Programms

Wie oben erläutert werden vom bootimage (genauer gesagt vom Programm /usr/local/bin/master.py) die Konfigurationsinformationen aus dem opsi-webservice und dhcp gesammelt um sie dann in entsprechende andere Konfigurationsdateien wie z.B. die unattended.txt einzupflegen. Das Einpflegen übernimmt das Programm /usr/local/bin/patcha.

Das Skript gleicht anhand eines Suchmusters **`#@flagname(*)#`** eine Konfigurationsdatei mit den Einträgen aus einer anderen Datei (hier cmdline) ab, die Einträge der Art "Flagname=Wert" enthalten muß und patcht diese bei Übereinstimmung des Suchmusters. Das Suchmuster kann nach dem Flagnamen einen "*" enthalten und muß einen oder beliebig viele "#" als Abschluß enthalten. Default wird /proc/cmdline benutzt.

Wenn man patcha ohne irgendwelche Optionen und ohne Dateiübergabe aufruft, werden die 'Flagname=Wert'-Paare aus der /proc/cmdline ausgegeben.

Wenn man **`patcha dateiname`** eingibt, patcht er die datei mittels der /proc/cmdline.

Eine andere cmdline als /proc/cmdline, gibt man mit **`patcha -f andere_cmdline`** mit. Ohne zusätzlich mitgegebenen Dateinamen werden die Werte der andere_cmdline

ausgegeben, mit Dateiname wird die Datei mit den Werten aus andere_cmdline gepatcht.

```
Version 0.93 23.10.2003 (c) J.W.\n";

##############################################################################

Usage:

Aufruf: $prog [Optionen] [cmdline] [file]\n";
Optionen:  -v   gibt nur Versionskennung aus\n";
           -f   gibt anderen Pfad zur cmdline mit\n";
           -h   gibt diese Hilfe aus\n";
$prog patcht Datei anhand eines Strings aus /proc/cmdline (default), der
Flag=Wert enthaelt,
mittels eines Flagsuchmusters
(Ohne Parameter werden default nur Werte aus der /proc/cmdline ausgegeben und
nichts gepatcht).\n";
```

patcha patcht nur einen Tag pro Zeile

Der Platzhalter wird auf die Länge des zu ersetzenden Wertes getrimmt bzw erweitert und dann ersetzt. D.h unabhängig von der Laenge des Platzhalters wird dieser durch den Wert ersetzt. Anhängende Zeichen bleiben anhängend.
Beispiel:

Mit der Datei

```
cat try.in
 tag1=hallohallohallo1 tag2=t2
```

und der Datei

```
cat patch.me
<#@tag1############################>
<#@tag2############################>
<#@tag1#>
<#@tag2#>
<#@tag1*############################>
<#@tag2*############################>
<#@tag1*#>
<#@tag2*#>
<#@tag1#><#@tag1#####>
<#@tag2*########><#@tag1#>
```

ergibt

```
./patcha -f try.in patch.me

cat patch.me
<hallohallohallo1>
<t2>
```

```
<hallohallohallo1>
<t2>
<hallohallohallo1>
<t2>
<hallohallohallo1>
<t2>
<hallohallohallo1><#@tag1#####>
<t2><#@tag1#>
```

5.1.8. Aufbau der Produkte zur unattended Installation

Diese Kapitel betrifft die folgenden Produkte ab opsi Version 3.3.1:

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

5.1.8.1. Übersicht des Verzeichnisbaums

```
<productid>┐
           ├─i386/                          NT5 only: Installations files
           ├─installfiles/                  NT6/7 only: Installations files
           ├─winpe/                         NT6/7 only
           ├─opsi/                          Scripte und Templates by opsi.org
           │   ├─$oem$/                     $oem$ gemaess MS
           │   ├─posinst.d/                 Scripte nach OS-Install by opsi.org
           │   └─unattend.txt.template      Template by opsi.org
           ├─custom/                        Scripte und Templates by customer
           │   ├─$oem$/                     $oem$ gemaess MS by customer
           │   ├─posinst.d/                 Scripte nach OS-Install by customer
           │   └─unattend.txt               unattend.txt by customer
           ├─drivers/                       drivers Verzeichnis
           │   ├─drivers/                   drivers Verzeichnis
           │   ├─pciids/                    Symlinkbaum zu Treibern
           │   ├─vendors/                   Symlinkbaum zu Treibern
           │   ├─classes/                   Symlinkbaum zu Treibern
           │   ├─usbids/                    Symlinkbaum zu Treibern
           │   ├─hdaudioids/                Symlinkbaum zu Treibern
           │   ├─pci.ids                    PCI-IDs DB
           │   └─usb.ids                    USB-IDs DB
           ├─setup.py                       Installationsscript
           ├─<productid>_<version>.control      Meta Daten (nur zur Info)
           ├─<productid>.files              Dateiliste (automatisch erstellt)
           ├─create_driver_links.py         Script zur Treiberverwaltung
           ├─show_drivers.py                Script zur Treiberverwaltung
           ├─download_driver_pack.py        Script zur Treiberverwaltung
           └─extract_driver_pack.py         Script zur Treiberverwaltung
```

5.1.8.2. Die Dateien

•setup.py
Dies ist das Installationsscript, welches vom Bootimage ausgeführt wird.

•<productid>_<version>.control
enthält die Metadaten des Produkts, so wie sie vom Paketierer bereitgestellt wurden. Die Datei liegt hier nur zu Informationszwecken, d.h. Änderungen an dieser Datei haben keinerlei Auswirkungen auf das System.

•<productid>.files
Diese Datei wird automatisch erzeugt und sollte nicht verändert werden.

•create_driver_links.py
show_drivers.py
download_driver_pack.py

extract_driver_pack.py
Dies sind Scripte zur Treiberintegration die im Kapitel 5.1.9 Vereinfachte Treiberintegration in die automatische Windowsinstallation auf Seite 91 näher erläutert werden.

5.1.8.3. Verzeichnis i386 / installfiles / winpe

Beachten Sie zu diesen Verzeichnissen die Informationen aus dem Installationshanduch.

•i386
Dieses Verzeichnis enthält den i386 Baum der Windows Installations-CD bei Windows Versionen 2000 bis XP (NT 5.x).
Es kann mehrere i386 Verzeichnisse geben (i386 , i386_en , i386_xxx). Welcher bei der Installation zur Verwendung kommt, wird über das Produktproperty 'i386_dir' gesteuert.

•installfiles
Enthält bei Windows Vista und größer (NT 6.x / 7) den Inhalt der Installations-CD.

•winpe
Enthält bei Windows Vista und größer (NT 6.x / 7) ein bootbares winpe Image.

5.1.8.4. Verzeichnis opsi / custom

Diese beiden Verzeichnisse enthalten Scripte und Konfigurationsdateien zur Steuerung der Betriebssysteminstallation. Während der Installation wirken diese Verzeichnisse zusammen, indem die Dateien aus custom Vorrang haben.

Das Verzeichnis opsi enthält Dateien, die mit Updates jederzeit überspielt werden können. Hier sollten also keine Änderungen vorgenommen werden. Für Anpassungen können Sie Änderungen im Verzeichnis custom vornehmen, welches bei Updates nicht überspielt wird.

Das Unterverzeichnis postinst.d enthält Scripte, welche nach der eigentlichen Installation des Betriebssystems gestartet werden um z.B. den opsi-preloginloader zu installieren. Die Scripte werden dabei in alphabetischer Reihenfolge abgearbeitet. Um die Reihenfolge zu verdeutlichen, fangen die Dateinamen mit einer zweistelligen Nummer an (10_dhcp.cmd). Wollen Sie hier Erweiterungen vornehmen, so können Sie

im Verzeichnis custom/posinst.d Scripte mit Nummern zwischen den vollen 10ern ablegen (13_myscript.cmd). Die vollen 10er sind für die Pflege durch opsi.org/uib reserviert.

5.1.8.5. Verzeichnis drivers

Dieses Verzeichnis dient der Treiberintegration und ist im folgenden Kapitel beschrieben.

5.1.9. Vereinfachte Treiberintegration in die automatische Windowsinstallation

Administriert man einen Pool von PCs, die Geräte besitzen, deren Treiber nicht in der Windows-Standardinstallation enthalten sind, so ist es meist sinnvoll, diese Treiber direkt in die Installation zu integrieren. Bei Netzwerkgeräten kann dies teilweise sogar unumgänglich sein, denn ein startendes Windows ohne Netzwerkkarte ist für den Administrator nicht ohne weiteres erreichbar.

opsi unterstützt Sie durch eine Automatisierung der Treibereinbindung und vereinfacht so die Bereitstellung der Treiber. Dabei müssen die Treiber nur in dem korrekten Verzeichnis abgelegt werden. Durch den Aufruf eines Scriptes werden dann die Treiberverzeichnisse durchsucht und ein Katalog erstellt, anhand dessen das Bootimage automatisch die richtigen Treiber erkennen und einbinden kann. Dabei können sowohl Standard-Treiber , USB-Treiber, HD-Audio-Treiber wie auch Treiber für Festplattencontroller (Textmode Treiber) abgelegt und automatisch eingebunden werden.

Damit die Treiber sofort bei der Windowsinstallation mit installiert werden, müssen Sie in einer bestimmten Form auf dem Server hinterlegt werden. Hierzu sind Treiberverzeichnisse geeignet die eine *.inf-Datei enthalten, die den Treiber für das Windows-Setupprogramm beschreibt. Irgendwelche in setup.exe, *.zip oder anders verpackten Treiber sind hier unbrauchbar. Mit dem Programm 'double driver' (http://www.boozet.org/dd.htm) können Sie von einem installierten Rechner die Treiber im geeigneten Format extrahieren.

Haben sie nur wenige unterschiedliche Hardware zu unterstützen, so können Sie die Treiber bei den Herstellern suchen.

Haben Sie sehr viel unterschiedliche Hardware, so können Sie komplette Packs für XP-Treiber bei http://driverpacks.net/ finden.

Die Driverpacks (!!! ca 2,5 GB !!!) von http://driverpacks.net/DriverPacks/overview.php herunterladen und auf dem Server abspeichern. Mit dem Befehl:
`/opt/pcbin/install/winxppro/extract_driver_pack.py` <pfad zu den komprimierten driverpacks>
werden die Treiber entpackt und in das Verzeichnis winxppro/drivers/drivers/D abgelegt.

Alternativ kann mit dem Befehl:
`/opt/pcbin/install/winxppro/download_driver_pack.py` versucht werden, die Treiber von einem Mirrorserver von driverpacks.net herunter zu laden und zu entpacken.

Struktur des Treiber Verzeichnisses und Ablage der Treiber:

```
/opt/
  └─pcbin/
     └─install/
        └─winxppro/
           └─drivers
              ├─classes/            (Links auf Treiber über Geräteklassen)
              ├─hdaudioids/         (Links auf HD-Audio Treiber)
              ├─pciids/             (Links auf Treiber über PCI-Kennung)
              ├─pci.ids             (PCI Datenbank)
              ├─usbids/             (Links auf Treiber über USB-Kennung)
              ├─usb.ids             (USB Datenbank)
              ├─vendors/            (Links auf Treiber über Hersteller)
              └─drivers             (Platz für allg. Treiber Packs)
                 ├─additional/      (Für Treiber die nicht erkannt werden)
                 ├─buildin/         (Daten aus dem i386 Baum)
                 ├─preferred/       (geprüfte Treiber)
                 └─mydriverpacks/   (Beispiel Treiber Pack)
```

Zusätzliche bzw. geprüfte Treiber gehören in jeweils eigene Verzeichnisse (Name und Tiefe der Verzeichnisstruktur egal) unterhalb des Verzeichnisses **`winxppro/drivers/drivers/preferred`** .

Danach bzw. nach jeder Änderung im **`drivers/drivers`** Verzeichnis rufen Sie im winxppro Verzeichnis das Script **`create_driver_links.py`** auf. Dieses durchsucht die Verzeichnisse unterhalb von **`drivers/drivers`** und erzeugt eine Reihe von Links anhand deren die Zuordnung der Treiber zu bestimmter Hardware (PCI-IDs, USB-IDs, HD-Audio-IDs) zu erkennen ist. Die Treiber aus dem **`preferred`** Verzeichnis werden

von dem Script bevorzugt verwendet. Das winxxpro.py Script des Bootimages untersucht die Hardware des zu installierenden Computers und identifiziert die notwendigen Treiber. Diese werden dann auf die Platte kopiert und die unattend.txt entsprechend gepatcht. Das Script **create_driver_links.py** durchsucht auch einmalig den i386 Baum und extrahiert die Inf-Dateien der von Windows mitgelieferten Treiber nach windows_builtin. Sollten Sie am i386-Baum eine Änderung vornehmen (z.B. durch das Einspielen eines Servicepacks) so löschen Sie dieses Verzeichnis und führen **create_driver_links.py** erneut aus.

Liegt zu einem Client eine Hardware-Inventarisierung vor, so kann über den Befehl:
winxppro/show_drivers.py <clientname>
ausgegeben werden, welche Treiber das Bootimage via PCI-IDs und USB-IDs zur Installation auswählen würde und zu welcher Hardware noch kein Treiber bereit steht.

Zusätzliche Treiber, die unabhängig von ihrer Zuordnung bzw. Erkennung über die PCI- oder USB-IDs installiert werden sollen, gehören in jeweils eigene Verzeichnisse (Name und Tiefe der Verzeichnisstruktur egal) unterhalb des Verzeichnisses **winxppro/drivers/drivers/additional** . Über das Produkt-Property 'additional' von winxppro können sie einen oder mehrere Pfade von Treiberverzeichnissen innerhalb von additional einem Client zu ordnen. Im Produkt-Property 'additional-drivers' angegebene Verzeichnisse werden rekursiv durchsucht und alle enthaltenen Treiber eingebunden. Dabei wird auch symbolischen Links gefolgt. Dies können Sie nutzen, um für bestimmte Rechner-Typen ein Verzeichnis zu erstellen (z.B. dell-optiplex-815).

Beispiel einer show_drivers.py Ausgabe:

```
./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
```

5.2. Ntfs-images (write + restore)

Mit den Produkten ntfs-write-image und ntfs-restore-image können Sie Abbilder von Partitionen sichern bzw widerherstellen. Ziel bzw. Quelle der Imagedatei müssen auf dem opsi-depot-server liegen und werden per ssh (user pcpatch) erreicht und im Produktproperty angegeben.

Entsprechende Produkte zum Sichern und Wiederherstellen von NTFS-Partitionen gibt es auch auf der opsi-client-boot-cd und sind in einem gesonderten Manual beschrieben.

5.3. Memtest

Das Produkt memtest dient dazu einen Memory-Test des Cliens durchzuführen.

5.4. hwinvent

Das Produkt hwinvent dient dazu eine Hardwareinventariserung des Clients durchzuführen.

5.5. wipedisk

Das Produkt wipedisk überschreibt überschreibt die gesamte Festplatte (partion=0) oder einzelne Partitionen mit unterschiedlichen Mustern. Die Anzahl der Schreibvorgänge wird über das product property 'iterations' gesteuert (1-25).

6. opsi-server

6.1. Überblick

Der opsi-server ist eine speziell konfigurierte Serverinstallation auf der Basis von GNU/ Debian Linux. Er dient als Basis für die Module Softwareverteilung und Betriebssysteminstallation.

Für die Softwareverteilung stellt er abgesicherte Fileshares bereit, in denen Konfigurationsdateien und Softwarepakete (Softwaredepots) vor unbefugten Zugriffen geschützt sind. Dazu werden die notwendigen Passwörter an die Clients verschlüsselt übertragen, so dass nur die Programme der automatischen Softwareinstallation und der Systemverwalter Zugriff auf diese Shares erhalten können.

Zentraler Dienst des opsi-servers ist seit opsi V3 die Bereitstellung eines Webservice zur Konfiguration von opsi und zur Abstraktion von der Datenhaltung.

Eine weitere wesentlich Funktion des opsi-servers ist die Bereitstellung der Dienste für die automatische Betriebssysteminstallation. Hierzu gehören die Dienste:

- dhcp für die Verwaltung von IP-Nummern,
- tftp für die Übertragung von Bootimages und Konfigurationsinformationen.

Weiterhin werden interaktive und skriptbare Werkzeuge zur Verwaltung der Konfigurationsdateien sowie die Bootimages selbst bereitgestellt.

Da der opsi-server aus Gründen der Sicherheit, der Stabilität und des Ressourcenverbrauchs auf das Notwendige beschränkt ist, sind die Hardwareanforderungen sehr gering. Soll der opsi-server nicht auf einer eigenen Hardware laufen, so ist auch der Betrieb in einer virtuellen PC-Instanz, wie z.B. von vmware® (http://www.vmware.com) möglich.

6.2. Installation und Inbetriebnahme

Die Installation und Inbetriebnahme eines opsi-servers ist in dem gesonderten Handbuch: 'opsi-server' Installation ausführlich erläutert.

6.3. Zugriff auf die grafische Benutzeroberfläche des opsi-servers über VNC

Der opsi-server verfügt über keinen X-Server für eine bestimmte Hardware. Die Konsole des opsi-servers ist daher rein textbasiert. Als X-Server wird der (tight)vnc-Server eingesetzt. Zugriff auf die X-Oberfläche des opsi-servers erhalten Sie somit über einen vncviewer von jedem Rechner in Ihrem Netz. Dieser Aufbau des depotservers erlaubt einen Einsatz ohne hardwarespezifische Anpassungen und damit ein großes Maß an Standardisierung, was sowohl die Stabilität erhöht als auch die Wartung vereinfacht.

Nach dem Starten des opsi-servers wird ein vncserver für jeden Administrationsuser, der in der Datei /etc/vncuser eingetragen ist, gestartet. Sollte aus irgendwelchen Gründen der entsprechende vncserver nicht laufen, können Sie diesen an der Kommandozeile mit **`vncserver`** starten.

Einrichten des VNC-Servers:

Die Datei /etc/vncusers dient als Steuerdatei für VNC-Server. Hier kann ein Mal je user ein „default“ VNC-Server eingetragen werden. Wahlweise startet dieser VNC-Server dann automatisch beim Booten des opsi-servers.

```
#Parameterdatei zum Start der Vncserver
#
#Beispiele:
#
#benutzername:displaynr:aufloesung:start_at_boot:localhost:
#test2:45:800x600:start_at_boot:localhost:
#test3:46:800x600::: # dies waere ein Minimaleintrag
#
#start_at_boot und localhost koennen also weggelassen werden,
#Doppelpunkte hingegen nicht!
#Doppelt auftretende Display-Nummern sind zu vermeiden
root:49:1260x960:start_at_boot::
```

Die Datei ist ähnlich der /etc/passwd aufgebaut, d.h. die Anzahl der Doppelpunkte ist wichtig, da hiermit die Anzahl der Felder vorgegeben ist.

1. Feld: username

2. Feld: Entspricht der „Server-Nummer“ (Port auf dem der VNC-Server lauscht)(1-99 zulässig), wird eigentlich Displaynummer genannt
3. Feld: gewünschte Auflösung, bei 1024x786er Bildschirm auf dem PC ist z.B. 1050x700 eine gute Wahl
4. Feld: start_at_boot, Nur wenn dieser Text dort steht, wird beim Booten des opsi-servers dieser VNC-Server automatisch gestartet, normalerweise nicht notwendig.
5. Feld: localhost, der VNC-Server lässt nur Verbindungen von localhost, nicht aus dem Netz zu, kann für ssh Tunnelung benutzt werden.

Ist man als user auf dem opsi-server eingeloggt, kann durch Aufruf von **`vncserver`** der user spezifische vncserver mit den gewünschten Parametern gestartet werden.

Beim erstem Aufruf des vncservers, wird nach einem Passwort gefragt, welches dann in ~USER/.vnc/passwd abgelegt wird und für alle VNC-Server dieses Benutzers gilt.

Beenden des VNC Servers:

```
vncserver -kill :<DisplayNummer>
```

Was ist VNC ?

VNC ist Opensource Software und wird unter der GNU Public License verteilt.
VNC (Virtuel Network Computing) - das ist eine (fast) betriebssystemunabhängige Client/Server-Anwendung, die es ermöglicht, die graphische Oberfläche eines ausgewählten Rechners (=Server) auf den eigenen Desktop (=Client) zu holen und mit diesem Rechner zu arbeiten.
VNC besteht aus einer Server- und aus einer Client-Applikation, die für verschiedene Zwecke unterschiedlich konfiguriert werden können.
Der Server ist bei VNC der Rechner, auf welchem über das Netzwerk gearbeitet werden soll! Der Client ist der Rechner, auf dem direkt gearbeitet wird.
Die Server-Applikation übermittelt den eigenen Bildschirminhalt zur Client-Applikation und bietet eine Schnittstelle für die Tastatur- sowie Mauseingaben der Client-Applikation. Die Server-Applikation muss aus Sicherheitsgründen mit einem Passwort konfiguriert werden, das vom Client für eine Verbindung eingegeben werden muss.

Webadressen von vnc : http://www.realvnc.com/
und tightvnc: http://www.tightvnc.com/

6.4. Bereitstellung eines Shares für Softwarepakete und Konfigurationsdateien

6.4.1. Samba Konfiguration

Um den Client-PCs Zugriff auf Konfigurationsinformationen und Softwarepakete zu ermöglichen, stellt der opsi-server Shares bereit, die von den Clients als Netzlaufwerke gemountet werden können. Für die Windows-Clients wird dazu die Software SAMBA in der Version 3.x eingesetzt.

Die Konfigurationsdateien von SAMBA liegen auf dem opsi-server unter /etc/samba. In der Datei /etc/samba/smb.conf sind die allgemeinen Einstellungen festgelegt. Die Einstellungen zu den Shares liegen auch dort oder in der Datei /etc/samba/share.conf. In dieser Datei wird festgelegt, welche Verzeichnisse für die Funktionen Softwaredepot, Hilfsprogramme und Konfiguration freigegeben werden und ob dies auf einem oder bis zu drei Shares geschieht. In der Voreinstellung liegen alle drei Funktionen auf einem Share. Dieser gibt das Verzeichnis /opt/pcbin als Share opt_pcbin frei. Änderungen gegenüber diesen Voreinstellungen müssen sowohl in der Datei /etc/samba/share.conf als auch in der globalen Netzwerkkonfiguration von opsi eingetragen werden. Nach einer Änderung der Samba-Konfigurationsdateien ist ein reload der Samba-Software notwendig (/etc/init.d/samba reload).

Beispiel für share.conf:

```
[opt_pcbin]
 available = yes
 comment = opsi depot share
 path = /opt/pcbin
 oplocks = no
 level2 oplocks = no
 writeable = yes
 invalid users = root

[opsi_config]
 available = yes
 comment = opsi config share
 path = /var/lib/opsi/config
 writeable = yes
 invalid users = root

[opsi_workbench]
 available = yes
 comment = opsi workbench share
 path = /home/opsiproducts
 writeable = yes
 invalid users = root
```

Im Prinzip bietet die Software SAMBA 3.x die Möglichkeit, den opsi-server zu einem vollwertigen File- und Printserver auszubauen. Auch hierzu bietet Ihnen die Firma uib umfangreiche Supportmöglichkeiten an.

6.4.2. Notwendige System-User und Gruppen

6.4.2.1. User opsiconfd

Dies ist der user unter dem der opsiconfd Deamon läuft.

6.4.2.2. User pcpatch

Der Zugriff auf die Konfigurationsdaten und das Softwaredepot sollte vor beliebigen Zugriffen geschützt sein. Einen Zugriff benötigen die Systemadministratoren sowie die Installationssoftware auf dem Client (opsi-PreLoginLoader). Dies ist zum Beispiel eine Voraussetzung dafür, dass der Systemadministrator seiner Verantwortung für die Einhaltung der Lizenzbestimmungen der zu verteilenden Software gerecht werden kann.

Um dies zu ermöglichen, gibt es einen System-User pcpatch mit der user-ID 992. Dieser User hat per Voreinstellung das Heimatverzeichnis /opt/pcbin/pcpatch und das Passwort 'Umwelt'. Ändern Sie das Passwort mit **`opsi-admin -d task setPcpatchPassword`**. Dieser Account wird vom opsi-PreLoginLoader verwendet, um auf die Shares zuzugreifen.

6.4.2.3. Gruppe pcpatch

Neben dem User pcpatch gibt es noch die Gruppe pcpatch. Die meisten Dateien sind sowohl für den User als auch für die Gruppe im Vollzugriff. Die Systemadministratoren des opsi-servers sollten daher Mitglieder der Gruppe pcpatch sein, damit Sie vom Client PC aus schreibend auf die Konfigurationsdaten zugreifen können.

Mit **`addgroup`** <username> **`pcpatch`** nehmen Sie <username> in die Gruppe auf.

6.4.2.4. Gruppe opsiadmin

Seit opsi V3 gibt es die Gruppe opsiadmin.

Die Mitglieder dieser Gruppe können sich gegenüber dem opsi-webservice Authentifizieren und damit z.B. mit dem opsi-configed arbeiten. Daher sollten alle Mitarbeiter die mit opsi arbeiten Mitglied dieser Gruppe sein.

Mit **`addgroup`** <username> **`opsiadmin`** nehmen Sie <username> in die Gruppe auf.

6.4.3. Bereich: Depotshare mit Softwarepaketen (opt_pcbin)

Auf dem depot-Share liegen die für die Installation durch das Programm opsi Winst vorbereiteten Softwarepakete. In der Voreinstellung liegt dieses Verzeichniss auf dem opsi-server unter /opt/pcbin/install. Unterhalb von diesem Verzeichnis findet sich für jedes Softwarepaket ein Verzeichnis mit dem Namen des Softwarepakets. Wiederum unterhalb dieses Verzeichnisses liegen dann die Installationsskripte und -dateien.

6.4.4. Bereich: Arbeitsverzeichnis zum Pakethandling (opsi_workbench)

Unter /home/opsiproducts ist der Bereich um Pakete zu erstellen und in dem Pakete vor der Installation mit opsi-package-manager abgelegt werden sollen.

Dieses Verzeichnis ist als share opsi_workbench freigegeben.

6.4.5. Bereich: Konfigurationsdateien File31-Backend (opsi_config)

Unter /var/lib/opsi liegen die Konfigurationsdateien des File31 Backends.

Dieses Verzeichnis ist als share opsi_config freigegeben.

Achtung: wenn Sie über diesen Share auf den Dateien arbeiten, verwenden Sie keine Editoren die das Dateiformat (Unix/DOS) verändern und entfernen Sie Sicherungsdateien wie z.B. *.bak.

7. opsi-server mit mehreren Depots

7.1. Support

Die hier beschriebenen Funktionen sind komplex und werden durch uib nur im Rahmen eines Professional Supportvertrages unterstützt. Wir empfehlen Ihnen, diese Funktionen nicht ohne einen entsprechenden Supportvertrag einzusetzen.

7.2. Konzept

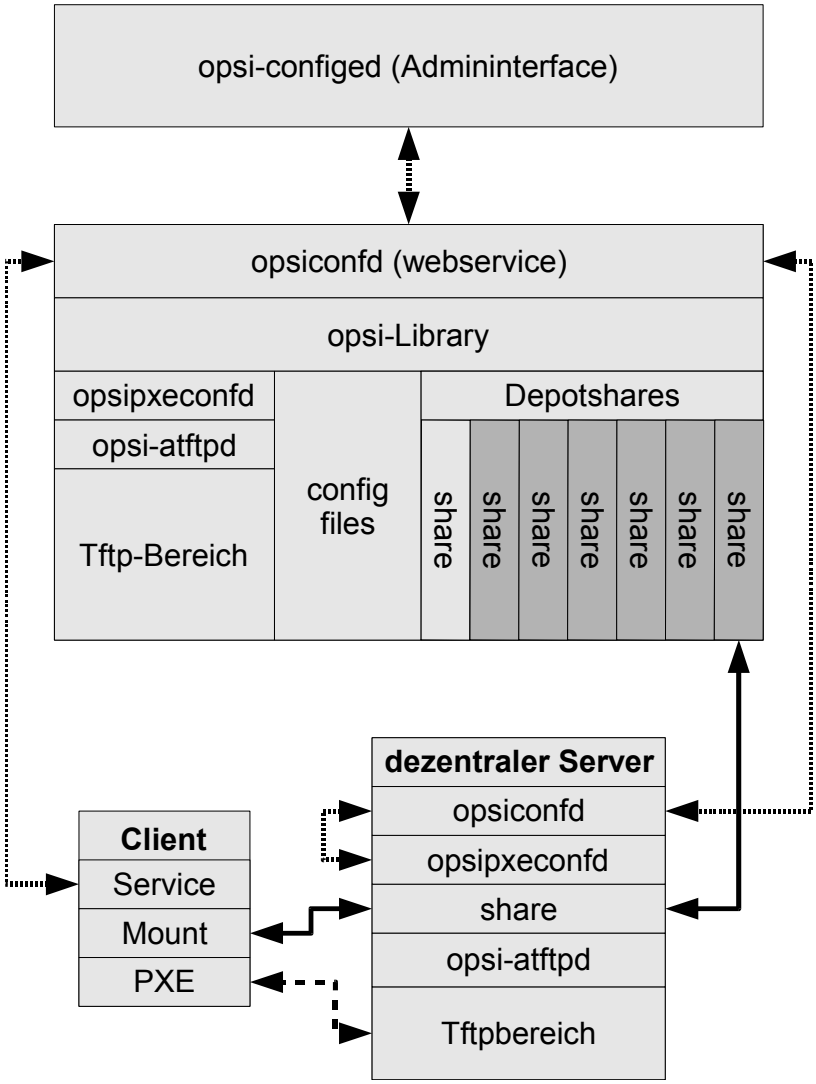
Die Unterstützung von mehreren Depots in opsi hat folgende Merkmale:

- Zentrale Speicherung und Administration der Konfigurationsdaten
- Dezentrale Bereitstellung der Softwaredepots
- Automatisierte Verteilung der installierten Softwarepakete auf die dezentralen Depots
- Verwaltung der Clients standortübergreifend in einem Administrationsinterface

Zur Umsetzung wurde folgendes Konzept verwirklicht:

- Die Konfigurationsdaten für alle Clients werden auf einem opsi-server (config-server) gehalten.
- Alle Clients verbinden sich über den opsi-Webservice mit dem config-server und erhalten von dort ihre Konfigurationsinformationen.
- Die Softwaredepots liegen auf dezentralen depot-servern und werden dem zentralen config-server als Netzwerkmounts zur Installation von Paketen zur Verfügung gestellt.
- Die Funktionalität zum Start von Bootimages mittels PXE wird ebenfalls auf dem dezentralen depot-server installiert. Diese wird aber zentral gesteuert.

7. opsi-server mit mehreren Depots



Schema: opsi mit dezentralen Depotshares

- opsi-packet-manager: Programm zur Unterstützung von mehreren Depotshares beim Installieren und Deinstallieren von opsi-Paketen.
- Transport der opsi-Pakete via webdav auf die depot-server und Installation durch den opsiconfd via webservice-call
- Unterstützung von mehreren Depotshares im Administrationswerkzeug opsi-configed.

- Automatisierte Erkennung von Inkonsistenzen zwischen dem Master-Depotshare und anderen Depotshares anhand der hinterlegten opsi-controlfiles.
- Ermöglichung der Selektion einzelner oder mehrerer Depotshares zur Auswahl der Clients im opsi-configed.
- Unterbinden der gemeinsamen Bearbeitung von Clients, die an Depotshares hängen und zueinander inkonsistent sind.
- Zuordnung der Clients zu Depotshares über den opsi-configed, Umzug von Clients.
- Konfigurations- und Verbindungsdaten der einzelnen Depotshares über den opsi-configed editierbar machen.

Das folgende Schema gibt einen detaillierteren Überblick über die Kommunikation zwischen den Komponenten:

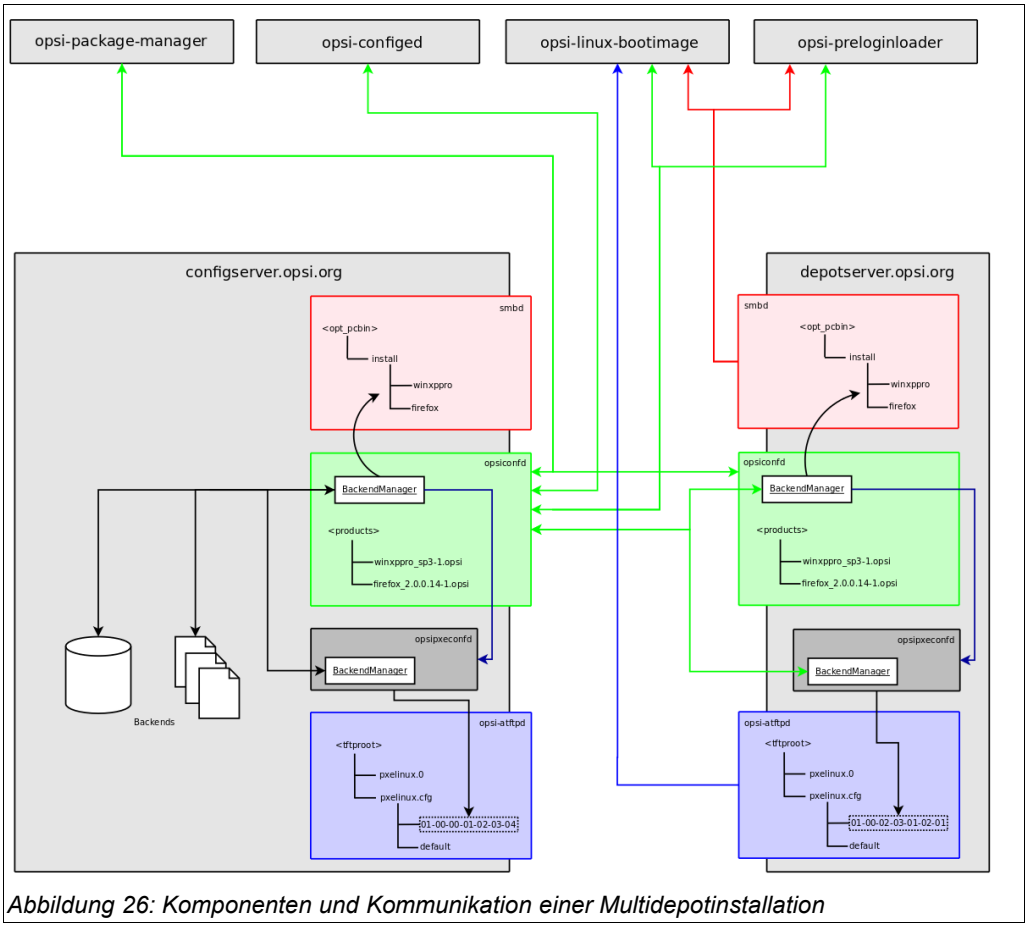


Abbildung 26: Komponenten und Kommunikation einer Multidepotinstallation

7.3. Erstellung und Konfiguration eines (slave) depot-servers

Zur Erstellung eines externen depot-servers wird zunächst ein normaler opsi-server aufgesetzt. Dieser wird mit Hilfe des scriptes /usr/share/opsi/register-depot.py zum externen depot-server konfiguriert. Da hierbei nicht nur der depot-server konfiguriert wird sondern dieser auch noch per Webservice dem zentralen config-server bekannt gemacht wird, müssen username und password eines Mitgliedes der Gruppe opsiadmin eingegeben werden.

Beispiel:
vmix12.uib.local wird als depot-server für den config-server bonifax.uib.local eingerichtet:

```
vmix12:~# /usr/share/opsi/register-depot.py

*******************************************************************************
***
*         This tool will register the current server as depotserver.
*
* The config file /etc/opsi/backendManager.d/15_jsonrpc.conf will be
recreated. *
*                  =>>> Press <CTRL> + <C> to abort <<<=
*
*******************************************************************************
***

 Config server [localhost]: bonifax.uib.local
 Account name to use for login [root]: oertel
 Account password [password]:
Connecting to host 'bonifax.uib.local' as user 'oertel'
 The subnet this depotserver is resonsible for [192.168.4.0/24]:
 Description for this depotserver [Depotserver vmix12]:
 Additional notes for this depotserver [Notes for vmix12]:

Creating depot 'vmix12.uib.local'
Requesting base-url '/rpc', query '{"params":
["vmix12","uib.local","file:///opt/pcbin/install","smb://vmix12/opst_pcbin/ins
tall","file:///var/lib/opsi/products","webdavs://vmix12.uib.local:4447/product
s","192.168.4.0/24","Depotserver vmix12","Notes for
vmix12"],"id":1,"method":"createDepot"}' failed:
Trying to reconnect...
Testing connection and setting pcpatch password
Connection / credentials ok, pcpatch password set!
Creating jsonrpc backend config file
/etc/opsi/backendManager.d/15_jsonrpc.conf
Patching config file /etc/opsi/backendManager.d/30_vars.conf
```

7.4. Paketmanagement auf mehreren depots

siehe auch Kapitel 3.4 Werkzeug opsi-package-manager: opsi-Pakete (de-) installieren Seite 25

Zur Verwaltung der Pakete auf mehreren Depots kennt der opsi-packet-manager die Optionen -d bzw. --depots mit denen die Depots angegeben werden können auf denen ein Paket installiert bzw. deinstalliert werden soll. Mit dem Schlüsselwort 'ALL' kann auf alle bekannten Depots verwiesen werden. Bei einer Installation mit der Option -d wird das Paket zunächst in das Verzeichnis /var/lib/opsi/products des depot-servers hochgeladen und dann von dort aus installiert.

Wird -d nicht angegeben, so wird nur das lokale Depot behandelt und das Paket ohne upload nach /var/lib/opsi/products installiert.

Beispiel:
Installiere das Paket softprod_1.0-5.opsi auf allen Depots:

```
opsi-package-manager -d ALL -i softprod_1.0-5.opsi
Processing upload of 'softprod_1.0-5.opsi' to depot 'bonifax.uib.local'
Processing upload of 'softprod_1.0-5.opsi' to depot 'vmix13.uib.local'
Processing upload of 'softprod_1.0-5.opsi' to depot 'vmix12.uib.local'
Overwriting destination 'softprod_1.0-5.opsi' on depot 'bonifax.uib.local'
Starting upload of 'softprod_1.0-5.opsi' to depot 'bonifax.uib.local'
   100.00%        3 KB       0 KB/s  00:00 ETAs - softprod_1.0-5.opsi >>
bonifax.uib.local
Upload of 'softprod_1.0-5.opsi' to depot 'bonifax.uib.local' done
Installing package 'softprod_1.0-5.opsi' on depot 'bonifax.uib.local'
Overwriting destination 'softprod_1.0-5.opsi' on depot 'vmix13.uib.local'
Starting upload of 'softprod_1.0-5.opsi' to depot 'vmix13.uib.local'
Overwriting destination 'softprod_1.0-5.opsi' on depot 'vmix12.uib.local'
Starting upload of 'softprod_1.0-5.opsi' to depot 'vmix12.uib.local'
   100.00%        3 KB       3 KB/s  00:00 ETAs - softprod_1.0-5.opsi >>
vmix13.uib.local
   100.00%        3 KB       3 KB/s  00:00 ETAs - softprod_1.0-5.opsi >>
vmix12.uib.local
Upload of 'softprod_1.0-5.opsi' to depot 'vmix12.uib.local' done
Installing package 'softprod_1.0-5.opsi' on depot 'vmix12.uib.local'
Upload of 'softprod_1.0-5.opsi' to depot 'vmix13.uib.local' done
Installing package 'softprod_1.0-5.opsi' on depot 'vmix13.uib.local'
Installation of package '/var/lib/opsi/products/softprod_1.0-5.opsi' on depot
'bonifax.uib.local' finished
Installation of package '/var/lib/opsi/products/softprod_1.0-5.opsi' on depot
'vmix13.uib.local' finished
Installation of package '/var/lib/opsi/products/softprod_1.0-5.opsi' on depot
'vmix12.uib.local' finished
```

In diesem Beispiel sind drei Depots bekannt (bonifax.uib.local, vmix12.uib.local, vmix13.uib.local). Zunächst wird gleichzeitig auf alle Depots das Paket kopiert (upload)

und danach das Paket auf dem Depot installiert. Das auf dem opsi-server lokal vorhandene Depot (bonifax.uib.local) wird dabei genauso behandelt wie die entfernten Depots.

Um die Differenzen zwischen Depots angezeigt zu bekommen dient die Option -D (bzw. --differences)

Beispiel:
Unterschiede zwischen den bekannten Depots bezüglich des Produktes mshotfix

```
opsi-package-manager -D -d ALL mshotfix
mshotfix
    vmix12.uib.local :  200804-1
    vmix13.uib.local :  200804-1
    bonifax.uib.local:  200805-2
```

7.5. Konfigurationsdateien

siehe:

Kapitel 12.3.1.5 Konfgurationsdateien in /var/lib/opsi/config/depots/<depotid> Seite 131Support

Die hier beschriebenen Funktionen sind komplex und werden durch uib nur im Rahmen eines Professional Supportvertrages unterstützt. Wir empfehlen Ihnen, diese Funktionen nicht ohne einen entsprechenden Supportvertrag einzusetzen.

8. DHCP und Namensauflösung (DNS)

(in Erstellung)

####################################

9. Datenhaltung von opsi (Backends)

9.1. File-Backends

In den File-Backends liegen die Konfigurationsinformationen in Ini-File artigen Textdateien auf dem Server.

9.1.1. File3.1-Backend (opsi 3.1)

Wesentliche Merkmale des Backends 'File' :

- Aktuelles Defaultbackend von opsi
- Linux Standard Base konform
- Nicht Rückwärtskompatibel zu opsi 2.x/3.0
- Alle Funktionen von opsi sind mit diesem Backend verfügbar
- Funktioniert nur mit Clients die im 'Service'-Mode laufen, sprich auf Ihre Konfigurationen über den Service zugreifen.

Die Dateien dieses Backends liegen unter /var/lib/opsi.

Inhalt und Aufbau dieser Dateien ist im Kapitel 12 Wichtige Dateien des opsi-servers Seite 123 näher erläutert.

9.2. LDAP Backend

Das opsi-LDAP-Schema findet sich unter /etc/ldap/schemas.

Zur Aktivierung des LDAP-Backends muss ein funktionierender LDAP-Server verfügbar sein.

In der Konfigurationsdatei /etc/ldap/slapd.conf muss das opsi ldap-Schema eingebunden werden, dafür die Zeile

```
include         /etc/ldap/schema/opsi.schema
```

einfügen und den slapd neu starten.

Nun muss noch die Backend-Konfiguration von OPSI angepasst werden.

9.2.1. Das LDAP-Backend einbinden

Um das LDAP-Backend zu aktivieren muß in der /etc/opsi/backendmanager.conf folgender Eintrag geändert werden:

Einstellung für File-Backend:

```
self.backends[BACKEND_FILE] =           { 'load': True }
self.backends[BACKEND_LDAP] =           { 'load': False }
```

Einstellung für LDAP-Backend:

```
self.backends[BACKEND_FILE] =           { 'load': False }
self.backends[BACKEND_LDAP] =           { 'load': True }
```

9.2.2. Das LDAP-Backend konfigurieren

```
    self.backends[BACKEND_LDAP]['config'] = {
        "host":     "localhost",
        "bindDn":   "cn=admin,%s" % baseDn,
        "bindPw":   "password",
    }
```

9.2.3. Das LDAP-Backend den gewünschten Methoden zuordnen

```
self.defaultBackend = BACKEND_LDAP
self.clientManagingBackend = [ BACKEND_DHCPD, BACKEND_LDAP ]
self.pxebootconfBackend    = BACKEND_OPSIPXECONFD
self.passwordBackend       = BACKEND_FILE31
self.pckeyBackend          = BACKEND_FILE31
self.swinventBackend       = BACKEND_MYSQL
self.hwinventBackend       = BACKEND_MYSQL
self.loggingBackend        = BACKEND_FILE31
```

In diesem Beispiel werden die PC-Keys und pcpatch-Passwörter weiterhin im BACKEND_FILE31 verwaltet.

Nun muss der opsiconfd neu gestartet werden:

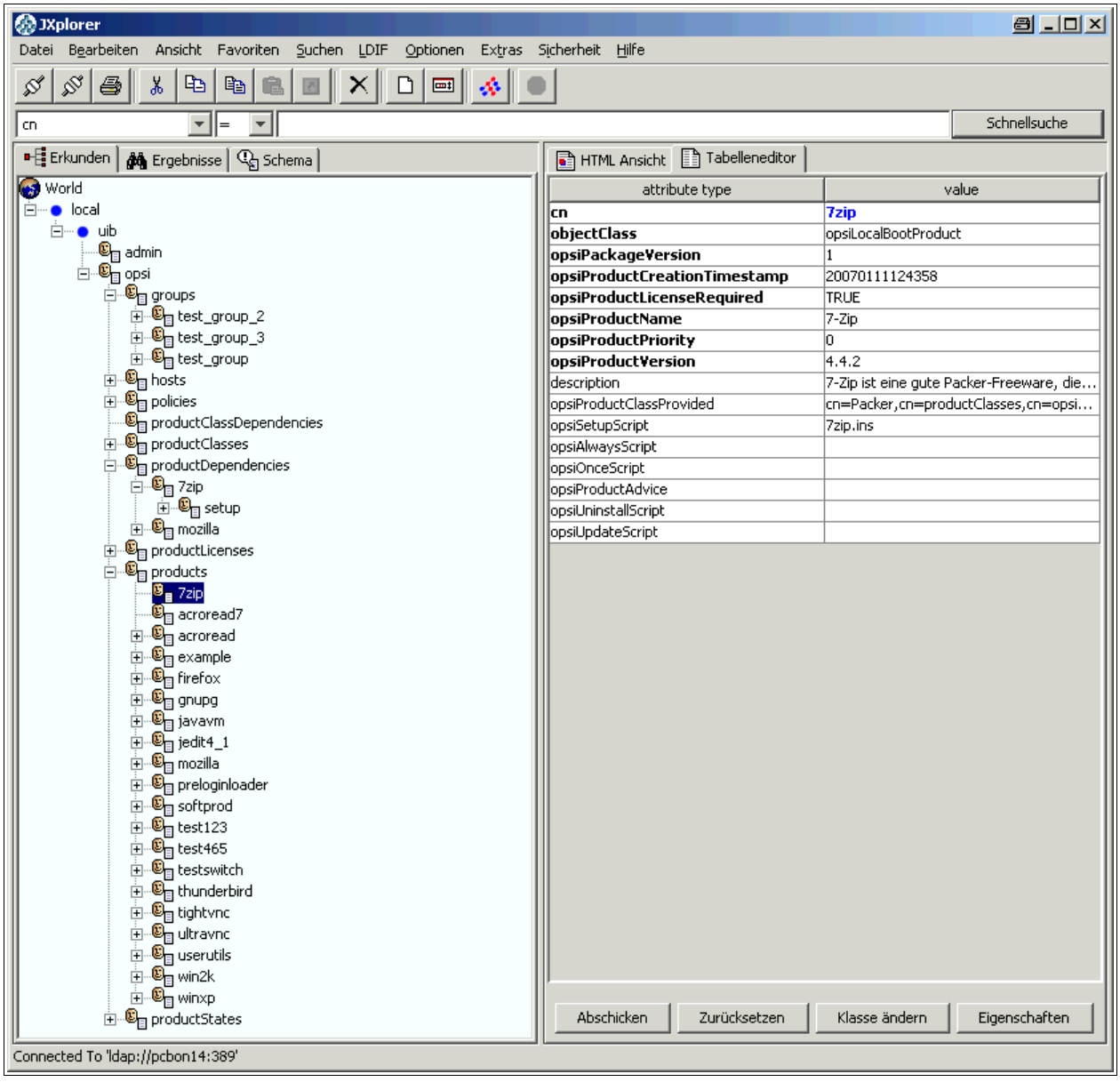
```
/etc/init.d/opsiconfd restart
```

Um die Basis-Struktur im LDAP anzulegen einfach den Befehl:

```
opsi-admin -d method createOpsiBase
```

ausführen.

Unterhalb der LDAP-Basis liegt eine organizationalRole cn=opsi (z.B. cn=opsi, dc=uib, dc=local). Unterhalb von opsi finden Sie die komplette LDAP-Datenhaltung. Diese läßt sich mit einem grafischen Frontend wie dem Jxplorer (z.B. aus den opsi-adminutils) leicht erkunden.



9.3. MySQL-Backend für Inventarisierungsdaten

9.3.1. Übersicht und Datenstruktur

Die Daten der Hardware- und Softwareinventarisierung werden per default über das opsi File31-Backend in Textdateien abgelegt. Diese Form der Ablage ist für freie Abfragen und Reports weniger geeignet. Hierfür bietet sich die Ablage der Daten in einer SQL-Datenbank an.

Wesentliche Merkmale des Backends 'MySQL' :

- Nur für die Inventarisierungsdaten
- Optional (nicht das Default Backend)
- Fein granulierte Datenstruktur zur Datenhaltung und zusätzlich vereinfachtes Datenmodell für Abfragen.
- Eine Historyfunktion, welche Änderungen an den Inventarisierungsdaten protokolliert.

Seit opsi 3.3 gibt es ein MySQL-Backend für die Inventarisierungsdaten. Bedingt durch die sehr unterschiedliche Natur der zu inventarisierenden Komponenten ist die Datenstruktur in etwa wie folgt aufgebaut:

Eine Tabelle host beschreibt alle bekannten Clients und stellt eine eindeutige host_id bereit. Für jeden Device-Typ gibt es zwei Tabellen: HARDWARE_DEVICE_ .... beschreibt das Device z.B. Netzwerkkartentyp mit PCI-Kennung sowie HARDWARE_CONFIG... Konfiguration der konkreten Netzwerkkarte z.B. MAC-Adresse. Die beiden Tabellen sind über das Feld hardware_id miteinander verbunden. Daraus ergibt sich folgende Liste von Tabellen:

```
HARDWARE_CONFIG_1394_CONTROLLER
HARDWARE_CONFIG_AUDIO_CONTROLLER
HARDWARE_CONFIG_BASE_BOARD
HARDWARE_CONFIG_BIOS
HARDWARE_CONFIG_CACHE_MEMORY
HARDWARE_CONFIG_COMPUTER_SYSTEM
HARDWARE_CONFIG_DISK_PARTITION
HARDWARE_CONFIG_FLOPPY_CONTROLLER
HARDWARE_CONFIG_FLOPPY_DRIVE
HARDWARE_CONFIG_HARDDISK_DRIVE
HARDWARE_CONFIG_IDE_CONTROLLER
```

```
HARDWARE_CONFIG_KEYBOARD
HARDWARE_CONFIG_MEMORY_BANK
HARDWARE_CONFIG_MEMORY_MODULE
HARDWARE_CONFIG_MONITOR
HARDWARE_CONFIG_NETWORK_CONTROLLER
HARDWARE_CONFIG_OPTICAL_DRIVE
HARDWARE_CONFIG_PCI_DEVICE
HARDWARE_CONFIG_PCMCIA_CONTROLLER
HARDWARE_CONFIG_POINTING_DEVICE
HARDWARE_CONFIG_PORT_CONNECTOR
HARDWARE_CONFIG_PRINTER
HARDWARE_CONFIG_PROCESSOR
HARDWARE_CONFIG_SCSI_CONTROLLER
HARDWARE_CONFIG_SYSTEM_SLOT
HARDWARE_CONFIG_TAPE_DRIVE
HARDWARE_CONFIG_USB_CONTROLLER
HARDWARE_CONFIG_VIDEO_CONTROLLER
HARDWARE_DEVICE_1394_CONTROLLER
HARDWARE_DEVICE_AUDIO_CONTROLLER
HARDWARE_DEVICE_BASE_BOARD
HARDWARE_DEVICE_BIOS
HARDWARE_DEVICE_CACHE_MEMORY
HARDWARE_DEVICE_COMPUTER_SYSTEM
HARDWARE_DEVICE_DISK_PARTITION
HARDWARE_DEVICE_FLOPPY_CONTROLLER
HARDWARE_DEVICE_FLOPPY_DRIVE
HARDWARE_DEVICE_HARDDISK_DRIVE
HARDWARE_DEVICE_IDE_CONTROLLER
HARDWARE_DEVICE_KEYBOARD
HARDWARE_DEVICE_MEMORY_BANK
HARDWARE_DEVICE_MEMORY_MODULE
HARDWARE_DEVICE_MONITOR
HARDWARE_DEVICE_NETWORK_CONTROLLER
HARDWARE_DEVICE_OPTICAL_DRIVE
HARDWARE_DEVICE_PCI_DEVICE
HARDWARE_DEVICE_PCMCIA_CONTROLLER
HARDWARE_DEVICE_POINTING_DEVICE
HARDWARE_DEVICE_PORT_CONNECTOR
HARDWARE_DEVICE_PRINTER
HARDWARE_DEVICE_PROCESSOR
HARDWARE_DEVICE_SCSI_CONTROLLER
HARDWARE_DEVICE_SYSTEM_SLOT
HARDWARE_DEVICE_TAPE_DRIVE
HARDWARE_DEVICE_USB_CONTROLLER
HARDWARE_DEVICE_VIDEO_CONTROLLER
HARDWARE_INFO
HOST
SOFTWARE
SOFTWARE_CONFIG
```

Da für viele Abfragen diese Datenstruktur etwas unhandlich ist, wird zusätzlich die Tabelle HARDWARE_INFO zur Verfügung gestellt. Diese fasst Informationen zu unterschiedlichsten Devices in einer Tabelle zusammen und vereinfacht so Abfragen:

```
CREATE TABLE  `opsi`.`HARDWARE_INFO` (
  `config_id` int(11) NOT NULL,           //Verweis auf Device Configtabelle
  `host_id` int(11) NOT NULL,             //Verweis auf host-Tabelle
  `hardware_id` int(11) NOT NULL,         //Verweis auf Device Tabelle
  `hardware_class` varchar(50) NOT NULL,  //Device
  `audit_firstseen` timestamp NOT NULL,   //
  `audit_lastseen` timestamp NOT NULL,
  `audit_state` tinyint(4) NOT NULL,      //1=aktuell 0=nicht mehr aktuell
  `internalConnectorType` varchar(60) ,
  `verticalResolution` int(11),
  `totalPhysicalMemory` bigint(20),
  `family` varchar(50) ,
  `vendorId` varchar(4) ,
  `addressWidth` tinyint(4),
  `videoProcessor` varchar(20) ,
  `numberOfFunctionKeys` int(11),
  `maxDataWidth` tinyint(4),
  `memoryType` varchar(20) ,
  `maxSize` int(11),
  `tag` varchar(100) ,
  `voltage` double,
  `slots` tinyint(4),
  `screenWidth` int(11),
  `connectorType` varchar(60) ,
  `maxCapacity` bigint(20),
  `size` bigint(20),
  `formFactor` varchar(10) ,
  `driveLetter` varchar(2) ,
  `capacity` bigint(20),
  `socketDesignation` varchar(100) ,
  `externalConnectorType` varchar(60) ,
  `numberOfButtons` tinyint(4),
  `capabilities` varchar(200) ,
  `port` varchar(20) ,
  `dataWidth` tinyint(4),
  `horizontalResolution` int(11),
  `version` varchar(50) ,
  `maxClockSpeed` bigint(20),
  `location` varchar(50) ,
  `paperSizesSupported` varchar(200) ,
  `deviceType` varchar(10) ,
  `subsystemVendorId` varchar(4) ,
  `adapterRAM` bigint(20),
  `speed` int(11),
  `architecture` varchar(50) ,
  `status` varchar(20) ,
  `freeSpace` bigint(20),
  `product` varchar(100) ,
  `vendor` varchar(50) ,
  `description` varchar(100) ,
  `index` int(11),
  `systemType` varchar(50) ,
  `macAddress` varchar(20) ,
  `installedSize` int(11),
  `driverName` varchar(100) ,
  `subsystemDeviceId` varchar(4) ,
  `internalDesignator` varchar(60) ,
  `currentUsage` varchar(20) ,
```

```
  `extClock` int(11),
  `heads` int(11),
  `autoSense` varchar(20) ,
  `currentClockSpeed` bigint(20),
  `netConnectionStatus` varchar(20) ,
  `partitions` tinyint(4),
  `maxSpeed` int(11),
  `busId` varchar(60) ,
  `name` varchar(100) ,
  `sectors` bigint(20),
  `level` varchar(10) ,
  `serialNumber` varchar(50) ,
  `screenHeight` int(11),
  `startingOffset` bigint(20),
  `externalDesignator` varchar(60) ,
  `filesystem` varchar(50) ,
  `cylinders` int(11),
  `model` varchar(100) ,
  `revision` varchar(4) ,
  `deviceLocator` varchar(100) ,
  `adapterType` varchar(20) ,
  `deviceId` varchar(4) ,
  PRIMARY KEY  (`config_id`,`host_id`,`hardware_class`,`hardware_id`)
)
```

Die Zuordnung der Spaltennamen zu einzelnen Deviceklassen ergibt sich aus folgender Liste (/etc/opsi/hwaudit/locales/de_DE):

```
DEVICE_ID.deviceType = Gerätetyp
DEVICE_ID.vendorId = Hersteller-ID
DEVICE_ID.deviceId = Geräte-ID
DEVICE_ID.subsystemVendorId = Subsystem-Hersteller-ID
DEVICE_ID.subsystemDeviceId = Subsystem-Geräte-ID
DEVICE_ID.revision= Revision
BASIC_INFO.name = Name
BASIC_INFO.description = Beschreibung
HARDWARE_DEVICE.vendor = Hersteller
HARDWARE_DEVICE.model = Modell
HARDWARE_DEVICE.serialNumber = Seriennummmer
COMPUTER_SYSTEM = Computer
COMPUTER_SYSTEM.systemType = Typ
COMPUTER_SYSTEM.totalPhysicalMemory = Arbeitsspeicher
BASE_BOARD = Hauptplatine
BASE_BOARD.product = Produkt
BIOS = BIOS
BIOS.version = Version
SYSTEM_SLOT = System-Steckplatz
SYSTEM_SLOT.currentUsage = Verwendung
SYSTEM_SLOT.status = Status
SYSTEM_SLOT.maxDataWidth = Max. Busbreite
PORT_CONNECTOR = Port
PORT_CONNECTOR.connectorType = Attribute
PORT_CONNECTOR.internalDesignator = Interne Bezeichnung
PORT_CONNECTOR.internalConnectorType = Interner Typ
PORT_CONNECTOR.externalDesignator = Externe Bezeichnung
PORT_CONNECTOR.externalConnectorType = Externer Typ
PROCESSOR = Prozessor
PROCESSOR.architecture = Architektur
```

```
PROCESSOR.family = Familie
PROCESSOR.currentClockSpeed = Momentane Taktung
PROCESSOR.maxClockSpeed = Maximale Taktung
PROCESSOR.extClock = Externe Taktung
PROCESSOR.processorId = Prozessor-ID
PROCESSOR.addressWidth = Adress-Bits
PROCESSOR.socketDesignation = Zugehöriger Sockel
PROCESSOR.voltage = Spannung
MEMORY_BANK = Speicher-Bank
MEMORY_BANK.location = Position
MEMORY_BANK.maxCapacity = Maximale Kapazität
MEMORY_BANK.slots = Steckplätze
MEMORY_MODULE = Speicher-Modul
MEMORY_MODULE.deviceLocator = Zugehöriger Sockel
MEMORY_MODULE.capacity = Kapazität
MEMORY_MODULE.formFactor = Bauart
MEMORY_MODULE.speed = Taktung
MEMORY_MODULE.memoryType = Speichertyp
MEMORY_MODULE.dataWidth = Datenbreite
MEMORY_MODULE.tag = Bezeichnung
CACHE_MEMORY = Zwischenspeicher
CACHE_MEMORY.installedSize = Installierte Größe
CACHE_MEMORY.maxSize = Maximale Größe
CACHE_MEMORY.location = Position
CACHE_MEMORY.level = Level
PCI_DEVICE = PCI-Gerät
PCI_DEVICE.busId = Bus-ID
NETWORK_CONTROLLER = Netzwerkkarte
NETWORK_CONTROLLER.adapterType = Adapter-Typ
NETWORK_CONTROLLER.maxSpeed = Maximale Geschwindigkeit
NETWORK_CONTROLLER.macAddress = MAC-Adresse
NETWORK_CONTROLLER.netConnectionStatus = Verbindungsstatus
NETWORK_CONTROLLER.autoSense = auto-sense
AUDIO_CONTROLLER = Audiokarte
IDE_CONTROLLER = IDE-Controller
SCSI_CONTROLLER = SCSI-Controller
FLOPPY_CONTROLLER = Floppy-Controller
USB_CONTROLLER = USB-Controller
1394_CONTROLLER = 1394-Controller
PCMCIA_CONTROLLER = PCMCIA-Controller
VIDEO_CONTROLLER = Grafikkarte
VIDEO_CONTROLLER.videoProcessor = Video-Prozessor
VIDEO_CONTROLLER.adapterRAM = Video-Speicher
DRIVE.size = Größe
FLOPPY_DRIVE = Floppylaufwerk
TAPE_DRIVE = Bandlaufwerk
HARDDISK_DRIVE = Festplatte
HARDDISK_DRIVE.cylinders = Cylinder
HARDDISK_DRIVE.heads = Heads
HARDDISK_DRIVE.sectors = Sektoren
HARDDISK_DRIVE.partitions = Partitionen
DISK_PARTITION = Partition
DISK_PARTITION.size = Größe
DISK_PARTITION.startingOffset = Start-Offset
DISK_PARTITION.index = Index
DISK_PARTITION.filesystem = Dateisystem
DISK_PARTITION.freeSpace = Freier Speicher
DISK_PARTITION.driveLetter = Laufwerksbuchstabe
```

```
OPTICAL_DRIVE = Optisches Laufwerk
OPTICAL_DRIVE.driveLetter = Laufwerksbuchstabe
MONITOR = Monitor
MONITOR.screenHeight = Vertikale Auflösung
MONITOR.screenWidth = Horizontale Auflösung
KEYBOARD = Tastatur
KEYBOARD.numberOfFunctionKeys = Anzahl Funktionstasten
POINTING_DEVICE = Zeigegerät
POINTING_DEVICE.numberOfButtons = Anzahl der Tasten
PRINTER = Drucker
PRINTER.horizontalResolution = Vertikale Auflösung
PRINTER.verticalResolution = Horizontale Auflösung
PRINTER.capabilities = Fähigkeiten
PRINTER.paperSizesSupported = Unterstützte Papierformate
PRINTER.driverName = Name des Treibers
PRINTER.port = Anschluss
```

Beispiele für Abfragen:

Gesamte Hardwareliste geordnet nach Clients und Devices:

```
select
HOST.hostId,HARDWARE_INFO.*
from
HOST,HARDWARE_INFO
where
(HOST.host_id = HARDWARE_INFO.host_id)
ORDER BY
HOST.hostId,HARDWARE_INFO.hardware_class,HARDWARE_INFO.config_id
```

Gesamte Hardwareliste eines Clients geordnet nach Devices:

```
select
HOST.hostId,HARDWARE_INFO.*
from
HOST,HARDWARE_INFO
where
(HOST.host_id = HARDWARE_INFO.host_id)
and HOST.hostId = 'pcuwb03.uib.local'
ORDER BY
HOST.hostId,HARDWARE_INFO.hardware_class,HARDWARE_INFO.config_id
```

Liste aller Festplatten:

```
SELECT * FROM HARDWARE_DEVICE_HARDDISK_DRIVE D
LEFT OUTER JOIN HARDWARE_CONFIG_HARDDISK_DRIVE H ON
D.hardware_id=H.hardware_id ;
```

9.3.2. Initialisierung des MySQL-Backends

Wenn der mysql-server noch nicht installiert ist, muss dies zunächst erfolgen mit:

```
apt-get install mysql-server
```

Danach muss für der root Zugang von mysql ein Passwort gesetzt werden:

```
mysqladmin --user=root password linux123
```

mit dem Script /usr/share/opsi/init-opsi-mysql-db.py kann nun die Datenbank aufgebaut werden.

Eine Beispiel-Sitzung:

```
svmopside:/usr/share/opsi# ./init-opsi-mysql-db.py

*******************************************************************************
*
*  This tool will create an initial mysql database for use as opsi backend.
*
* The config file /etc/opsi/backendManager.d/21_mysql.conf will be recreated.
*
*                 =>>> Press <CTRL> + <C> to abort <<<=
*
*******************************************************************************
*

 Database host [localhost]:
 Database admin user [root]:
 Database admin password [password]:
 Opsi database name [opsi]:
 Opsi database user [opsi]:
 Opsi database password [opsi]:

Connecting to host 'localhost' as user 'root'
Creating database 'opsi' and user 'opsi'
Testing connection
Connection / credentials ok!
Creating mysql backend config file /etc/opsi/backendManager.d/21_mysql.conf
Creating opsi base
Got duplicate property 'name' of same type 'varchar' but different sizes: 100,
50
Using type varchar(100) for property 'name'
Got duplicate property 'name' of same type 'varchar' but different sizes: 100,
60
Using type varchar(100) for property 'name'
Got duplicate property 'name' of same type 'varchar' but different sizes: 100,
50
Using type varchar(100) for property 'name'
Got duplicate property 'location' of same type 'varchar' but different sizes:
50, 10
Using type varchar(50) for property 'location'
Got duplicate property 'name' of same type 'varchar' but different sizes: 100,
50
Using type varchar(100) for property 'name'
```

Bei den Abfragen können außer beim Passwort alle Vorgaben mit Enter bestätigt werden.

Die Warnungen am Endes des Scriptes sind zu ignorieren.

In der Datei /etc/opsi/backendManager.d/30_vars.conf ist festgelegt, welche Backendmanager von opsi zum Einsatz kommen.

Für die Hard- und Softwarewareinventarisierung ist das BACKEND_MYSQL einzutragen unabhängig davon, welches Backend ansonsten verwendet wird:

```
self.swinventBackend          = BACKEND_MYSQL
self.hwinventBackend          = BACKEND_MYSQL
```

Nach Anpassung der Backendkonfiguration muss der opsiconfd neu gestartet werden:

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

9.4. Konvertierung zwischen Backends

Der Befehl **opsi-convert** dient zum Konvertieren der opsi-Konfigurationsdaten zwischen verschiedenen Backends. Das Ziel oder Quelle kann auf verschieden Arten bestimmt werden:

- Backendnamen:
  Durch Angabe des Namen wird ein entsprechendes Backen auf dem aktuellen Server angegeben. So konvertiert **opsi-convert File File31** auf dem aktuellen Server vom File-Backend zum File31-Backend.

- Service-Adresse
  Durch Angaben von Serviceadressen kann ein Server z.B. auch Remote angesprochen werden. Die Service Adresse hat die Form
  https://<username>@<ipadresse>:4447/rpc
  Nach den Passwörtern wird gefragt. Beispiel:
  ```
  opsi-convert -s -l /tmp/log https://uib@192.168.2.162:4447/rpc \
  https://opsi@192.168.2.42:4447/rpc
  ```

- Konfigurationsverzeichnis
  Durch Angabe von Konfigurationsverzeichnissen für die entsprechende Backendmanagerkonfiguration können Quelle bzw. Ziel sehr detailliert beschrieben werden.

9.5. Bootdateien

Unter /tftpboot/linux finden sich die Bootdateien, die im Zusammenspiel mit den PXE-Bootproms benötigt werden.

9.6. Absicherung der Shares über verschlüsselte Passwörter

Die Installationssoftware opsi-PreLoginLoader greift auf die vom opsi-server zur Verfügung gestellten Shares zu um Software zu installieren sowie um Konfigurationsinformationen und Logdateien schreiben zu können. Hierzu wird der System-User pcpatch verwendet. Die Absicherung dieser Shares und damit der Authentifizierungsdaten des Users pcpatch sind wichtig für die:

- allgemeine Systemsicherheit und Datenintegrität
- Absicherung der potenziell lizenzpflichtigen Softwarepakete gegen missbräuchliche Nutzung

Um dem opsi-PreLoginLoader ein Zugriff auf die Authentifizierungsdaten zu ermöglichen, wird für jeden Client bei der Reinstallation durch den opsipxeconfd ein spezifischer Schlüssel erzeugt. Dieser Schlüssel wird zum einen in der Datei /etc/opsi/pckeys abgelegt und zum anderen dem PC bei der Reinstallation übergeben. Der übergebene Schlüssel wird im Rahmen der Betriebsysteminstallation in der Datei c:\opsi\cfg\locked.cfg so abgelegt, dass nur Administratoren Zugriff darauf haben. Ebenso hat auf dem opsi-server nur root, pcpatch und Mitglieder der Gruppe pcpatch Zugriff auf die Datei /etc/opsi/pckeys. Auf diese Weise verfügt jeder PC über einen Schlüssel, der nur dem PC und dem opsi-server bekannt ist und der gegenüber dem Zugriff durch normale Anwender geschützt ist. Mit diesem Schlüssel wird das aktuelle Passwort des system users pcpatch auf dem opsi-server verschlüsselt und im Backend abgelegt. Dieses verschlüsselte Passwort wird vom Client bei jedem Boot neu gelesen, so dass eine Änderung des pcpatch Passwortes jederzeit möglich ist und der Client auf verschlüsseltem Wege das veränderte Passwort erfährt.

10. Anpassen des preloginloaders an die Corporate Identity (CI)

Ab winst Version 4.6 lässt sich der winst über eine Manipulation der Grafik bg.png im Unterverzeichnis winstskin frei anpassen. Nach einer Veränderung der bg.png unter /opt/pcbin/install/opsi-winst/files/opsi-winst/winstskin setzen Sie mit dem Befehl touch /opt/pcbin/install/opsi-winst/files/opsi-winst/winst32.exe das Datum des Winst neu.

In der Folge wird beim nächsten Start des preloginloaders der opsi-winst mit der veränderten bg.png automatisch auf den Client geladen.

Zur Anpassung des preloginloaders selbst empfehlen wir die Verwendung des preloginloaders 3.4. Hier können die Dateien notifier\action.bmp und notifier\event.bmp an Ihre Bedürfnisse angepasst werden.

11. Übersicht: Ein PC bootet vom Netz

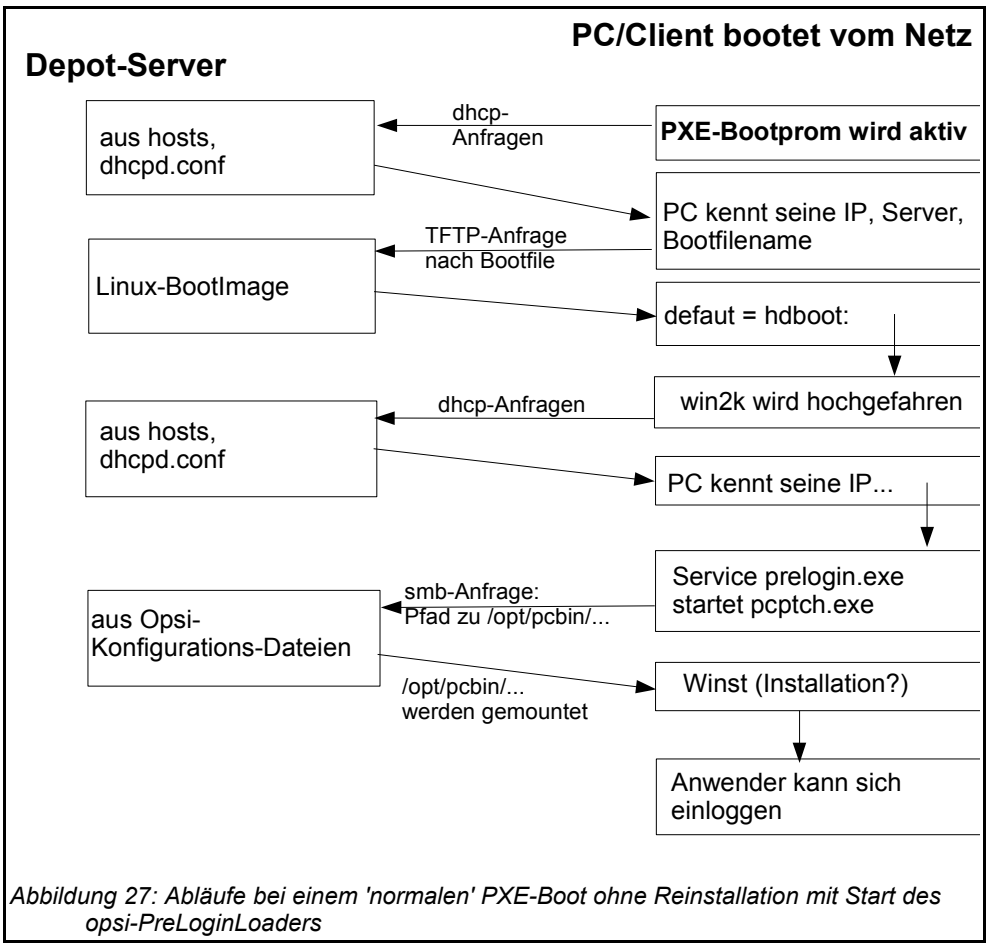


Abbildung 27: Abläufe bei einem 'normalen' PXE-Boot ohne Reinstallation mit Start des opsi-PreLoginLoaders

Auch hier wird zuerst die Firmware des Bootproms aktiv und sendet seine Anfrage ins Netz.

Bei einem Boot ohne Betriebssysteminstallation findet das Bootimage keine PC-spezifische Datei unter /tftpboot/linux/pxelinux.cfg, die default-Datei wird eingelesen und es erfolgt ein Boot von der lokalen Festplatte.

Nach diesem lokalen Boot muss das Betriebssystem die gleiche dhcp-Abfrage (nach IP und Netzinformationen) wie zuvor die Firmware des PXE durchführen.

12. Wichtige Dateien des opsi-servers

12.1. Allg. Konfigurationsdateien

12.1.1. Konfgurationsdateien in /etc

12.1.1.1. /etc/hosts

Hier können IP-Nummer und IP-Name der Clients eingetragen werden(zusätzliche Namen sind Aliase, ab dem Zeichen „#“ ist der Eintrag Kommentar).

opsi V3 hätte gerne den 'full qualified hostname' (also inclusive Domain) und dieser kann auch statt aus der /etc/hosts aus dem DNS kommen.

Beispiel:

```
192.168.2.104   laptop.uib.local laptopop
192.168.2.106  dplaptop.uib.local     # laptop maxdata 2004
192.168.2.153  schleppi.uib.local
192.168.2.178  test_pc1.uib.local # Test-PC PXE-bootprom
```

12.1.1.2. /etc/group

Hier müssen zwei Gruppen Angelegt sein: pcpatch und opsidamin. In der Gruppe pcpatch sollten alle user sein die mit Paketverwaltung zu tun haben. In der Gruppe opsiadmin müssen alle user sein die den opsiconfd-Webservice verwenden wollen z.B. über den opsi-Configed (oder das Applet).

12.1.1.3. /etc/opsi/pckeys

Hier sind die clientspezifischen Schlüssel des opsi-Reinstallationsmanagers sowie der Schlüssel des Servers selber abgelegt.

Beispiel:

```
schleppi:fdc2493ace4b372fd39dbba3fcd62182
laptop:c397c280fc2d3db81d39b4a4329b5f65
pcbon13:61149ef590469f765a1be6cfbacbf491
```

12.1.1.4. /etc/opsi/passwd

Hier sind die mit demSchlüssel des Servers verschlüsselten Passwörter (z.B. für pcpatch) abgelegt.

12.1.1.5. /etc/opsi/backendManager.conf

Ab Version 3. Deprecated ab Version 3.1 und ersetzt durch /etc/opsi/backendManager.conf.d/*

Konfigurationsdatei für den opsiconfd in dem festgelegt wird welche Backends (Datei/LDAP) verwendet werden und wo welche Daten gespeichert werden bzw. über welche Befehle bestimmte Aktionen ausgeführt werden.

12.1.1.6. /etc/opsi/backendManager.conf.d/*

Ab Version 3.1

Konfigurationsdateien für den opsiconfd in denen festgelegt wird welche Backends (Datei/LDAP) verwendet werden,wo welche Daten gespeichert werden bzw. über welche Befehle bestimmte Aktionen ausgeführt werden und welche Serviceaufrufe zur Verfügung stehen.

Die *.conf Dateien dieses Verzeichnis werden zur Laufzeit in alphabetischer Reihenfolge aneinander gefügt und ersetzen in diesem Zustand die backendManager.conf. Durch Einfügen von kundenspezifischen Dateien können ausgewählte Methoden 'überschrieben' werden ohne das diese Änderungen beim nächsten Update verloren gehen.

12.1.1.7. /etc/opsi/hwaudit/*

Ab Version 3.2

Hier finden sich Konfigurationen zur Hardwareinventarisierung

Im Verzeichnis `locales` liegen die Sprachanpassungen.

In der Datei `opsihwaudit.conf` ist die Abbildung zwischen WMI Klassen und der opsi Datenhaltung konfiguriert.

12.1.1.8. /etc/opsi/opsiconfd.conf

Ab Version 3

Konfigurationsdatei für den opsiconfd in dem sonstige Konfigurationen wie Ports, interfaces, Logging hinterlegt sind.

12.1.1.9. /etc/opsi/opsiconfd.pem

Ab Version 3

Konfigurationsdatei für den opsiconfd in dem das ssl Zertifikat hinterlegt ist.

12.1.1.10. /etc/opsi/opsipxeconfd.conf

Konfigurationsdatei für den opsipxeconfd der für das Schreiben der Startdateien für das Linux-Bootimage zuständig ist. Hier können Verzeichnisse, Defaults und Loglevel konfiguriert werden.

12.1.1.11. /etc/opsi/version

Enthält die Versionsnummer des installierten opsi.

12.1.1.12. /etc/init.d/

Start-Stop Skripte für

- opsi_reinstmgr
- opsi-webconfigedit
- opsi-atftpd

1. opsiconfd

opsipxeconfd

12.2. Bootdateien

12.2.1. Bootdateien in /tftpboot/linux

12.2.1.1. pxelinux.0

Bootfile, der im ersten Schritt vom PXE-Bootprom geladen wird.

12.2.1.2. install und miniroot.gz

Installationsbootimage das per tftp an den Client bei der Reinstallation übertragen wird.

12.2.2. Bootdateien in /tftpboot/linux/pxelinux.cfg

12.2.2.1. 01-<mac adresse> bzw. <IP-NUMMER-in-Hex>

Dateien mit der Hardwareadresse des Clients und dem Prefix 01- sind auf dem depot-server als clientspezifische Bootfiles zu finden. Sie sind zumeist über den reInstallationsManagers als named pipes erzeugt und sollen eine Reinstallation des Clients einleiten.

12.2.2.2. default

Die Datei default wird geladen, wenn es keine clientspezifischen Dateien gibt. Wird diese Datei geladen, so bootet der Client danach lokal weiter.

12.2.2.3. install

Informationen zum boot des Installationsbootimages die vom opsi-Reinstallationsmanager in die named pipe geschrieben werden.

12.3. Dateien der File-Backends

Achtung: opsi ist hochgradig Konfigurierbar. Die hier angegebenen Orte entsprechen den Defaults mit denen opsi ausgeliefert wird. Welche Orte tatsächlich in Benutzung sind ist in den Dateien in /etc/opsi/backendManager.conf.d/ festgelegt.

12.3.1. File3.1-Backend

12.3.1.1. Übersicht

Die Dateien des File31 Backends finden sich in /var/lib/opsi dem Heimatverzeichnis des opsiconf-Daemons. Das folgende Schema gibt einen Überblick der Verzeichnisstruktur.

```
/var/lib/opsi┐
             ├─depot/                      (for future use: depotshare)
             ├─log/                        (for future use: logshare)
             ├─utils/                      (for future use: utilsshare)
             ├─audit                       Inventory - Dateien
             ├─products                    Ablage der per opsi-package-manager
             │                             hochgeladenen Pakete
             ├─config/┐                    configshare
                      ├─clientgroups.ini   Client gruppen
                      ├─global.ini         Netzwerk und Zusatzkonfiguration
                      ├─clients/           (<pcname.ini> files)
                      ├─templates/         (template für <pcname.ini>
                      ├─depots/┐
                               ├─<depotid>/┐
                                           ├─depot.ini
                                           ├─products/┐
                                                      ├─localboot/┐
                                                      │  (product control
                                                      │   files)
                                                      └─netboot/┐
                                                         (product control
                                                         files)
```

- Logging sowie Hard- und Softwareinventarisierung
  Die durch das Produkt hwaudit bzw. durch das bootimage ermittelten Hardwareinformationen werden unter <configshare>/pclog/<pcname>.hw gespeichert.
  Die durch das Produkt swaudit ermittelten Softwareinformationen werden unter <configshare>/pclog/<pcname>.sw gespeichert.

12.3.1.2. Konfgurationsdateien in /var/lib/opsi/config

12.3.1.2.1. clientgroups.ini

Die Datei enthält die Informationen über Client-Gruppen.

```
[groupname]
membername
membername
(....)
```

Beispiel

```
[Abteilung 3]
pca26
pca39
pcmeyer
```

12.3.1.2.2. global.ini

Hier finden sich die Defaultwerte der Sektionen `[networkconfig]` und `[generalconfig]` der Clientkonfiguration. Diese können für den einzelnen Client in der <pcname>.ini überschrieben werden. Die Struktur dieser Datei ist daher für diese Sektionen die selbe, wie weiter unten für die <pcname>.ini Dateien beschrieben.

12.3.1.3. Konfgurationsdateien in /var/lib/opsi/config/clients

12.3.1.3.1. <pcname>.ini

In der dieser Datei werden die Client spezifischen Konfigurationen zusammen gefasst. Die Informationen werden mit denen aus der global.ini zusammengefasst, wobei Informationen aus der <pcname>.ini Vorrang haben.

Diese Dateien können folgende Sektionen haben:

12.3.1.3.1.1. [generalconfig]

Allgemeine Einträge für den Client. Einträge aus dieser Sektion werden über den Serviceaufruf getGeneralConfig_hash und im bootimage zum Patchen von Konfigurationsdateien zur Verfügung gestellt.

Beispiel:

```
pcptchbitmap1 = winst1.bmp
pcptchbitmap2 = winst2.bmp
pcptchlabel1 = opsi
pcptchlabel2 = uib umwelt informatik buero gmbh
```

Icons und Beschriftung im 'netmount' Fenster der pcptch.exe

```
SecsUntilConnectionTimeOut = 120
```

Timeout der pcptch.exe (netmount-Fenster)

```
button_stopnetworking=immediate
```

Im netmount-Fenster soll der Abbrechenknopf sofort gezeigt werden.

```
test = 123
```

Benutzerdefinierter key

```
os = winxppro
```

Defaultwert für die Betriebssysteminstallation

12.3.1.3.1.2. [networkconfig]

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

Die URL besteht aus drei Teilen:

1. Protokoll: Hier wird zur Zeit nur smb unterstützt.
2. Sharename (\\schleppi\opt_pcbin): Dieser Teil wird gemountet. Ist weiter unten für diesen Share ein Laufwerksbuchstaben beschrieben, so wird der Share auf diesen Laufwerksbuchstaben gemountet.
3. Das Path-Verzeichnis, in dem sich die konkreten Informationen (hier Softwarepakete) auf dem Share finden.

```
depotdrive=<Laufwerksbuchstaben auf den depoturl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

```
configdrive=<Laufwerksbuchstaben auf den configurl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

```
utilsdrive=<Laufwerksbuchstaben auf den utilsurl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

```
nextbootservertype = service
```

Soll der Client über den opsi-Service arbeiten oder mit direkten Dateizugriff ('classic'). Classic geht nur im Backend 'File' nicht aber in den Backends 'File31' oder 'LDAP'. Der Wert wird vom Client ausgelesen und und unter **`HKEY_LOCAL_MACHINE\SOFTWARE\opsi.org\pcptch opsiServerType`** gespeichert.

```
nextbootserviceurl = https://192.168.1.14:4447
```

URL über den der Client den opsi Service auf dem Server erreicht. Achtung: Wenn hier ein Name statt einer IP-Nummer steht muss dieser Name vom Client auflösbar sein. Der Wert wird vom Client ausgelesen und und unter **HKEY_LOCAL_MACHINE\SOFTWARE\opsi.org\pcptch opsiServiceUrl** gespeichert.

```
windomain = dplaptop
```

Name der Samba(Windows)-Domain

12.3.1.3.1.3. [localboot_product_states]

Ersetzt die Sektion [products-installed] aus den früheren Backends.

```
productid = installation state : required action
```

z.B.

```
firefox = installed:setup
```

12.3.1.3.1.4. [netboot_product_states]

```
productid = installation state : required action
```

z.B.

```
winxppro = installed:none
```

12.3.1.3.1.5. [<product>-state]

Welches Paket dieses Produktes ist auf dem Client installiert und wann ist dies geschehen.

```
laststatechange = <timestamp>
packageversion = <value>
productversion = <value
```

z.B.

```
laststatechange = 20070525105058
packageversion = 1
productversion = 2.0.0.3
```

12.3.1.3.1.6. [<product>-install]

```
product property = value
```

z.B.

```
viewer = off
```

12.3.1.3.1.7. [info]

Die im opsi-Konfigurator eingegebenen Infos zu dem Client werden in der Sektion Info abgespeichert. Weiterhin wird hier aufgezeichnet wann der Client sich zuletzt beim opsiconfd gemeldet hat. z.B.

```
[info]
notes =
description = detlef
lastseen = 20070105090525
```

12.3.1.4. Konfgurationsdateien in /var/lib/opsi/config/templates

Hier findet sich die Datei pcproto.ini welche das Standardtemplate zur Erzeugung neuer <pcname>.ini-Dateien ist. Sie hat die selbe Struktur wie die <pcname>.ini Datei.

12.3.1.5. Konfgurationsdateien in /var/lib/opsi/config/depots/<depotid>

Hier findet sich die Datei depot.ini welche die Konfigurationsinfos für das depot enthält: Wie findet der Client den depotshare und wie und wo findet sich auf dem Server der depotshare.

```
[depotshare]
remoteurl = smb://vmix12/opst_pcbin/install
localurl = file:///opt/pcbin/install

[depotserver]
notes = Notes for vmix12
network = 192.168.4.0/24
description = Depotserver vmix12

[repository]
remoteurl = webdavs://vmix12.uib.local:4447/products
localurl = file:///var/lib/opsi/products
```

12.3.1.6. Product control files in /var/lib/opsi/config/depots/<depotid>/products

Diese Verzeichnis hat die Unterverzeichnisse localboot und netboot in denen die control-files der jeweiligen Produkte liegen. Diese enthalten die Metainformationen der Produkte wie z.B. Name, Properties und deren Defaultwerte, Abhängigkeiten,....

Die control files entsprechen den control files wie Sie bei der Erstellung von opsi-Produkten im Verzeichnis <produktname>/OPSI/control erzeugt werden.

Die Controlfiles bestehen aus folgenden Sektionen

- Sektion [Package]
  Beschreibung der Paketversion und ob es sich um ein incrementelles Paket handelt.
- Sektion [Product]
  Beschreibung des Produktes
- Sektion(en) [ProductProperty] (optional)
  Beschreibung von Veränderbaren Produkteigenschaften
- Sektion(en) [ProductDependency] (optional)
  Beschreibung von Produktabhängikeiten

Ein Beispiel:

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

[Product]
type: localboot
id: thunderbird
name: Mozilla Thunderbird
description: Mailclient von Mozilla.org
advice:
version: 2.0.0.4
priority: 0
licenseRequired: False
productClasses: Mailclient
setupScript: thunderbird.ins
uninstallScript:
updateScript:
alwaysScript:
onceScript:

[ProductProperty]
name: enigmail
description: Installiere Verschluesselungs Plugin fuer GnuPG
values: on, off
default: off

[ProductDependency]
action: setup
requiredProduct: mshotfix
```

```
requiredStatus: installed
requirementType: before
```

- [Package]-'Version' ist die Version des Paketes für die Produktversion. Die dient dazu um Pakete mit gleicher Produktversion aber z. B. korrigiertem Winst-Script zu unterscheiden.
- [Package]-'depends' gibt bei einem inkrementellen Paket das Basis Paket an zu dem es inkrementell ist.
- [Package]-'Incremental' gibt an ob es ein inkrementelles Paket ist.
- [Product]-'type' gibt die Art des Produktes an localboot/netboot
- [Product]-'Id' ist ein eindeutiger Bezeichner für das Produkt in der Regel unabhängig von der Version (In opsi 2 hieß das Produktname)
- [Product]-'name' ist der Klartextname des Produkts
- [Product]-'Description' ist eine ergänzende Beschreibung zum Produkt die z.B. im opsi-Configeditor unter 'Beschreibung' angezeigt wird.
- [Product]-'Advice' ist eine ergänzende Beschreibung in der Regel zum Umgang mit dem Produkt die zu Beachten ist und im im opsi-Configeditor unter 'Notiz' angezeigt wird.
- [Product]-'version' ist die Version der eingepackten Software
- [Product]-'Priority' wird zur Zeit noch nicht verwendet. Soll neben Produktabhängigkeiten die Installationsreihenfolge beeinflussen.
- [Product]-'class' wird zur Zeit noch nicht verwendet (und auch nicht angezeigt).
- [ProductProperty]-'name': Anzeigename der Eigenschaft
- [ProductProperty]-'description': Beschreibung der Eigenschaft (Tooltip im opsiconfiged)
- [ProductProperty]-'values' : Liste möglicher, erlaubte Werte. Wenn leer dann ist der Wert frei editierbar.

- [ProductProperty]-'default' : Default Wert der Eigenschaft
- [ProductDependency]-'Action' : Für welche Aktion des Produktes welches Sie gerade erstellen soll die Abhängigkeit gelten (setup, deinstall,...)
- [ProductDependency]-'Requiredproduct': Productid (Bezeichner) des Produkts zu dem eine Abhängigkeit besteht.
- [ProductDependency]-'Required action': Sie können entweder eine Aktion anfordern oder (siehe unten) einen Status. Aktionen können z.B. sein : setup, deinstall, update,...
- [ProductDependency]-'Required installation status': Status den das Produkt zu dem eine Abhängigkeit besteht haben soll. Typischerweise 'installed', liegt ein anderer Status vor so wird das Produkt auf setup gestellt.
- [ProductDependency]-'Requirement type': Installationsreihenfolge. Wenn das Produkt zu dem eine Abhängigkeit besteht installiert sein muss bevor mit der Installation des aktuellen Produkts begonnen werden kann dann ist dies 'before'. Muss es nach dem aktuellen Produkt installiert werden so ist dies 'after'. Ist die Reihenfolge egal so muss hier nichts eingetragen werden.

12.3.1.7. Inventarisierungsdateien /var/lib/opsi/audit

Hier liegen die Dateien der Hardwareinventarisierung (*.hw) und der Softwareinventarisierung (*.sw).

12.4. Dateien des LDAP-Backends

Das opsi-LDAP Schema finden Sie unter `/etc/ldap/schema/opsi.schema`.

12.5. opsi Programme und Libraries

12.5.1. Python Bibliothek

Die opsi Python Module finden sich unter:

/usr/lib/python2.3/site-packages/OPSI/
bzw. unter
/usr/share/python-support/python-opsi/OPSI

12.5.2. Programme in /usr/sbin

1. opsiconfd
   Opsi Konfigurations daemon

opsipxeconfd
: Opsi Daemon welcher für den PXE-Start der Clients die notwendigen Dateien im tftp-Bereich des Servers verwaltet.

12.5.3. Programme in /usr/bin

- opsi-admin
  Kommandozeilen-Interface zur opsi python Library
- opsi-configed
  Aufruf des opsi-Managementinterface
- opsi-convert
  Skript zum Konvertieren zwischen verschiedenen Backends.
- opsi-makeproductfile
  Skript zum opsi-Paket packen
- opsi-newprod
  Skript zum Erstellen eines neuen Produktes
- opsi-packet-manager
  Skript zum installieren und deinstallieren von opsi-Paketen auf einem opsi-server
- opsi-winipatch
  Script zum patchen von INI-Dateien
- opsiuninst (deprecated -> opsi-packet-manager)
  Skript zum löschen von Produkten

- opsiinst (deprecated -> opsi-packet-manager)
  Skript zum Auspacken von opsipaketen
- opsiinstv2 (sehr deprecated Version 2 Kompatibel)
  Skript zum Auspacken von opsipaketen
- opsi-makeproductfilev2 (sehr deprecated Version 2 Kompatibel)
  Skript zum opsi-Paket packen
- makeproductfile (makeproductfilev2) (deprecated)
  Ersetzt durch opsi-makeproductfile
  Skript zum opsi-Paket packen (opsiV2 kompatible Version)
- newprod (deprecated)
  Ersetzt durch opsi-newprod
  Skript zum Erstellen eines neuen Produktes
- sysbackup
  Systemsicherung auf Band oder Platte

12.6. opsi-Logdateien

12.6.1. /var/log

Der opsi reinstallations Manager logt per syslog nach /var/log/syslog

12.6.2. /var/log/opsi/opsiconfd

Hier findet sich die Logdatei des opsiconfd selbst sowie log Dateien zu den Clients. Dabei werden die Dateien als <IP-Nummer>.log angelegt und soweit in /etc/opsi/opsiconfd.conf so eingestellt, zu diesen symbolische Links als <IP-Name>.log erzeugt.

12.6.3. /var/log/opsi/bootimage

Hier findet sich die Logdateien der bootimages zu den Clients. Dabei werden die Dateien als <IP-Name>.log angelegt. Sollte das bootimage den Webservice nicht erreichen können, so findet sich die Logdatei im bootimage unter /tmp/log.

12.6.4. /var/log/opsi/opsipxeconfd.log

Logdatei des opsipxeconfd welcher für den PXE-Start der Clients die notwendiogen Dateien im tftp-Bereich des Servers verwaltet.

12.6.5. Softwareinstallation (c:\tmp)

Der opsi-preloginloader Service prelogin.exe logt normalerweise nicht. Erhöht man in der Registry den Wert von HKEY_LOCAL_MACHINE\SOFTWARE\opsi.org\preloginloader\DebugOutput von Null auf maximal 4 erzeugt die prelogin.exe Einträge im der Windows Ereignisanzeige.

Das netmount Programm pcptch.exe erzeugt eine Logdatei unter c:\tmp\logonlog.txt.

Der opsi-Winst erzeugt eine ausfühliche Logdatei zum letzten Lauf unter c:\tmp\instlog.txt. Eine kumulative Logdatei welche nur Fehler aufzeichnet findet sich unter dem Namen instlog.txt in c:\tmp. Per Konfiguration können diese Informationen auch unter <configshare>/pclog/<pcname>.err gespeichert werden oder per syslog Protokoll an einen Logserver übergeben werden.

13. Registryeinträge

13.1. Registryeinträge des opsi-PreLoginLoaders 3.3.x

13.1.1. opsi.org/general

Die nachfolgenden Registryeinträge liegen unter HKLM/Software/opsi.org/general.

```
configlocal = <0/1> (dword)
```

Die Werte der nachfolgenden Keys werden per tftp aus den opsi Konfigurationsdateien geholt, wenn der Key 'configlocal' nicht existiert oder ungleich 1 ist. Dann muss der Key 'tftpserver' vorhanden sein. Ist der Wert des Keys gleich 1, so wird auf den Versuch per tftp verzichtet und die nachfolgenden Registryeinstellungen verwendet.

```
tftpserver = <server von dem per tftp aus (/tftpboot)/opsi die opsi
Konfigurationsdateien geholt werden können.>
```

13.1.2. opsi.org/shareinfo

Die nachfolgenden Registryeinträge liegen unter HKLM/Software/opsi.org/shareinfo.

```
user = <username der beim mounten der shares verwendet wird>
```

Beispiel: pcpatch

```
pcpatchpass = <mit blowfish und key verschlüsseltes pcpatch passwort> (text)
depoturl = <Url die zu den Softwarepaketen verweist. Muster:
protokoll:\\server\share\dir>
```

Beispiel: smb: (\\schleppi\opt_pcbin\install)

Die URL besteht aus drei Teilen:

1. Protokoll (smb:): Hier wird zur Zeit nur smb unterstützt. Andere Protokolle führen zu einer Fehlermeldung.
2. Share (\\schleppi\opt_pcbin): Dieser Teil wird gemountet. Ist weiter unten für diesen Share ein Laufwerksbuchstabe beschrieben, so wird der Share auf diesen Laufwerksbuchstaben gemountet.

3. Dir-Verzeichnis, in dem sich die konkreten Informationen (hier Softwarepakete) auf dem Share finden.

```
configurl = <Url die zu den pcname.ini dateien verweist. Muster:
protokoll:\\server\share\dir>
```

Hinweis: die Verwendung von configurl ist abgekündigt

Beispiel: smb:\\schleppi\opt_pcbin\pcpatch
Beschreibung: Analog zu depoturl

```
utilsurl = <Url die zu dem Verzeichnis verweist in dem sich winst* und andere
opsi hilfsdateien befinden. Muster: protokoll:\\server\share\dir>
```

Beispiel: smb:\\schleppi\opt_pcbin\utils

Beschreibung: Analog zu depoturl

Hinweis: die Verwendung von utilsurl ist abgekündigt

```
depotdrive = <Laufwerksbuchstaben auf den depoturl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

```
configdrive = <Laufwerksbuchstaben auf den configurl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

```
utilsdrive = <Laufwerksbuchstaben auf den utilsurl gemountet wird>
```

Beispiel: P: (mit Doppelpunkt)

try_secondary_user=0
Wenn 1 wird der user aus Eintrag smbusername1 für den mount verwendet.

smbusername1=opsiserver\pcpatch

13.1.3. opsi.org/preloginloader

Schlüssel [HKEY_LOCAL_MACHINE\SOFTWARE\opsi.org\preloginloader]

"PcPCallMode"=dword:00000001
- deprecated

"DebugOutput"=dword:00000004
- Eventlog: 0=errors only >=4 = verbose

"RebootOnBootmodeReins"=dword:00000001

- Reboot, wenn Bootmode REINS

"RebootOnServicePackChange"=dword:00000001

- Reboot falls Servicepack sich geändert hat

"WaitForPcpExit"=dword:00000000

- deprecated

"RemoveMsginaOnDeinst"=dword:00000001

- bei Reinstallation msgina deinstallieren

"UtilsDir"="C:\\opsi\\utils"

- Utilsdir

"PcptchExe"="C:\\opsi\\utils\\pcptch.exe"

- das wird von prelogin.exe gestartet

"WinstRegKey"="SOFTWARE\\Hupsi\\winst"

- hier wird geschaut, ob der Winst rebooten möchte

"LoginBlockerStart"=dword:00000001

- pgina wartet auf READY aus der Named Pipe

"LoginBlockerTimeoutConnect"=dword:00000005

- Timeout in Minuten für ein Pipe-Connect zum preLoginLoader
- nur wenn LoginBlockerStart auf 1 steht

"LoginBlockerTimeoutInstall"=dword:000000B4

- Timeout in Minuten für warten auf READY (danach Freigabe des Login)
- nur wenn LoginBlockerStart auf 1 steht

RunServiceAs=1

wenn 1 werden die nachfolgenden RunServiceAs Einträge ausgewertet.
wenn dort kein user angegeben ist (was empfohlen ist) ,
wird der user pcpatch erstellt
und nach Verwendung wieder gelöscht

RunServiceAsDom=
: zu verwendende Domain des login users (Verwendung nicht empfohlen)

RunServiceAsUsr=
: zu verwendende Name des login users (Verwendung nicht empfohlen)

RunServiceAsPas=
: zu verwendendes Passwort des login users
(Verwendung nicht empfohlen)

ShutdownNum=
: Welche Shutdownmethode soll verwendet werden.
Default = 0 = interne API

13.1.4. opsi.org/pcptch

Schlüssel [HKEY_LOCAL_MACHINE\SOFTWARE\opsi.org\pcptch]

1. repeatServiceConnectNo
   Wie oft soll der Verbindungsversuch mit dem Service wiederholt werden (Default: insgesamt 3 Versuche)
2. Bitmap1=winst1.bmp
   Name der ersten Bitmap
3. Bitmap2=winst2.bmp
   Name der zweiten Bitmap
4. button_stopnetworking=immediate
   Wenn das String-Feld button_stopnetworking den Wert 'immediate' hat, wird der Knopf StopNetzzugriff?-Knopf sofort angezeigt und kann sofort disabled werden (für PCs, die nicht am Netz hängen)
5. copyDefaultUser=0
6. label1=opsi
7. label2=uib gmbh

8. loadBitmap=1
   Bitmaps werden nachgeladen

9. makeLocalCopyOfIniFile=0
   deprecated

10. makeLocalWinst=1
    lege lokale Kopie des Winst an

11. mountdrive=1
    Mounte Laufwerke

12. opsiServerType=service
    Bestimmt ob die pcptch.exe im opsi 2 Stil File basiert oder nach einem opsi-service suchen soll. Mögliche Werte:
    classic -> opsi 2 Stil
    service -> opsi 3 Stil

13. opsiServiceURL=https://<serverip>:4447
    Gibt die URL an unter dem der opsi-Service erreicht werden kann.
    z.B. https://bonifax.uib.local:4447
    (IP-Nummer angeben wenn Server nicht im DNS)

14. patchleveltyp=

15. pcprotoname=pcproto.ini

16. pingcheck=1
    Versucht per Ping die Erreichbarkeit des Servers zu überprüfen

17. “SecsUntilConnectionTimeOut“=“10“
    wartet 10 Sekunden auf Netzwerkverbindung, sonst ohne Netz weiter
    Wert = 0 bedeutet deaktiviert

18. winstLocalDirectory=C:\Programme\opsi.org\preloginloader\utils

13.2. Registryeinträge des opsi-Winst

13.2.1. Steuerung des Logging per syslog-Protokoll

Schlüssel HKLM\Software\opsi.org\syslogd

DWord-Variable remoteerrorlogging wird nach folgendem Schema ausgewertet:

RemoteErrorLogging = (trel_none, trel_filesystem, trel_syslog);
// in registry values 0, 1, 2

Falls das Logging mittels dem syslog-Protokoll erfolgt ("remoteerrorlogging"=dword:00000002), so gibt die String-Variable sysloghost den IP-Namen des LogHost an.

DWORD-Variable syslogfacility gibt an, was als Quelle der syslog-Nachricht übergeben wird. Der default ist ID_SYSLOG_FACILITY_LOCAL0

Bedeutungen:

```
  ID_SYSLOG_FACILITY_KERNEL     = 0;  // kernel messages
  ID_SYSLOG_FACILITY_USER       = 1;  // user-level messages
  ID_SYSLOG_FACILITY_MAIL       = 2;  // mail system
  ID_SYSLOG_FACILITY_SYS_DAEMON = 3;  // system daemons
  ID_SYSLOG_FACILITY_SECURITY1  = 4;  // security/authorization messages (1)
  ID_SYSLOG_FACILITY_INTERNAL   = 5;  // messages generated internally by
syslogd
  ID_SYSLOG_FACILITY_LPR        = 6;  // line printer subsystem
  ID_SYSLOG_FACILITY_NNTP       = 7;  // network news subsystem
  ID_SYSLOG_FACILITY_UUCP       = 8;  // UUCP subsystem
  ID_SYSLOG_FACILITY_CLOCK1     = 9;  // clock daemon (1)
  ID_SYSLOG_FACILITY_SECURITY2  = 10; // security/authorization messages (2)
  ID_SYSLOG_FACILITY_FTP        = 11; // FTP daemon
  ID_SYSLOG_FACILITY_NTP        = 12; // NTP subsystem
  ID_SYSLOG_FACILITY_AUDIT      = 13; // log audit
  ID_SYSLOG_FACILITY_ALERT      = 14; // log alert
  ID_SYSLOG_FACILITY_CLOCK2     = 15; // clock daemon (2)
  ID_SYSLOG_FACILITY_LOCAL0     = 16; // local use 0  (local0)
```

```
ID_SYSLOG_FACILITY_LOCAL1    = 17; // local use 1  (local1)
ID_SYSLOG_FACILITY_LOCAL2    = 18; // local use 2  (local2)
ID_SYSLOG_FACILITY_LOCAL3    = 19; // local use 3  (local3)
ID_SYSLOG_FACILITY_LOCAL4    = 20; // local use 4  (local4)
ID_SYSLOG_FACILITY_LOCAL5    = 21; // local use 5  (local5)
ID_SYSLOG_FACILITY_LOCAL6    = 22; // local use 6  (local6)
ID_SYSLOG_FACILITY_LOCAL7    = 23; // local use 7  (local7)
```

14. History

14.1. Unterschiede der opsi Version 3.3 zu Version 3.2

14.1.1. Was ist neu in opsi 3.3

- Unterstützung von mehreren Standorten bei zentraler Verwaltung
    - Mit Hilfe der Multi-Depotshare Erweiterung von opsi werden verschiedene Standorte bei zentraler Verwaltung und dezentralen Software-Depots und dezentralen Bootservern unterstützt.
    - Bereitstellung von dezentralen shares über dezentrale depotserver
    - Bereitstellung von dezentralen TFTP/PXE bootservern über dezentrale depotserver
    - Einfache Installation eines dezentralen depotservers und Anbindung an den zentralen config-server
    - Diese Funktion ist nur für das BACKEND_FILE31 implementiert. Die Implementation des entsprechenden BACKEND_LDAP folgt in Kürze
    - Diese Funktion gehört zur 'Professional Edition' und wird nur im Rahmen eines entsprechenden Supportvertrages unterstützt.
- Neues Werkzeug opsi-package-manager zur Paketverwaltung
    - Installation und Deinstallation von Paketen von einem oder mehreren depot-servern (Ersetzt die abgekündigten Befehle opsiinst und opsiuninst)
    - Ausgabe welche Pakete in welcher Version auf welchem Server installiert sind.
    - Ausgabe welche Differenzen es zwischen depotservern gibt.
    - Extrahieren eines existierenden Paketes um es zu modifizieren
- Verbesserungen beim opsi-configed

    - Unterstützung von mehreren depots
    - MAC-Adresse eines Clients nachträglich editierbar
    - verbesserter Dialog zum Anlegen eines Clients
- MySQL-basiertes Backend für die Inventarisierung
    - Das MySQL basierende Backend ermöglicht beliebige SQL-basierte Abfragen zur Erstellung eigener Reports.
    - Das MySQL-Backend enthält eine History-Funktion, die es erlaubt Änderungen an der inventarisierten Hardwäre nachzuverfolgen.
    - Diese Funktion gehört zur 'Professional Edition' und wird nur im Rahmen eines entsprechenden Supportvertrages unterstützt.
- opsi-winst Erweiterungen
    - Die winst-Oberfläche ist mit frei gestaltbaren Skins aufgebaut und kann so von opsi-Kunden besser an ihre 'Corporate Identity' angepasst werden.
    - Erweiterung um ein Kommando ExitWindows/ShutdownWanted, welches den Rechner nach Abschluss der Installationen herunterfährt.
    - Erweiterung des copy Befehls zur Unterdrückung automatischer reboots.
    - Neue Funktion zur Feststellung der System Lokalisierung.
    - Neue Funktion zur Feststellung der Versionsinformationen einer Datei.
    - Neue Funktion zur Feststellung des Exitcodes eines aufgerufenen Programms.
    - Neue Sekundäre Sektion zum Aufruf eines Scriptes mit einem externen Interpreter.
    - Neue Funktion zum Ermitteln von Werten aus einer Map (Stringliste).
    - Vereinfachungen beim Bilden von Teillisten einer Stringliste.

    - Neue Anweisungen zur Untersuchung des zeitlichen Verhaltens eines Skripts

- Verbesserungen beim opsi-Agenten 'preloginloader'
    - Ausrollen des preloginloader mit opsi-preloginloader-deploy ohne reboot.
    - Verbessertes Zeitverhalten bei fehlendem opsi-server
    - Initiale Installation des Loginblockers als Default
    - Kein permanenter lokaler user mehr
    - Verbessertes und automatisiertes Erfassen von MAC-Adressen
    - Erweiterte Möglichkeiten zur Konfiguration mit welchem user Laufwerke gemountet werden
    - Verbesserte Installationsroutine
- Verbesserung Treiberintegration
    - Erweiterung der Befehle zur Treiberintegration
- Erweiterung der opsi-admintools
    - Aktueller opsi-Configed als Applikation
    - Entpacken auch von *.opsi Paketen: 7-zip
    - Editor (nicht nur) mit opsi-Winst Syntax-Highlightning: jedit
    - SSH-Terminal: Putty
    - Vergleichen von Dateien: WinMerge.
    - XML-Editor: Pollo
    - XML-Diff-Tool
    - LDAP-Explorer: JXplorer

    - Werkzeug für Interaktives Setup mit aufgezeichneten Antworten: Autohotkey.
    - Setup Switch detector: Ussf
- aktualisiertes bootimage
    - aktueller Kernel
    - NTFS Write Unterstützung
- Verbesserung bei WakeOnLAN
- Diverse Bugfixes
- Abkündigung der Unterstützung des opsi 2.x/3.0 File-Backends für die nächste opsi-Release.
    - Anwender dieser Versionen sollten dringend auf opsi 3.3 und File31-Backend updaten.
- Aktualisierte Dokumentationen und Installationsmedien.

Vokabular:

config-server	Die Funktionalität, welche die Haltung, Bereitstellung und Management von Konfigurationsdaten auf einem ->opsi-server verwirklicht.
depot-server	Die Funktionalität, welche die Bereitstellung von Software-Depots auf Shares für die Clients und einen tftp-Bereich für die PXE-Boots auf einem ->opsi-server verwirklicht. Vor opsi 3.3 allgemein für opsi-server verwendet.
opsi-server	Ein Server, der im Rahmen von opsi Dienste bereitstellt. Üblicherweise -> config-server und -> depot-server.

14.1.2. Was sollten Sie lesen

In diesem Handbuch:

•3.2.3 Depotauswahl Seite 15

•3.2.5 Client Bearbeitung / WakeOnLan / Client erstellen / Client umziehen Seite 18

•3.4 Werkzeug opsi-package-manager: opsi-Pakete (de-) installieren Seite 25

•5.1.9 Vereinfachte Treiberintegration in die automatische Windowsinstallation Seite 91

•7 opsi-server mit mehreren Depots Seite 102

•9.3 MySQL-Backend für Inventarisierungsdaten Seite 112

Im Winst-Handbuch:

- ExitWindows /Shutdownwanted
- skinnable Winst

14.2. Unterschiede der opsi Version 3.2 zu Version 3.1

14.2.1. Überblick

- Verbesserte Hardwareinventarisierung
- Mit dem opsi-Produkt hwaudit werden Hardwareinformationen per WMI ausgelesen und an den opsi-server zurückgemeldet.
- Darstellung der Ergebnisse der Hardwareinventarisierung im opsi-configed in einer nach Geräteklassen sortierten Übersicht.
- Möglichkeit der Auswahl von Clients nach Hardware-Kriterien wie z.B. Größe des Arbeitsspeichers.

- Bereitstellung von serverseitigen Erweiterungen um Hardware-Inventarisierungsdaten über den Webservice zu kommunizieren und im Backend zu speichern.
- Erweiterung des opsi-Winst zur direkten Ausführung von Python Skripten aus Winst-Sektionen heraus.
- Verbesserte Softwareinventarisierung
    - Mit dem opsi-Produkt swaudit werden Softwareinformationen aus der Registry ausgelesen und an den opsi-server zurückgemeldet.
    - Darstellung der Ergebnisse der Softwareinventarisierung im opsi-configed.
    - Bereitstellung von serverseitigen Erweiterungen um Software-Inventarisierungsdaten über den Webservice zu kommunizieren und im Backend zu speichern.
- Beschleunigtes Verfahren zur unattended Installation von WinXP/2k (ohne DOS)
- Verbessertes Verfahren zum Sichern und Wiederherstellen von NTFS-Images
- Weitere netboot Produkte:
    - wipedisk: zum schnellen oder sehr sicheren löschen von Festplatten
    - memtest: Memorytest des Clients
- Diverse Bugfixes
- Aktualisierte Dokumentationen und Installationsmedien:
    - opsi 3.2 Installationshandbuch
    - opsi 3.2 Handbuch
    - opsi-winst Handbuch
    - opsi-winst Quick Reference
    - Virtueller opsi 3.2 opsi-server für Vmware

- InstallationsCD für opsi 3.2 opsi-server

14.2.2. Was sollten Sie lesen

Opsi 3.2 bringt einige Neuerungen mit, über die Sie sich informieren sollten. Hierzu lesen Sie bitte:

Als Einführung dieses Kapitel

Weiterhin:

- Kapitel: swaudit und hwaudit: Produkte zur Hard- und Softwareinventarisierung
- Kapitel zu netboot Produkten ntfs-image, wipedisk, memtest
- Das opsi Integrationshandbuch ist ab opsi 3.2 in dieses Handbuch integriert und wird als eigenes Handbuch nicht weiter gepflegt.
- Aktualisiertes Winst-Handbuch

14.2.3. Umstellung auf opsi V3.2

Die Umstellung Ihrer opsi Umgebung von opsi 3.1 auf opsi 3.2 ist im opsi-server Installationshandbuch beschrieben.

14.3. Unterschiede der opsi Version 3.1 zu Version 3.0

14.3.1. Überblick

- Integration der Bootimage basierten Produkte in die Standard-Datenhaltung
    - Bootimage Produkte wie OS-Installation, Hardware Inventarisierung, Image erstellen oder Image zurückspielen werden in die normale Datenstruktur der anderen Produkte integriert und sind dann genauso zu verwalten.
    - Der opsi-reinstmgr wird durch den opsipxeconfd abgelöst, der über die opsi-Python-Library direkten Zugriff auf die opsi-Konfigurationen besitzt.

- Erweiterung des opsi-configed
    - Informationen zur auf dem Client installierten Software- und Paketversion eines Produktes werden angezeigt und ausgewertet.
    - opsi Basiskonfigurationen (Generalconfig) werden editierbar.
- Neues Scripte zum Initialen Rollout des opsi-preloginloaders
    - opsi-deploy-preloginloader
      Es ermöglicht, direkt vom Server ausgehend den opsi-prelogloader auf die Clients einzuspielen. Voraussetzung ist die Kenntnis des Admin-Passworts der Clients und ein offener C$- und Admin-Share.
    - setup_service.cmd
      Sind die Anforderungen für das Script opsi-deploy-preloginloader nicht gegeben, kann vom Administrator auf dem Client ein Script gestartet werden, welches nach Eingabe von User/Passwort eines opsi-admins (der über im Script hinterlegte Passwörter) den Client per opsi-Service erzeugt und den preloginloader installiert.
- Vereinfachte Treiberintegration auf Basis von PCI-Kennungen
    - Ein neues Script ermöglicht es der Windowsinstallation nur die tatsächlich benötigten zusätzlichen Treiber zu übergeben.
- Verbesserungen beim opsi-preloginloader / opsi-Winst
    - Der opsi-Preloginloader ist nun weniger empfindlich gegenüber Problemen mit nicht funktionierender Namensauflösung.
    - Bugfix bei der Unterstützung englischsprachiger Systeme.
    - opsi-Winst Funktion zu Ermittlung der aktuellen System-Sprache zur Laufzeit um multilinguale Pakete zu unterstützen.
    - opsi-Winst unterstützt Aufrufe des opsi-Service im Rahmen eines Winst-Scriptes

- Konvertierung zwischen beliebigen Backend-Konfigurationen, beispielsweise von File-Backend zu LDAP-Backend.
- Weitere Anpassungen an die Vorgaben des Linux Standard Base durch das neue File3.1-Backend.

14.3.2. Was sollten Sie lesen

Opsi 3.1 bringt einige Neuerungen mit, über die Sie sich informieren sollten. Hierzu lesen Sie bitte:

Als Einführung dieses Kapitel

Weiterhin:

- Kapitel: Nachträgliche Installation des opsi-Preloginloaders
- Die Beschreibung des von Ihnen verwendeten Backends
- Die Beschreibung des opsi-configed
- Kapitel zur Treiberintegration in die Windowsinstallation
- Aktualisiertes Winst-Handbuch

14.3.3. Backends

opsi 3.1 Unterstützt nun folgende Backends:

- File
  dateibasiertes Backend. Dieses ist kompatibel zu opsi2.x und mit seiner Lage in /opt/pcbin/pcptch nicht konform zum 'Linux Standard Base'.
- File3.1
  dateibasiertes Backend. Dieses ist nicht kompatibel zu opsi2.x und mit seiner Lage in /var/lib/opsi konform zum 'Linux Standard Base'. Die wesentlichen Unterschiede zum 'File'-Backend sind:

    - Zusammenfassung der Verwaltung von 'normalen' Produkten und bootimage basierten (localboot- und netboot Produkten) in einer Datei pro Client.
    - Auftrennung von Status und Required Action der Produkte

- LDAP
  Standard Open-LDAP opsi-Backend
- Univention-LDAP
  Backend der opsi Spezial Edition opsi4ucs.

Wenn Sie opsi neu installieren ist das Defaultbackend File3.1.

14.3.4. Umstellung auf opsi V3.1

Die Umstellung Ihrer opsi Umgebung von opsi 3.0 auf opsi 3.1 ist im opsi-server Installationshandbuch beschrieben.

14.4. Unterschiede der opsi Version 3 zu Version 2

14.4.1. Überblick (Was sollten Sie lesen)

Opsi 3 bringt eine Fülle von Neuerungen mit, über die Sie sich informieren sollten. Hierzu lesen Sie bitte:

Als Einführung dieses Kapitel 14.4 Unterschiede der opsi Version 3 zu Version 2

Zum neuen Distributions Paketformat sollten Sie das entsprechende Kapitel aus dem opsi Integrations-Handbuch lesen.

14.4.2. Konzeptionell

In opsi Version 2 war die gesamte Datenhaltung ausschließlich Datei basiert. Alle opsi Komponenten haben direkt mit diesen Dateien gearbeitet.

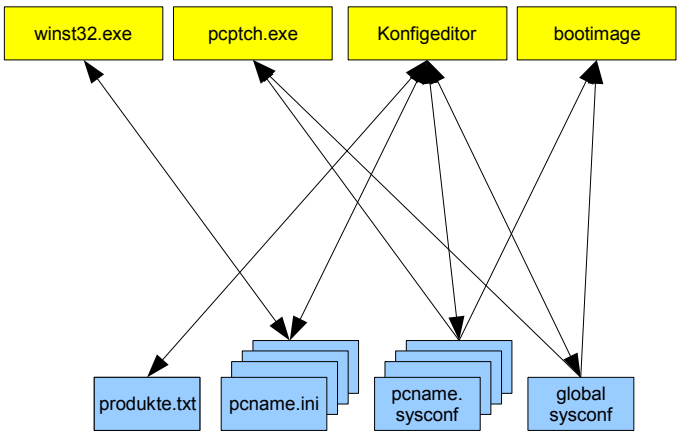


Abbildung 28: Veralteter opsi V2 Direktzugriff auf die Daten

In opsi Version 3 greifen die opsi Komponenten nicht mehr direkt auf die Datenhaltung zu. Vielmehr arbeiten alle Komponenten mit dem opsi-Konfigurations Daemon. Dieser stellt über einen Webservice die notwendigen Dienste bereit. Das Lesen und Speichern der Daten aus der Datenhaltung erfolgt durch den Dämonen.

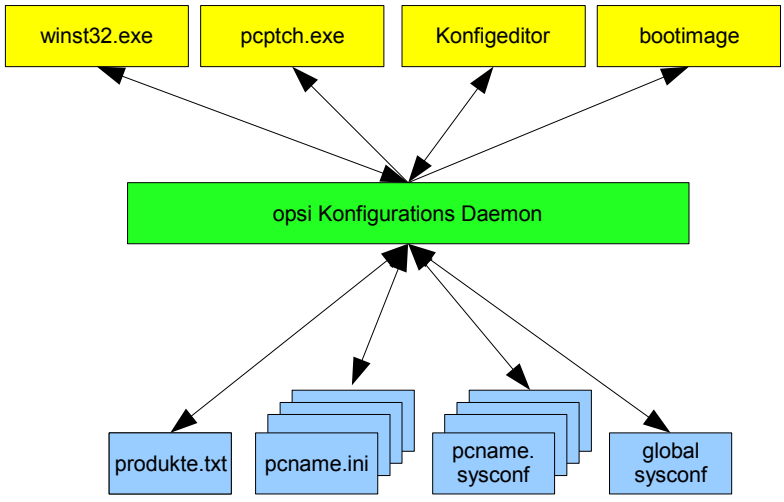


Abbildung 29: opsi V3: Verwendung eines Webservice zum Datenzugriff

Dadurch wird es relativ einfach möglich auch andere Datenhaltungen wie die bisherige Datei basierte bereit zustellen. So ist ab opsi Version 3 auch ein Betrieb mit LDAP als Datenbasis möglich.

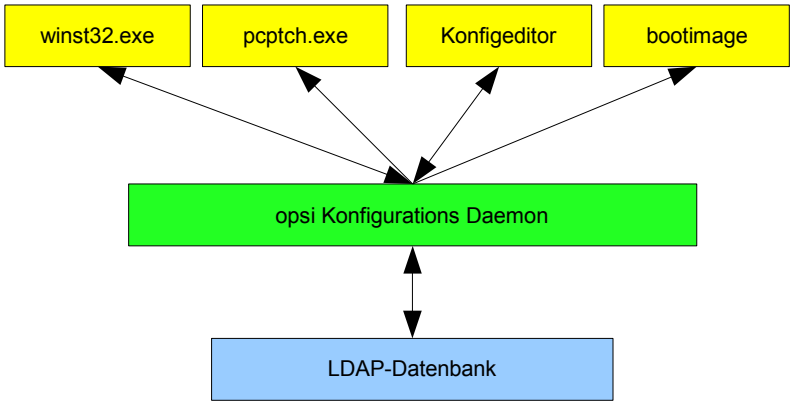


Abbildung 30: Verwendung alternativer Datenhaltungen durch den Webservice

Aus Kompatibilitätsgründen verfügen Winst und pcptch.exe auch noch über den klassischen Modus mit direkten Dateizugriff.

Die Umsetzung erfolgt in opsi 3 durch die Bereitstellung einer Python Library. In dieser sind von der Datenhaltung abstrahierte Aufrufe zur opsi Konfiguration implementiert. Diese Aufrufe bilden eine allgemeine API zur opsi-Konfiguration. Diese API wird durch den opsiconfigd in einem JSON basierten Webservice bereit gestellt der z. B. zur Kommunikation mit dem opsi-Configed verwendet wird. Das Programm opsi-admin stellt wiederum ein Kommandozeilen-Interface zu dieser API zur Verfügung.

Ein weiter Teil der opsi Python-Library implementiert die Zugriffe auf die konkreten Datenhaltungen (Backends) die über den Backendmanager konfiguriert werden können.

14.4.3. Verbesserungen in der Handhabung

Neben den technischen Veränderungen 'unter der Haube' bringt opsi Version 3 eine Reihe von Verbesserungen die das Arbeiten mit opsi vereinfachen:

- Konfigurationseditor: opsi-Configed
    - Gruppenverwaltung:
        - Mehrfachselektion von Clients und gleichzeitige Bearbeitung
        - Speichern und Laden von Gruppen die zur Selektion von Clients verwendet werden können.

        - Filtermöglichkeit der anzuzeigenden Clients z. B. nach installierter Software
    - Wake on LAN mit dem Konfigurationseditor
    - Clientliste sortierbar nach Clientname, Beschreibung und letzter Anmeldung bei opsi
    - Auftrennung der bisherigen Schalterstellungen in eine Statusinformation und eine Information über die nächste geplante Aktion
    - Produktliste sortierbar nach Installationsstatus und Aktionsstatus
    - Bereitstellung des Konfigurationseditors als Web-Applet
    - Verbesserte Darstellung des Hardwareinventars
    - Einfaches Erstellen und Löschen von Clients
- Neues Paketformat zur Installation von opsi-Produkten auf einem Depotserver
    - Vereinfachte menügeführte Erstellung
    - Informationen über Software- und Paketversion sowie möglicher kundenspezifischer Erweiterungen im Paketnamen, im Installationsverzeichnis und angezeigt im opsi-Configed zur Unterstützung der Produktverwaltung (Product Lifecycle Management).
    - Paketverwaltung ohne root Rechte
    - Eine Beschreibung hierzu finden Sie im opsi-Integrationshanduch im Kapitel: 'opsi 3: Einbindung des Produkts in das Verteilverfahren von opsi '
    - Die Befehle zum Handling von Paketen im alten Format stehen als opsiinstv2 und makeproductfilev2 weiterhin zur Verfügung.
- Kommandozeilen Werkzeug 'opsi-admin' zur Skriptgesteurten verwaltung von opsi.
- opsi4ucs: opsi für Univention Corporate Server (UCS)

- Integration der opsi-Datenhaltung in das UCS-LDAP
- Integration der opsi Konfiguration in das 'Univention Admin Interface'
- Hierzu gibt es ein gesondertes Handbuch 'opsi4ucs'.

14.4.4. Vokabular

Im Rahmen von opsi V3 sind einige neue Begrifflichkeiten entstanden bzw. Bedeutungen haben sich gegenüber opsi V2 leicht verändert.

Hier die wichtigsten Begriffe:

Actionrequest	Ab opsi V3 werden der derzeitige Installationsstatus und die nächste geplante Aktion (Actionrequest) getrennt betrachtet. Typische Actionrequests sind 'setup', 'deinstall' und 'update'. -> Installationsstatus
Backend	opsi V3 unterstützt unterschiedliche Methoden der Datenhaltung wie File oder LDAP. Diese werden als Backends bezeichnet und über den -> backendmanager konfiguriert.
backendmanager	Programm / Konfigurationsdatei in der festgelegt wird, welche Daten wie und wo gespeichert werden.
clientId	Eindeutige Bezeichnung des Clients durch Verwendung des 'full qualified hostnames' also IP-Name inclusive Domain z.B. dpvm02.uib.local
hostId	Eindeutige Bezeichnung eines Rechners durch Verwendung des 'full qualified hostnames' also IP-Name inclusive Domain z.B. dpvm02.uib.local
Installationsstatus	Ab opsi V3 werden der derzeitige Installationsstatus und die nächste geplante Aktion (Actionrequest) getrennt betrachtet.

	Typische Installationsstati sind 'installed' und 'not installed'. -> Actionrequest
LastSeen	Zeitstempel wann ein Client sich zuletzt über den Service bei opsi gemeldet hat.
localboot Produkt	Ein opsi Paket welches über den opsi-preloginloader installiert wird.
netboot Produkt	Ein opsi Paket welches über den Start eines bootimages ausgeführt wird.
opsi-admin	opsi V3 Kommandozeilen Interface zur opsi-Konfiguration
opsiHostKey	siehe pckey
opsi-Configed	opsi V3 Konfigurationswerkzeug als Java Applikation und Applet
opsiconfd	Deamon der die opsi-Konfigurations-API als JSON basierten Webservice zur Verfügung stellt
product properties	Zusätzliche Einstellungen zu einem opsi-Produkt die clientspezifisch gesetzt werden können und bei der Installation ausgewertet werden.
Produkt-ID	Eindeutiger Bezeichner eines opsi-Produkts. Dieser darf keine Leerzeichen oder Sonderzeichen (außer Bindestrich enthalten). In opsi V2 auch als -> Produktname was aber in opsi V3 eine andere Bedeutung hat. Beispiel für eine ProductId: acroread
Produktname	In opsi V3 der Klartextname eines Produktes. Beispiel für einen Produktnamen: 'Adobe Acrobat Reader'
Server Produkt	Ein opsi Produkt welches Installationen auf dem Server ausführt die nicht für den Client bestimmt sind.

14.4.5. Umstellung auf opsi V3

Die Umstellung Ihrer opsi Umgebung von opsi 2.5 auf opsi 3.0 ist im opsi-server Installationshandbuch beschrieben.

15. Anhang

15.1. Verwaltung der PCs über dhcp

15.1.1. Was ist dhcp?

Mittels dhcp wird die Anbindung der PC-Clients am Netz realisiert. Über dieses *TCP/IP*-Protokoll können Server und seine Clients Informationen über die Konfiguration des Netzes und seiner Bestandteile austauschen und festlegen.

Das Protokoll *dhcp* ist als Erweiterung des älteren *BOOTP* zu sehen. Es ermöglicht darüber hinaus eine dynamische Zuordnung von IP-Nummern zu PC-Clients.
Diese Funktion wird hier allerdings nicht genutzt bzw. beschrieben.

Mittels *dhcp* können die meisten gängigen Netzwerkkarten benutzt werden, sofern sie über ein Bootprom verfügen oder sich nachrüsten lassen:

- Netzwerkkarten mit dem (neueren) PXE (= Preboot Execution Environment).
- Netzwerkkarten mit älteren TCP/IP-Bootproms, die lediglich das Protokoll BOOTP kennen (z.B. Bootproms der Firma bootix).

Die *IP*-Adresse eines PC-Clients steht in der /etc/hosts.

Die übrigen Konfigurationsdaten befinden sich in der Datei **/etc/dhcp3/dhcpd.conf**. Diese Datei kann (neben den unix-/linux-üblichen Editoren) auch webbasiert mit Hilfe des *guitools* **Webmin** (web-based interface for system administration for Unix) durchgeführt werden.

Grundsätzlich gibt es bei *dhcp*-Servern drei Arten von IP-Adressen-Zuordnungen:

> **Dynamisch:** Aus einem bestimmten Adressbereich werden freie Adressen für eine bestimmte Zeit vergeben. Nach Ablauf dieser Zeit – auch während einer Arbeitssitzung - muss der Client versuchen, diese Zuordnung zu verlängern; evtl. wird dann aber auch eine neue Adresse vergeben. Auf diese Weise kann die gleiche IP-Adresse zu verschiedenen Zeiten von verschiedenen Clients genutzt werden.

Automatisch: Jedem Client wird automatisch eine freie IP-Adresse für eine unbegrenzte Zeit fest zugeordnet.

Manuell: Die Zuordnung der IP-Adressen wird von der Systembetreuung fest vorgegeben. Bei einer *dhcp*-Anfrage wird diese Adresse dem Client mitgeteilt. Für den opsi-server wird die manuelle feste Zuordnung als Methode empfohlen, da dies die Administration des Netzes deutlich vereinfacht.

PCs, die eine feststehende IP besitzen, können generell beim Booten dhcp oder bootp nutzen – abhängig davon, was die Netzwerkkarte „kann“. Eine dynamische (d.h. auf Zeit) oder eine automatische IP-Adressen-Zuordnung aus einem festgelegten Adressbereich (= range) kann nur über dhcp und PXE realisiert werden.

BOOTP (Bootstrap Protocol) unterstützt nur eine statische Zuordnung von MAC- und IP-Adressen, die der manuellen Zuordnung bei dhcp entspricht.
Es gibt bei BOOTP nur 2 Typen von Datenpaketen: **BOOTREQUEST** (Client-Broadcast an Server = Anfrage nach IP-Adresse und Parameter an einen Server) und **BOOTREPLY** (Server an Client: Mitteilung der IP-Adresse sowie der angeforderten Parameter).

Der PC kennt zu Beginn seiner Anbindung ans Netz lediglich seine Hardware-Adresse (= hardware ethernet, MAC-Nummer der Netzwerkkarte), bestehend aus sechs zweistelligen Hexadezimalzeichen.

Die Firmeware des PXE wird beim Starten eines PCs aktiv: Sie „kann“ dhcp und schickt damit eine Rundfrage ins Netz. Es ist eine **DHCPDISCOVER**-Anfrage über Standard-Port per Broadcast (= an alle Rechner im Netz): „Ich brauche eine IP-Nummer und wer ist mein dhcp-Server?“.

Mittels **DHCPOFFER** macht der dhcp-Server diesbezüglich einen Vorschlag.

DHCPREQUEST ist die Antwort des Clients an den Server (wenn er die angebotene IP akzeptiert; Hintergrund ist hier: es können in einem Netz mehrere dhcp-Server tätig sein). Der Client fordert damit die angebotene Adresse an.

Mit **DHCPACK** bestätigt der dhcp-Server diese Anforderung des Client. Die Infomationen werden an den Client übertragen.

Weitere Datenpakete:

DHCPNACK	Ablehnung eines DHCPREQUEST durch den Server.
DHCPDECLINE	Ablehnung des Clients, da die IP-Adresse schon verwendet wird.
DHCPRELEASE	Ein Client gibt seine Konfiguration frei (damit steht die IP zur erneuten Vergabe zur Verfügung).
DHCPINFORM	Clientanfrage nach Parametern ohne IP-Adresse.

15.1.2. dhcpd.conf

Die vorliegende Konfigurationsdatei wurde von uns auf wesentliche Informationen und Funktionen reduziert.

- PC-Name,
- Hardware-Ethernet-Adresse,
- IP-Adresse des Gateways,
- Netzmaske,
- IP-Adresse des Bootservers,
- Name des Bootfiles,
- URL der OPSI-Konfigurationsdateien.

Aufbau der dhcpd.conf

Einzelne Anweisungszeilen werden durch ein Semikolon (;) abgeschlossen. Leerzeilen sind erlaubt. Durch eine Raute (#) zu Beginn einer Zeile wird diese auskommentiert (analog wird auch mit der „host description“, einer zusätzlichen Bezeichnung für den PC vor dem eigentlichen host-Name, verfahren.).

Zu Beginn der /etc/dhcp3/dhcpd.conf finden sich allgemeine Parameter.
Im zweiten Teil stehen die Einträge für subnets, groups und hosts. Dadurch wird eine hierachische Gruppierung der Clients innerhalb dieser Textdatei ermöglicht. Blöcke (z.B.: subnet und group) werden durch Klammerung mit geschweiften Klammern gebildet. Die Vorgaben eines Blocks beziehen sich auf alle Elemente (z.B. hosts) innerhalb dieses Blocks.

Allgemeine Parameter/Beispiel

```
# Sample configuration file for ISC dhcpd for Debian
# also answer bootp questions
allow bootp;
```

Das Protokoll BOOTP wird unterstützt.

PC-spezifische Einträge

Jede dhcp-Konfigurationsdatei muss mindestens eine Subnetz-Definition vorweisen. Was in diesem Klammernpaar eingefügt ist, gilt für alle hosts oder groups des entsprechenden Subnetzes.
Durch das Element group können Gruppen von Arbeitsplatzrechnern definiert werden, die bestimmte gemeinsame Eigenschaften haben (so muss z.B. nicht für jeden Host ein Bootfile eingetragen werden).
Wenn Anweisungen mehrfach vorkommen, so gelten immer die innersten.

```
subnet{
.....
        group{
        .....
                host{
                .....
                }
        }
}
```

Beispiel

```
# Server Hostname
server-name "schleppi";
subnet 194.31.185.0 netmask 255.255.255.0{
      option routers 194.31.185.5;
      option domain-name "uib.net";
      option domain-name-servers 194.31.185.14;
#Group the PXE bootable hosts together
   group {
```

Hier ist der Anfang einer Gruppe von PCs innerhalb des subnets;
Beispiel: Gruppe der PCs mit PXE-Netzwerkkarten.

```
# PXE-specific configuration directives...
  # option dhcp-class-identifier "PXEClient";
  # unless you're using dynamic addresses
  filename "linux/pxelinux.0";
```

Alle PC's dieser Gruppe werden einen Linux-Bootfile benutzen, falls in den Einträgen der PCs nicht etwas anderes festgelegt ist.

```
host pcbon13 {
hardware ethernet 00:00:CB:62:EB:2F;
        }
```

Dieser Eintrag enthält lediglich PC-Name und Hardware-Adresse. Die Hardware-Adresse besteht aus sechs zweistelligen Hexadezimalzeichen, die durch einen Doppelpunkt voneinander getrennt werden müssen!!! Klein- und Großschreibung spielt bei den Buchstaben keine Rolle.

```
  }
}
```

Die beiden geschweiften Klammern bilden den Abschluss der Segmente group und subnet.

Soll ein neuer PC im Netz bekannt gemacht werden, muss er in die dhcpd.conf eingetragen werden.

Nach Abspeichern von Änderungen muss der dhcp-Server (als Prozess) gestoppt und wieder gestartet werden, damit die aktuelle Konfiguration auf den laufenden Server angewendet wird:

/etc/init.d/dhcp3-server restart

15.1.3. Werkzeug: dhcp-Administration über Webmin

Da die Syntax der dhcpd.conf etwas unübersichtlich ist, verfügt der opsi-server über ein grafisches Werkzeug zur Administration des dhcp. Die Administration wird hierbei über das bekannt Web-Basierte Systemverwaltungswerkzeug webmin bereitgestellt.
Da auf dem opsi-server selbst kein Browser installiert ist, verwenden Sie im Folgenden den Standardbrowser Ihres PC's.

Der Webmindienst sollte nach Start des opsi-servers laufen. Sollte er nicht laufen, so starten Sie ihn mit root-Rechten mit **`/etc/init.d/webmin start`**.

Aufruf: Nach Aufruf über Ihren Browser (https://<servername>:10000/) und Eingabe von Benutzername und Passwort (Voreinstellung: admin/linux123) erscheint der Eröffnungsbildschirm von Webmin:

Abbildung 31: Startfenster des Werzeugs Webmin

Die Voreinstellungen für username und Passwort sind im Kapitel 'Installation und Inbetriebnahme' beschrieben. Bitte ändern Sie diese Voreinstellungen nach der Inbetriebnahme.

Die Arbeitssitzung wird mit einem Timeout gesteuert: Authentifizierung und Passwort-Timeout ist standardmäßig eingestellt auf 5 Minuten.

Hauptebene der Programmsteuerung sind die oberen drei Buttons (Webmin, System, Server); die ausgewählte Option erscheint auf dem Webmin-Desktop als Registerkarte und stellt weitere Auswahlmöglichkeiten ebenfalls als Buttons dar.

Wir haben die Funktion von webmin soweit als möglich auf die Administration von dhcp beschränkt.

Button **Webmin** enthält: Ändern der Sprache und des Designs, Webmin Ereignisanzeige (Protokollierung von Dateiänderungen möglich!), Webmin-Benutzer (Benutzerverwaltung), Webmin-Konfiguration (wenn z.B. der Host, auf dem Webmin läuft, mehrere IP-Adressen hat, kann dies hier verwaltet werden).
Button **System**: Geplante Cron-Aufträge.
Button **Server**: dhcp-Server (In diesem Skript wird das Hauptaugenmerk auf diesen Punkt gelegt.)

dhcp-Server/Netzübersicht: Wird der Button dhcp-Server angeklickt und auf der Index-Registerkarte der dhcp-Server ausgewählt, erscheint in einer Dreiteilung des Bildschirms die Übersicht:

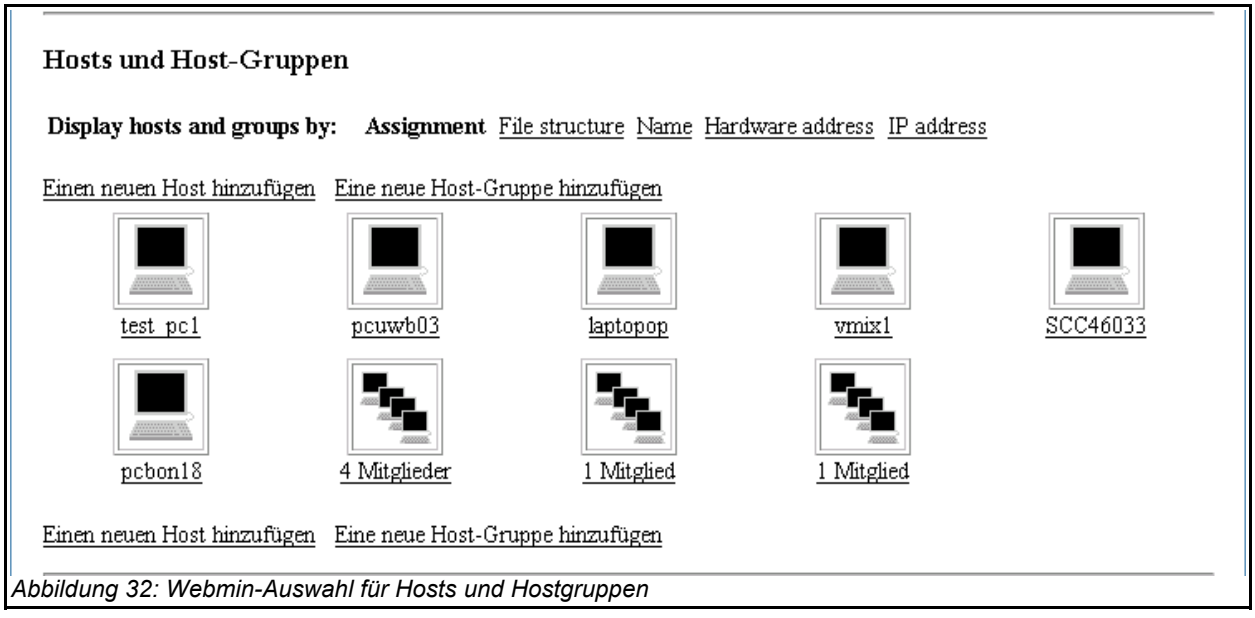


Abbildung 32: Webmin-Auswahl für Hosts und Hostgruppen

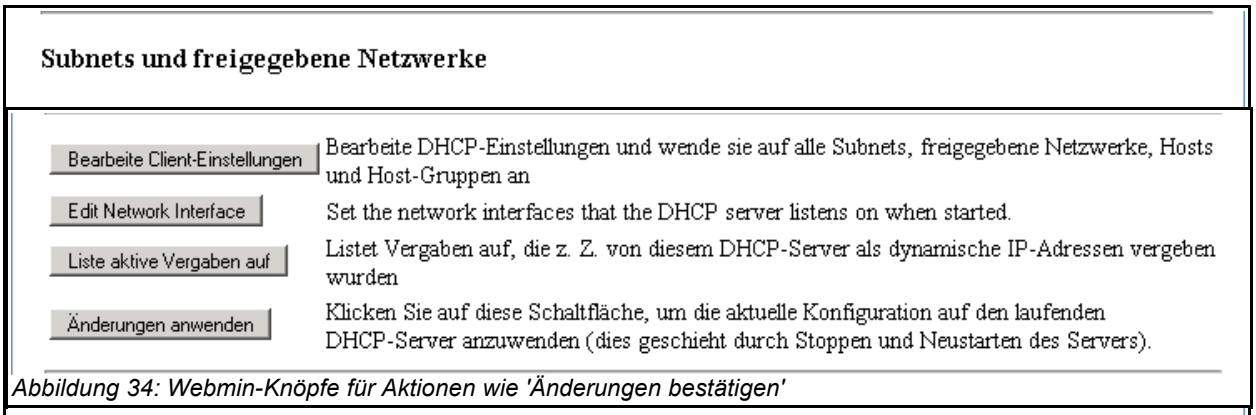


Abbildung 34: Webmin-Knöpfe für Aktionen wie 'Änderungen bestätigen'

Abbildung 33: Webmin-Auswahl für Netze und Subnetze

Im oberen Teil wird das subnet dargestellt.

Im mitteleren Bereich werden die PCs und die PC-Gruppen angezeigt.

Im unteren Teil gibt es direkte Befehlsmöglichkeiten.

Darstellung ändern: Im linken oberen Rand der Netzübersicht wird eine umgekehrte Registerkarte **„Modul-Konfiguration“** angezeigt:

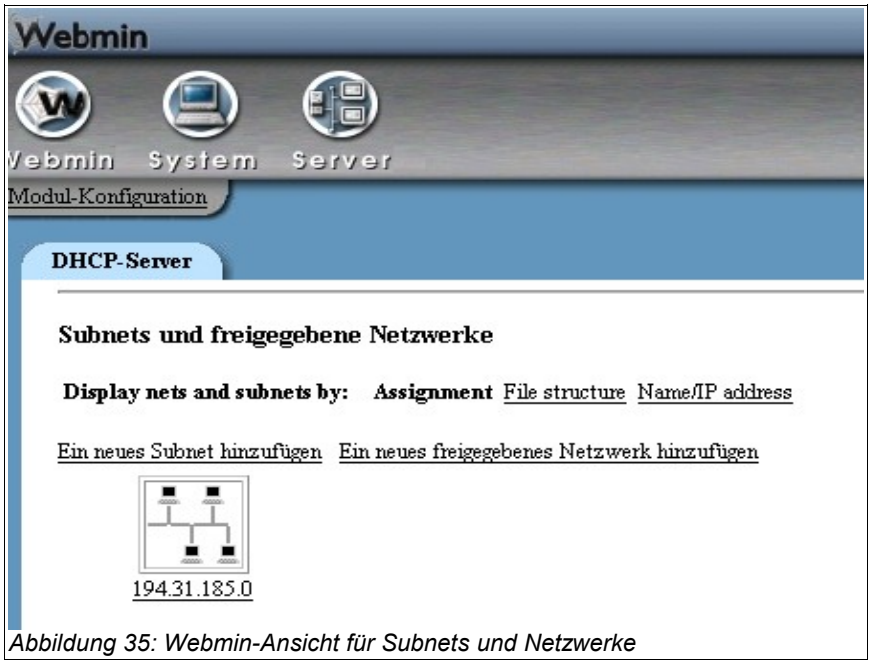


Abbildung 35: Webmin-Ansicht für Subnets und Netzwerke

Sie stellt den Zugang zu allen konfigurierbaren Einstellungen des dhcp-Servers dar. Neben Darstellungsoptionen für die Netzübersicht (oberer Bereich: „Configurable options“) werden unter „System configuration“ die grundlegenden Einstellungen für den dhcp-Server getätigt.

Empfehlung: Die Netzwerkübersicht wird übersichtlicher, wenn in **„Modul-Konfiguration“** folgende Optionen eingegeben werden:

Zeige Sub-Netze und Hosts als	**Liste**
Show object descriptions instead of names?	**No**
Show group names as	**Name or member count**

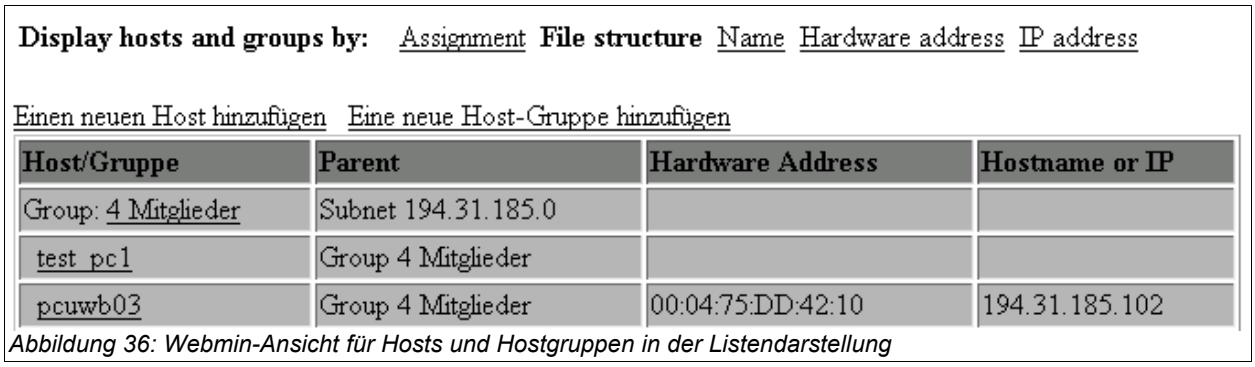


Abbildung 36: Webmin-Ansicht für Hosts und Hostgruppen in der Listendarstellung

Jetzt ändert sich die Darstellung der Netzwerkübersicht:
Direkt vor und nach der Liste zeigen Schaltflächen den Weg, um
„Einen neuen Host hinzufügen“ und „Eine neue Host-Gruppe hinzufügen“.

15.1.3.1. PC-Eintrag erstellen

Nach Anklicken der Schaltfläche „Einen neuen Host hinzufügen“ können die wesentlichsten Merkmale eines PC's in die Maske eingetragen werden:

Eingaben:

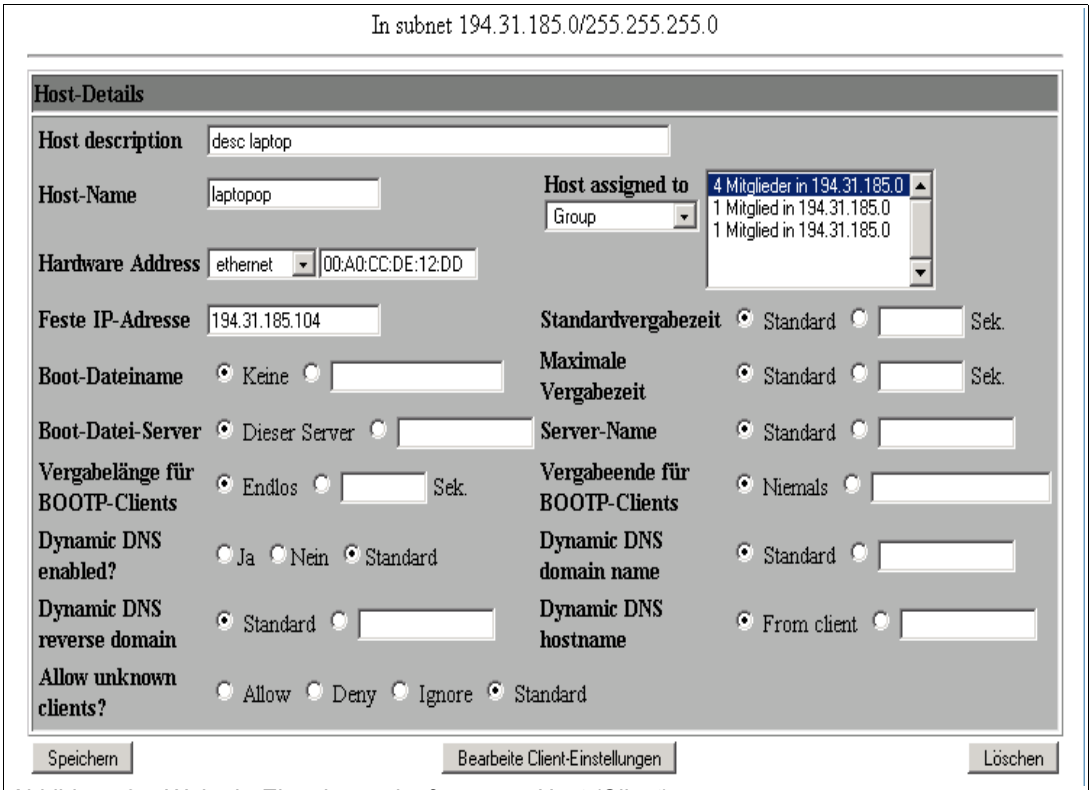


Abbildung 37: Webmin-Eingabemaske für neuen Host (Client)

- gegebenenfalls: Beschreibung des PC's: **Host description**
- PC-Name: **Host-Name**
- MAC-Adresse: **Hardware Address** (Doppelpunkte nicht vergessen!!)
- **Feste IP-Adresse**
- gegebenenfalls: Bootfile = **Boot-Dateiname**
- gegebenenfalls: Zugehörigkeit in eine Gruppe: **Host assigned to ...**

„Änderungen anwenden“: Nach dem Abspeichern von Änderungen in der dhcpd.conf müssen diese Änderungen dem dhcp-Server mitgeteilt werden:

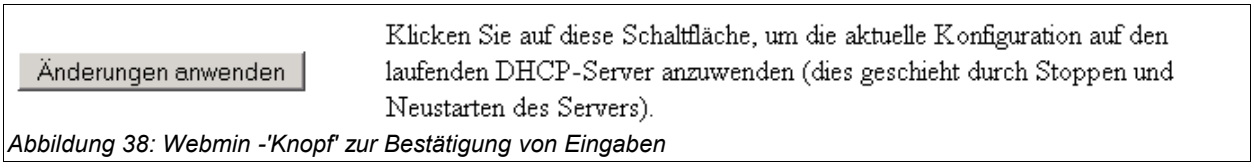


Abbildung 38: Webmin -'Knopf' zur Bestätigung von Eingaben

Dieser Button „Änderungen anwenden“ muss nach Änderungen angeklickt werden; er entspricht dem Befehl: `/etc/init.d/dhcp3-server restart`

15.1.3.2. Neue Gruppe bilden

Innerhalb eines Subnet's kann eine neue Gruppe gebildet werden, z.B.: Zusammenfassung nach Art der Netzwerkkarten, nach den Bootfiles...

15.2. opsi V3: opsi Konfigurations API, opsiconfd und backendmanager

In opsi V3 wird ein Python basierte Konfigurations-API zur Verfügung gestellt. Diese API stellt von der konkreten Datenhaltung (Backends) abstrahierte Aufrufe zur Konfiguration von opsi Bereit. Weiterhin werden Aufrufe zur Verwendung unterschiedlicher konkreter Backends bereitgestellt, die nur opsi-intern genutzt werden. Welche konkreten Backends wie genutzt werden wird über den backendmanager (/etc/opsi/backendmanager.d/) gesteuert.

Die Konfigurations-API wird auf der Kommandozeile durch das Werkzeug 'opsi-admin' bereitgestellt. In dem entsprechenden Kapitel zu diesem Werkzeug wird auch ein ausführlicher Überblick über die Funktionen der API gegeben.

Damit andere Komponenten von opsi wie z.B. die grafischen Konfigurationswerkzeuge, der opsi-Winst oder das opsi-bootimage mit dieser API arbeiten können wird diese über einen Webservice bereitgestellt. Der Webservice basiert nicht auf XML/Soap sondern auf dem deutlich schlankeren Standard JSON (www.json.org). Bereitgestellt wird dieser Webservice durch das Programm opsiconfd. Erreichbar ist der Webservice per default über https Port 4447. Der Webservice bietet auch ein rudimentäres Browserinterface mit dem die Funktion des Webservice erkunden kann.

Der opsiconfd läuft mit den Rechten des users pcpatch. Von daher muss dieser user auch die notwendigen Rechte zum Zugriff auf die Informationen besitzen. Dies ergibt eine Reihe von Rechteänderungen im Vergleich zu opsi V2.

Konfiguriert wird der opsiconfd über die Datei /etc/opsi/opsiconfd.conf.

Der opsiconfd logt per default nach /var/log/opsi/opsiconfd. Dabei legt der opsiconfd für jeden Client ein gesonderte Logdatei an.

16. Glossar

Actionrequest	Ab opsi V3 werden der derzeitige Installationsstatus und die nächste geplante Aktion (Actionrequest) getrennt betrachtet. Typische Actionrequests sind 'setup', 'deinstall' und 'update'. -> Installationsstatus
Backend	opsi V3 unterstützt unterschiedliche Methoden der Datenhaltung wie File oder LDAP. Diese werden als Backends bezeichnet und über den -> backendmanager konfiguriert.
backendmanager	Programm / Konfigurationsdatei in der festgelegt wird, welche Daten wie und wo gespeichert werden.
bootp	Aus dem Unix-Bereich kommendes Protokoll, mit dem ein Client in der Regel über ein →Bootprom von einem Server die Konfigurationsdaten eines PC's (wie z.B. Netzwerkadresse) abfragt.
Bootprom	Ein auf der Netzwerkkarte liegender Festspeicher (PROM: programmable read only memory), dessen Code beim Booten des Rechners ausgeführt wird. Wird eingesetzt, um Konfigurationsdaten von einem Server abzufragen. Hierzu werden die Protokolle →bootp oder →dhcp verwendet.
clientId	Eindeutige Bezeichnung des Clients durch Verwendung des 'full qualified hostnames' also IP-Name inclusive Domain z.B. dpvm02.uib.local

config-server	die Funktionalität welche die Haltung, Bereitstellung und Managment von Konfigurationsdaten auf einem ->opsi-server verwirklicht.
depot-server	die Funktionalität welche die Bereitstellung von Software-Depots auf shares für die Clients und einen tftpbereich für die PXE-Boots auf einem ->opsi-server verwirklicht. Vor opsi 3.3 allgemein für opsi-server verwendet.
dhcp	Dynamic Host Configuration Protocol: Ein Protokoll, das im Prinzip eine Erweiterung des →bootp-Protokolls darstellt und eine dynamische Vergabe von IP-Nummern ermöglicht.
ftpd	File Transfer Protocol-Dämon: Erlaubt das Einloggen von anderen Rechnern und das Übertragen von Dateien von und zu diesem. **telnetd** (Telnet-Dämon) erlaubt Terminalverbindungen von anderen Rechnern.
GINA	Graphic INteractive Authentification Ein Programm, welches unter MS-Windows den Loginvorgang durchführt. Per Voreinstellung ist dies die msgina.dll. Soll der Loginvorgang durch weitere Funktionalitäten erweitert werden, so können der msgina weitere ginas zur Seite gestellt werden. Der opsi-Loginblocker ist über eine pgina.dll realisiert (vgl. http://pgina.xpasystems.com/).
GNU	Die Abkürzung GNU steht für "GNU's not Unix" (Diese Art von rekursiven Abkürzungen ist im Computerbereich recht beliebt.). Das GNU-Projekt wurde 1983 von Richard Stallman, dem Gründer der Free Software Foundation, ins Leben gerufen, um ein freies, unixartiges Betriebssystem zu entwickeln.

	Zwar ist dieses Betriebssystem nach wie vor nicht fertiggestellt, aber das Projekt hat eine Fülle von Werkzeugen hervorgebracht, die die Entstehung des freien Betriebssystems Linux erst möglich gemacht haben. Daher wird Linux auch häufig unter der Bezeichnung GNU/Linux geführt.
GUI	Graphical User Interface: Grafische Schnittstelle zum Anwender
hostId	Eindeutige Bezeichnung eines Rechners durch Verwendung des 'full qualified hostnames' also IP-Name inclusive Domain z.B. dpvm02.uib.local
inetd	Internet-Dämon: Startet die meisten der Internet-Dienste erst bei Bedarf, z.B. bootpd, ftpd, tftpd, telnetd. inetd muss u.a. „wissen“, welches Programm er mit welchen Optionen starten muss, wenn eine Anfrage eines anderen Rechners kommt. Diese Informationen findet er in /etc/inetd.conf.
Installationsstatus	Ab opsi V3 werden der derzeitige Installationsstatus und die nächste geplante Aktion (Actionrequest) getrennt betrachtet. Typische Installationsstati sind 'installed' und 'not installed'. -> Actionrequest
IP	Eine IP-Adresse: Zentrale Vergabe einer Adresse, um innerhalb des Internets eine eindeutige Adresszuordnung zu gewährleisten. Die IP ist eine 32-Bit lange Zahl und besteht aus zwei Teilen: Einer Netzadresse und der Adresse des Rechners innerhalb des Netzes. Üblicherweise wird die 32-Bit Zahl als 4 Zahlen zwischen 0 und 255 dargestellt, die durch Punkte voneinander getrennt

	sind. Je nach Größe des Netzwerkes wird die IP einer der 3 wichtigsten Klassen zugeordnet: A, B oder C. Die Klassen unterscheiden sich nach Anzahl der verfügbaren Netzwerke und Anzahl der möglichen Hosts eines Netzwerkes. Ein Klasse A-Netz hat als erste Nummer eine Zahl zwischen 1 und 127. Die restlichen drei Zahlen bleiben für die Rechneradresse. Ein Klasse B-Netz hat als erste Nummer eine Zahl zwischen 128 und 191. Die zweite Zahl gehört mit zur Netznummer. Ein Klasse C-Netz hat als erste Nummer eine Zahl zwischen 192 und 223. Die zweite und dritte Zahl gehören mit zur Netznummer. Die letzte Zahl bleibt für die Rechneradresse.
JSON	**JSON**, kurz für **J**ava**S**cript **O**bject **N**otation und gesprochen wie der Name Jason, ist ein kompaktes Computer-Format in für Mensch und Maschine einfach lesbarer Textform zum Zweck des Datenaustauschs zwischen Anwendungen. Quelle: http://de.wikipedia.org/wiki/JSON siehe auch www.json.org
LastSeen	Zeitstempel wann ein Client sich zuletzt über den Service bei opsi gemeldet hat.
localboot Produkt	Ein opsi Paket welches über den opsi-preloginloader installiert wird.
MAC	Media Access Control bezeichnet eine weltweit eindeutige Nummer der Netzwerkkarte, die bei jeder Datenübertragung mitgesendet wird. Anhand dieser Adresse, kann man den PC (genauer: dessen Netzwerkkarte) eindeutig identifizieren und so die →IP-Nummer automatisch verteilen o.ä.. Die Nummer setzt sich aus 6 zweistelligen Hexadezimalzahlen

	zusammen, die durch einen Doppelpunkt voneinander getrennt sind.
netboot Produkt	Ein opsi Paket welches über den Start eines bootimages ausgeführt wird.
opsi	open pc server intergration
opsi-admin	opsi V3 Kommandozeilen Interface zur opsi-Konfiguration
opsi-Configed	opsi V3 Konfigurationswerkzeug als Java Applikation und Applet
opsi-preLoginLoader	Agent der auf dem Windowsclient installiert ist und dort zentral gesteuert Software installiert.
opsi-server	Ein Server der im Rahmen von opsi Dienste bereitstellt. Üblicherweise -> config-server und -> depot-server.
opsiconfd	Deamon der die opsi-Konfigurations-API als JSON basierten Webservice zur Verfügung stellt
opsiHostKey	siehe pckey
pckey	Ein String der dem Client bei der (preloginloader-) Installation zugewiesen wird und gleichzeitig auf dem Server gespeichert wird. Wird zur Authentifizierung verwendet und darf daher nicht frei zugänglich sein. →opsiHostKey
PDC	Primary Domain Controller: Vereinfacht= Authentifizierungsserver in einem Microsoft-NT-Netzwerk
pgina	siehe GINA
preloginloader	-> opsi-preLoginLoader

product properties	Zusätzliche Einstellungen zu einem opsi-Produkt die clientspezifisch gesetzt werden können und bei der Installation ausgewertet werden.
Produkt-ID	Eindeutiger Bezeichner eines opsi-Produkts. Dieser darf keine Leerzeichen oder Sonderzeichen (außer Bindestrich enthalten). In opsi V2 auch als -> Produktname was aber in opsi V3 eine andere Bedeutung hat. Beispiel für eine ProductId: acroread
Produktname	In opsi V3 der Klartextname eines Produktes. Beispiel für einen Produktnamen: 'Adobe Acrobat Reader'
PXE	Preboot eXecution Environment: Standard für Bootproms. Eingesetzt wird dabei in der Regel nicht →bootp, sondern →*dhcp*.
SAMBA	Freie Software, um unter Unix Dienste für das von Microsoft-Clients verwendet Protokoll →SMB anzubieten. Mit Hilfe des Paketes SAMBA können Unix-Server SMB verstehen.
Server Produkt	Ein opsi Produkt welches Installationen auf dem Server ausführt die nicht für den Client bestimmt sind.
SMB	Server Message Block: Protokoll von Microsoft, um Netzwerklaufwerke und Authentifizierung anzubieten. Wird neuerdings von Microsoft auch gerne als CIFS (Common Internet File System) bezeichnet.
Subnets	Falls man ein großes Netz hat, besteht oft die Notwendigkeit, das eigene Netz in Unternetze (Subnets) aufzuteilen, da nicht alle Rechner zum gleichen lokalen Netz gehören können. Hierbei wird ein beliebig großer Teil des Rechnerteils der IP-Adresse als Subnetzteil definiert. Die IP-Adresse gliedert sich damit in eine Netzadresse, eine

	Subnetzadresse und eine Rechneradresse. Mittels einer Subnetzmaske (Subnetmaske) wird bestimmt, welcher Teil zur Netz- und Subnetzadresse und welcher Teil zur Rechneradresse gehört, indem die Bits des Netz- und Subnetzteils auf Eins gesetzt werden und die Bits des Rechnerteils auf Null.
TCP/IP	Transmission Control Protocol/Internet Protocol: Das aus der Unix-Welt kommende, grundlegende Netzwerkprotoll, welches inzwischen allgemeine Verbreitung gefunden hat.
tftpd	tftp-Dämon (Trivial File Transfer Protocol) erlaubt das Übertragen von Dateien von und zu anderen Rechnern ohne Login-Prozedur, allerdings mit restriktiven Zugangsbeschränkungen. So dürfen nur solche Dateien gelesen oder geschrieben werden, die aufgrund ihrer Zugriffsrechte von allen Benutzern lesbar oder beschreibbar sind und die im Verzeichnisbaum /tftpboot liegen. Die PCs benutzen tftp, um die Bootmenüs und die Bootdateien vom Server zu holen.

17. Abbildungsverzeichnis

Abbildungsverzeichnis

17. Abbildungsverzeichnis

18. Änderungen

Änderungen in diesem Handbuch.

18.1. opsi 2.4 zu opsi 2.5

- Verwendung von https beim Webconfigurationseditor
- Kapitel zur Integration von Treibern in die automatische Betriebssysteminstallation
- Kapitel zur nachträglichen Installation des Preloginloaders
- Verweise auf opsi-wiki
- Verweise auf opsi-bootimage Handbuch
- Aufstellung der opsi-Logdateien

18.2. Nachtrag opsi 2.5 (25.09.06)

- Verschiebung der Option 'askBeforeInst' in der global.sysconf von der General-Sektion zu den Produkt-Sektionen
- Beschreibung zum neuen Schalter 'textcolor' (zum Anpassen des Winst) eingefügt

18.3. Nachtrag opsi 2.5 / opsi 3.0 (08.12.06)

- Registryeintrag button_stopnetworking liegt unter opsi.org/pcptch

18.4. Änderungen opsi 3.0 (1.2.07)

1. Kapitel: Unterschiede der opsi Version 3 zu Version 2
2. Kapitel: Programme in /opt/bin
3. Erweiterung: Konfigurationsdateien in /etc/opsi

4. Neue Einträge in der Registry
5. Erweiterung: Steuerdatei für die Softwareverteilung: <pcname>.ini
6. Kapitel: *.sysconf-Dateien
7. Kapitel Dateien in /etc/init.d
8. Kapitel /etc/group
9. Kapitel: Werkzeug: opsi-admin
10. Kapitel: Werkzeug: opsi V3 opsi-Configed
11. Kapitel: Werkzeug: opsi V3 opsi-Webconfigedit
12. Kapitel: Werkzeug: opsi V3 opsi-admin
13. Kapitel Logdateien unter /var/log und /var/log/opsi
14. Erweiterung des Glossars
15. Erweiterung: Nachträgliche Installation des opsi-PreLoginLoaders: Jeder Client brauch einen Eintrag in der /etc/opsi/pckeys

18.5. Nachträge opsi 3.0

1. 12.4.07: LDAP-Kapitel

18.6. Änderungen opsi 3.1 (15.6.07)

Kapitel Unterschiede 3.1

Kapitel File31 Backend

Löschen: Werkzeug reinstmanager

opsi-admin task setPcpatchPassword

opsi-admin Client Bootimage aktivieren

Aktualisierung der Beschreibung des opsi-configed

Aktualisierung des Kapitels zum Initialen Rollout des preloginloaders

Aktualisierung des Kapitels zur Treiberintegration

opsi-admin: neue Methoden:

method authenticated
method checkForErrors
method deleteProductProperties productId *objectId
method deleteProductProperty productId property *objectId
method deleteServer serverId
method getHost_hash hostId
method getNetBootProductIds_list
method getPossibleProductActionRequests_list
method setPXEBootConfiguration hostId *args
method setPcpatchPassword hostId password
method unsetPXEBootConfiguration hostId

18.7. Änderungen opsi 3.2 (21.11.07)

Aktualisierung des Kapitels „Vereinfachte Treiberintegration mit Symlinks“ zur Treiberintegration (**`download_driver_pack.py`** und **`preferred`**)

Kapitel zur Inventariseirung

Integration von Teilen des Integrationshandbuchs

Umstrukturierung des Handbuchs.